

# ویکی توسعه‌دهندگان بازی‌س

ورود با بازی‌س، گیتا کوین، صفحه‌ی جوایز و پرداخت بازی‌س را به بازی‌تان اضافه کنید.

1. مقدمه
2. شروع کار، قدم‌به‌قدم
3. API پلتفرم، قدم‌به‌قدم
4. صفحه‌های بازی‌س و پیام‌هایشان
5. پرداخت بازی‌س، قدم‌به‌قدم
6. ورود و اتصال حساب بازی‌س
7. گیتا کوین
8. صفحه‌ی جایزه («انتخاب جایزه»)
9. صفحه‌ی پیشنهاد آموزش (game-offer)
10. لینک‌ها و پارامترهای UTM
11. مهاجرت به کد اتصال و توکن‌های کوتاه‌عمر
12. کد سرور
13. کد کلاینت
14. شیت پایین (وب‌ویوی داخل بازی)
15. بازی نمونه – سنگ، کاغذ، قیچی
16. چک‌لیست انتشار و رفع اشکال
17. مرجع API

## مقدمه

بازینس چه چیزهایی به بازی شما می‌دهد، اجزای آن چطور کنار هم کار می‌کنند و این ویکی چطور مرتب شده است.

## بازینس چیست

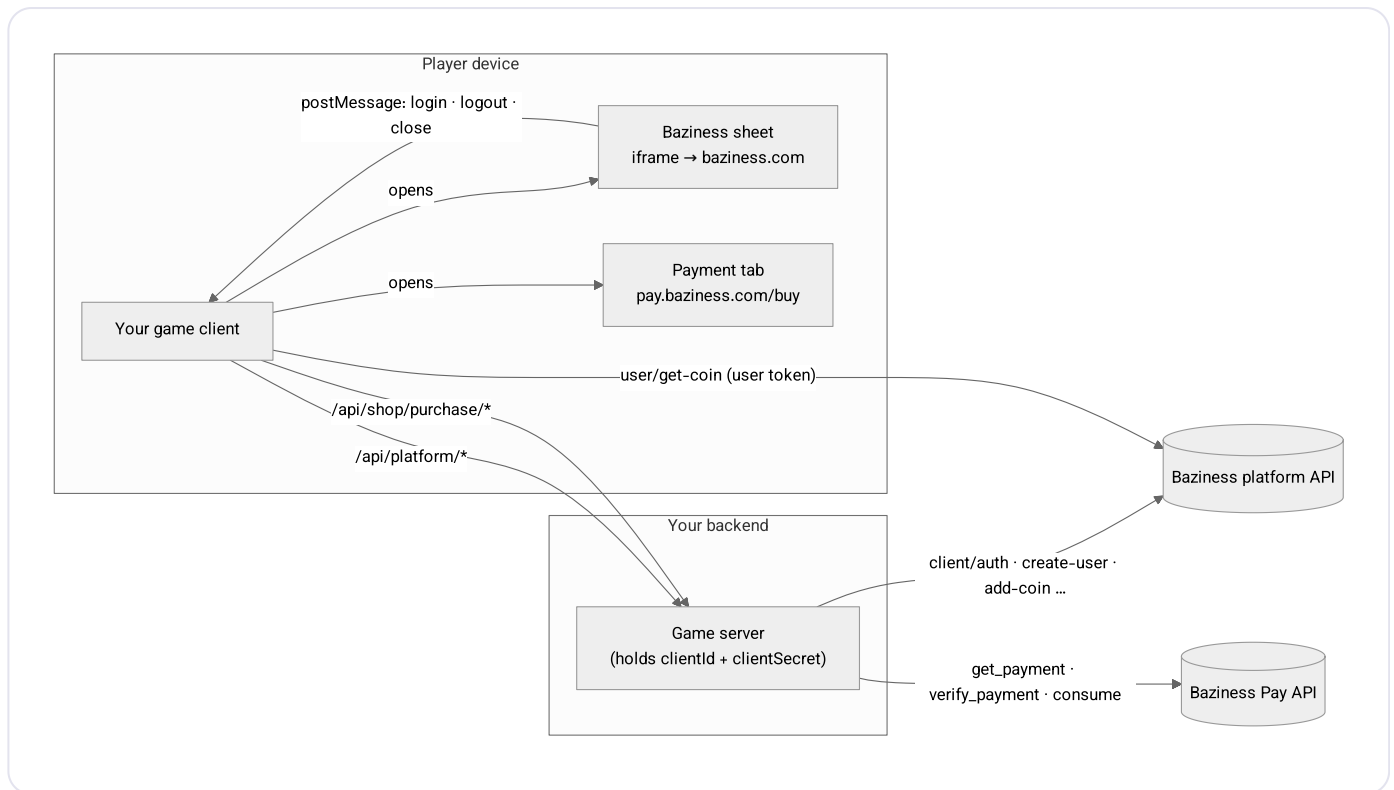
بازینس پلتفرم بازی گیتا گیمز است. وقتی بازی شما به آن وصل شود، بازیکنان این‌ها را می‌گیرند:

- **حساب بازینس:** با شماره موبایل وارد می‌شوند و همان حساب در همه‌ی بازی‌های بازینس کار می‌کند.
- **گیتا کوین:** واحد پول پلتفرم. بازیکن آن را در بازی‌ها به دست می‌آورد و برای جایزه‌های واقعی خرج می‌کند.
- **صفحه‌های بازینس داخل بازی شما:** ورود، پروفایل، فروشگاه جایزه‌ی «انتخاب جایزه» و صفحه‌ی پیشنهاد مخصوص بازی شما.
- **پرداخت بازینس:** خرید با پول واقعی از طریق صفحه‌ی پرداخت بازینس.

این‌ها دو یکپارچه‌سازی جدا از هم هستند. می‌توانید هر کدام را به‌تنهایی یا هر دو را با هم اضافه کنید:

| یکپارچه‌سازی  | چیزی که بازیکن می‌گیرد           | چه کسی با بازینس حرف می‌زند                                                                      | چیزهایی که لازم دارید                               |
|---------------|----------------------------------|--------------------------------------------------------------------------------------------------|-----------------------------------------------------|
| پلتفرم بازینس | حساب، گیتا کوین، صفحه‌های بازینس | سرور شما API پلتفرم را صدا می‌زند؛<br>کلاینت شما صفحه‌ها را نشان می‌دهد و موجودی سکه را می‌خواند | شناسه‌ی کلاینت، کلید محرمانه، نام بسته‌ی پلتفرم     |
| پرداخت بازینس | خرید با پول واقعی                | سرور شما API پرداخت را صدا می‌زند؛<br>کلاینت شما صفحه‌ی پرداخت را باز می‌کند                     | نام بسته‌ی پرداخت، SKUهای ثبت‌شده در داشبورد بازینس |

## اجزا چطور کنار هم کار می‌کنند



کلاینت فقط در دو مورد مستقیم با بازی‌نس حرف می‌زند: **صفحه‌های** بازی‌نس را نشان می‌دهد، و موجودی سکه را با توکن خود بازیکن می‌خواند ( `user/get-coin` ). همه‌ی کارهای دیگر از **سرور شما** می‌گذرند، چون فقط سرور شما کلید محرمانه را دارد.

## پنج قانون طلایی

1. کلید محرمانه هیچ‌وقت از سرور بیرون نمی‌رود. مرورگر یا اپ فقط تنظیمات عمومی و توکن خود بازیکن را می‌گیرد.
2. سرور تصمیم می‌گیرد و کلاینت فقط نمایش می‌دهد. اتصال حساب، دادن سکه و خرید همیشه در سرور بررسی می‌شوند.
3. هر لینک صفحه‌ی بازی‌نس `packageName` و `sdkType=js` دارد. بدون `sdkType=js` صفحه نمی‌تواند با بازی حرف بزند و حتی نمی‌تواند خودش را ببندد.
4. بازی بدون بازی‌نس هم کار می‌کند. اگر پلتفرم خاموش یا در دسترس نبود، بخش‌های بازی‌نس را پنهان کنید و ادامه دهید.
5. هر پرداخت یک شناسه‌ی یکتا و تکرارپذیر دارد. تلاش دوباره، دو بار زدن دکمه یا کرش سرور هیچ‌وقت نباید دو بار پاداش بدهد.

| توسعه (بازی‌های تستی) | اصلی                                                     |                                                     |
|-----------------------|----------------------------------------------------------|-----------------------------------------------------|
| API پلتفرم            | <code>https://platform-dev.karizmastudios.org/api</code> | <code>https://platform.karizmastudio.com/api</code> |
| صفحه‌های بازینس       | <code>https://platform-web-dev.karizmastudios.org</code> | <code>https://baziness.com</code>                   |
| پرداخت بازینس         | <code>https://pay-dev.karizmastudios.org</code>          | <code>https://pay.baziness.com</code>               |
| ورود                  | هر شماره موبایلی، کد 111111 (پیامکی فرستاده نمی‌شود)     | شماره‌ی واقعی و کد پیامکی                           |
| خرید تستی             | «خرید آزمایشی» در صفحه‌ی پرداخت (بدون پرداخت واقعی)      | پرداخت واقعی                                        |

هر محیط شناسه و کلید محرمانه‌ی خودش را دارد.

## این ویکی چطور مرتب شده

| بخش        | محتوا                                                                                                | کد دارد؟           |
|------------|------------------------------------------------------------------------------------------------------|--------------------|
| شروع کار   | همین صفحه و یکپارچه‌سازی قدم به قدم                                                                  | نه                 |
| API        | همه‌ی endpointها به ترتیب: چه کار می‌کند، کی صدایش بزنید، درخواست، پاسخ و قانون‌ها                   | فقط درخواست و پاسخ |
| راهنماها   | جریان‌ها: ورود، گیتا کوین، صفحه‌ی جایزه، صفحه‌ی پیشنهاد آموزش، لینک‌ها                               | نه                 |
| نمونه کد   | کد آماده‌ی سرور (TypeScript، C#، Python) و کلاینت (وب، Cocos، React، Creator، Unity)، از جمله وب‌ویو | بله                |
| بازی نمونه | یک بازی کامل و متن‌باز که همه‌ی این‌ها را دارد: اجرا و دانلود                                        | کل پروژه           |
| مرجع       | چک‌لیست انتشار، رفع اشکال و همه‌ی endpointها در یک جدول                                              | نه                 |

هر صفحه را با دکمه‌های زیر عنوانش به صورت **Markdown** یا **PDF** دانلود کنید. راهنمای کامل هم در هر دو قالب در نوار کناری هست.

# شروع کار، قدم به قدم

چه چیزهایی از تیم پلتفرم بگیرید، هر مقدار کجا استفاده می‌شود، و کل یکپارچه‌سازی در چند قدم کوتاه که هر کدام به جزئیاتش لینک دارد.

## ۱. کلیدها را بگیرید

تیم پلتفرم بازی‌نس این مقادارها را به شما می‌دهد، برای هر محیط یک دسته (توسعه و اصلی):

| مقدار                        | مثال                      | چه کسی استفاده می‌کند                                                   | محرمانه؟                 |
|------------------------------|---------------------------|-------------------------------------------------------------------------|--------------------------|
| شناسه‌ی کلاینت (client id)   | 12                        | سرور شما، برای گرفتن توکن کلاینت                                        | نه، ولی خصوصی نگهش دارید |
| کلید محرمانه (client secret) | .....                     | سرور شما، برای گرفتن توکن کلاینت                                        | بله، فقط سرور            |
| نام بسته‌ی پلتفرم            | com.karizmastudio.example | لینک همه‌ی صفحه‌های بازی‌نس (packageName)                               | نه                       |
| نام بازی                     | example                   | صفحه‌ی جایزه و صفحه‌ی پیشنهاد (game_name)                               | نه                       |
| نام بسته‌ی پرداخت            | com.karizmastudio.example | همه‌ی درخواست‌های پرداخت (pname)؛ ممکن است با نام بسته‌ی پلتفرم فرق کند | نه                       |
| SKUها                        | example_pack              | محصولاتی که با پرداخت بازی‌نس می‌فروشید                                 | نه                       |

### هشدار

کلید محرمانه هیچ‌وقت نباید به مرورگر یا بیلد اپ برسد: نه در کد کلاینت، نه در متغیرهای `VITE_*` یا `NEXT_PUBLIC_*`، نه در گیت. هر کسی که آن را داشته باشد می‌تواند به نام بازی شما بازیکن بسازد و گیتا کوین بدهد.

## ۲. دو نوع توکن را بشناسید

همه‌چیز در پلتفرم با یکی از این دو توکن کار می‌کند. جدا نگه داشتن این دو، کلید کل یکپارچه‌سازی است:

| توکن        | چه کسی دارد                 | چطور گرفته می‌شود                                                        | برای چه                                                                                         |
|-------------|-----------------------------|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| توکن کلاینت | فقط سرور شما                | شناسه + کلید محرمانه ←<br><code>client/auth</code>                       | همه‌ی درخواست‌های سرور به سرور:<br>ساختن بازیکن، بررسی توکن، دادن سکه                           |
| توکن کاربر  | کلاینت بازیکن (از سرور شما) | <code>client/create-user</code> یا<br><code>client/get-user-token</code> | باز کردن صفحه‌های بازیکن به نام همین بازیکن ( <code>linkedToken</code> ) و خواندن موجودی سکه‌اش |

### ۳. قدم‌ها را دنبال کنید

| # | قدم                                                                                                                     | کجا    | جزئیات                             |
|---|-------------------------------------------------------------------------------------------------------------------------|--------|------------------------------------|
| ۱ | سرور شما با شناسه و کلید محرمانه یک توکن کلاینت می‌گیرد                                                                 | سرور   | API پلتفرم ← قدم ۱                 |
| ۲ | اولین باری که بازیکن می‌آید، سرور کاربر بازیکن او را می‌سازد و شناسه‌اش را ذخیره می‌کند. دفعه‌های بعد توکن تازه می‌گیرد | سرور   | قدم‌های ۲ و ۳                      |
| ۳ | کلاینت صفحه‌های بازیکن (ورود، پروفایل، جایزه) را با توکن کاربر نشان می‌دهد و به پیام‌هایشان گوش می‌دهد                  | کلاینت | صفحه‌های بازیکن                    |
| ۴ | وقتی بازیکن در صفحه وارد شد، سرور توکن را بررسی می‌کند و حساب را وصل می‌کند (یا به حساب دیگر بازیکن جابه‌جا می‌کند)     | سرور   | ورود، قدم ۴                        |
| ۵ | کلاینت موجودی گیتا کوین را نشان می‌دهد و سرور برای برد و جایزه گیتا کوین می‌دهد                                         | هر دو  | گیتا کوین، قدم‌های ۵ و ۶           |
| ۶ | دکمه‌ی «انتخاب جایزه» صفحه‌ی جایزه را باز می‌کند؛ آموزش می‌تواند صفحه‌ی پیشنهاد بازی را باز کند                         | کلاینت | صفحه‌ی جایزه، صفحه‌ی پیشنهاد آموزش |
| ۷ | آیتم‌ها را با پرداخت بازیکن بفروشید                                                                                     | هر دو  | پرداخت بازیکن                      |
| ۸ | لینک‌هایی را که بازیکن می‌آورند با پارامترهای UTM برچسب بزنید                                                           | هر دو  | لینک‌ها و UTM                      |

کد آماده‌ی همه‌ی قدم‌ها در کد سرور و کد کلاینت است. بازی نمونه همه را کنار هم دارد.

### ۴. سرور را تنظیم کنید

بازی نمونه و نمونه کدها این تنظیمات را می‌خوانند. همین نام‌ها را نگه دارید تا نمونه‌ها بدون تغییر کار کنند:

| تنظیم                            | مثال                                   | معنی                                                                    |
|----------------------------------|----------------------------------------|-------------------------------------------------------------------------|
| BAZINESS_PLATFORM_ENABLED        | true                                   | کلید اصلی. تا وقتی<br>false است بازی<br>هیچ بخش بازیابی<br>نشان نمی‌دهد |
| BAZINESS_PLATFORM_USE_PRODUCTION | false                                  | آدرس‌های = false<br>توسعه، = true<br>آدرس‌های اصلی                      |
| BAZINESS_CLIENT_ID               | 12                                     | شناسه‌ی کلاینت                                                          |
| BAZINESS_CLIENT_SECRET           | ....                                   | کلید محرمانه<br>(محرمانه)                                               |
| BAZINESS_PLATFORM_PACKAGE_NAME   | com.karizmastudio.example              | packageName در<br>لینک صفحه‌ها                                          |
| BAZINESS_GAME_NAME               | example                                | game_name در<br>صفحه‌ی جایزه و<br>پیشنهاد                               |
| BAZINESS_PAY_ENABLED             | true                                   | دکمه‌های خرید را<br>روشن می‌کند                                         |
| BAZINESS_PAY_BASE_URL            | https://pay-<br>dev.karizmastudios.org | سرور پرداخت این<br>محیط                                                 |
| BAZINESS_PAY_PACKAGE_NAME        | com.karizmastudio.example              | pname در همه‌ی<br>درخواست‌های<br>پرداخت                                 |
| BAZINESS_SKU_PREFIX              | (خالی)                                 | اختیاری: SKU =<br>پیشوند + شناسه‌ی<br>محصول                             |

پلتفرم فقط وقتی روشن حساب می‌شود که BAZINESS\_PLATFORM\_ENABLED=true باشد و هم شناسه و هم کلید محرمانه تنظیم شده باشند. به این ترتیب سروری که نیمه‌کاره تنظیم شده، دقیقاً مثل وقتی رفتار می‌کند که بازیابی اصلاً وجود ندارد.

# API پلتفرم، قدم به قدم

همه‌ی endpointهای پلتفرم به ترتیبی که استفاده‌شان می‌کنید: چه کاری می‌کند، کی صدایش بزنید، چه بفرستید و چه برمی‌گردد. بدون کد؛ کد آماده در بخش «نمونه کد» است.

## پیش از شروع

### مهم

توکن‌های کاربر در حال تغییر به توکن دسترسی ۱۵ دقیقه‌ای، توکن تمدید روی سرور شما و کد اتصال یک‌بارمصرف هستند. قدم‌های زیر در دوره‌ی مهاجرت همچنان کار می‌کنند. چه چیزی را تا کی عوض کنید: **مهاجرت توکن**.

|                    |                                                                                                                              |
|--------------------|------------------------------------------------------------------------------------------------------------------------------|
| آدرس پایه<br>{api} | توسعه: <code>https://platform-dev.karizmastudios.org/api</code><br>اصلی: <code>https://platform.karizmastudio.com/api</code> |
| قالب               | بدنه‌ی درخواست و پاسخ JSON است ( <code>Content-Type: application/json</code> ) و نام فیلدها camelCase                        |
| چه کسی صدا می‌زند  | سرور شما، به جز قدم ۶ ( <code>user/get-coin</code> ) که کلاینت بازیکن صدا می‌زند                                             |
| احراز هویت         | قدم‌های ۲ تا ۵ هدر <code>&lt;توکن کلاینت&gt; Authorization: Bearer</code> از قدم ۱ را می‌خواهند؛ قدم ۶ توکن کاربر بازیکن را  |

کل ترتیب برای یک بازیکن جدید:



## قدم ۱ – گرفتن توکن کلاینت

چه کار می‌کند: شناسه‌ی کلاینت و کلید محرمانه را با یک توکن کلاینت عوض می‌کند. همه‌ی درخواست‌های دیگر سرور (قدم‌های ۲ تا ۵) این توکن را در هدر **Authorization** می‌فرستند.

**کی صدایش بزنید:** با اولین درخواست به پلتفرم بعد از بالا آمدن سرور، و دوباره هر وقت درخواستی پاسخ **401** داد، چون توکن منقضی شده است.

### HTTP

```

POST {api}/client/auth
Content-Type: application/json

{ "clientId": 12, "clientSecret": "....." }

→ 200 OK
{ "clientId": 12, "token": "eyJhbGciOiJIUzI1NiIs..." }
  
```

## قانون‌ها

- توکن را در حافظه‌ی سرور نگه دارید و برای همه‌ی درخواست‌ها از همان استفاده کنید؛ برای هر درخواست توکن تازه نگیرید.
- اگر درخواستی پاسخ `401 Unauthorized` داد، دوباره `client/auth` را صدا بزنید و همان درخواست را یک بار دیگر بفرستید.
- توکن کلاینت به اندازه‌ی کلید محرمانه حساس است؛ هرگز آن را به کلاینت نفرستید.

## قدم ۲ – ساختن کاربر بازیکن

چه کار می‌کند: برای یکی از بازیکنان شما یک کاربر بازیکن می‌سازد و این‌ها را برمی‌گرداند:

- `userId` بازیکن: آن را روی رکورد بازیکن ذخیره کنید؛ برای همه‌ی درخواست‌های بعدی لازم است
- یک توکن کاربر برای این بازیکن: آن را به کلاینت بفرستید
- `isLinkedToAccount` : آیا این کاربر با شماره موبایل وارد شده (`true`) یا هنوز ناشناس است (`false`)

کی صدایش بزنید: اولین باری که بازیکن شما به بازیکن نیاز دارد؛ معمولاً اولین اجرای بازی وقتی پلتفرم روشن است. اگر از قبل `userId` بازیکن را دارید، به جای این قدم ۳ را صدا بزنید.

### HTTP

```
POST {api}/client/create-user
Authorization: Bearer <client token>
Content-Type: application/json

{ "deviceId": "8f0c2b7e-...", "platform": "Web", "store": "Direct", "linkedToken": null }

→ 200 OK
{ "userId": 239236, "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLT...", "isLinkedToAccount": false }
```

| فیلد                     | مقدار                                                                                                                                                                                     |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>deviceId</code>    | شناسه‌ی ثابت بازیکن شما، مثلاً شناسه‌ی دستگاهی که بازی در اولین اجرا ساخته                                                                                                                |
| <code>platform</code>    | <code>Web</code> ، <code>Android</code> ، <code>Ios</code> یا <code>Editor</code>                                                                                                         |
| <code>store</code>       | <code>Direct</code> (وب)، <code>GooglePlay</code> ، <code>AppStore</code> ، <code>CafeBazaar</code> یا <code>Myket</code>                                                                 |
| <code>linkedToken</code> | توکن کاربری که بازیکن از قبل دارد، یا <code>null</code> . اپ بازیکن بازی‌ها را با <code>?linkedToken=...</code> باز می‌کند؛ فرستادن آن در این‌جا بازیکن را به همان حساب بازیکن وصل می‌کند |

### مهم

اگر `userId` برگشتی از قبل روی بازیکن دیگری از بازیکنان شما ذخیره شده، این همان شخص است، مثلاً روی گوشی جدید. آن را دوباره ذخیره نکنید؛ به جایش این دستگاه را به آن بازیکن جابه‌جا کنید (ورود ← جابه‌جایی حساب). روی ستونی که `userId` را نگه می‌دارد محدودیت یکتا بگذارید.



## HTTP

POST {api}/client/add-coin

Authorization: Bearer <client token>

Content-Type: application/json

```
{
  "userId": 239236,
  "amount": 50,
  "transactionId": "239236_50_9f2c1a7d0b4e8c3f5a6d2e1b",
  "metaData": { "source": "match_win", "matchId": 17 }
}
```

→ 200 OK

{ "resourceId": 1, "amount": 2450 } ← the player's new balance

→ 400 Bad Request (this transactionId was already credited)

{ "success": false, "error": "add resource failed" }

| فیلد          | مقدار                                                                                          |
|---------------|------------------------------------------------------------------------------------------------|
| userId        | شناسه‌ی کاربر بازینس از قدم ۲                                                                  |
| amount        | تعداد سکه (مثبت)                                                                               |
| transactionId | یک شناسه‌ی یکتا برای همین پرداخت (پایین را ببینید). پلتفرم هر شناسه را فقط یک بار واریز می‌کند |
| metaData      | هر JSON که می‌خواهید کنار تراکنش ذخیره شود (منبع، شناسه‌ی مسابقه و ...)، برای گزارش‌ها         |

امن کردن transactionId : آن را از ماهیت پرداخت بسازید، نه از زمان:

## Text

```
transactionId = <userId>_<amount>_<first 24 hex chars of sha1(grantKey)>
grantKey      = "match:17" | "reward_claim:42" | "payment:88:GITA_COIN" | "tutorial_gift:
<userId>"
```

- یک پرداخت همیشه یک شناسه‌ی ثابت می‌سازد. تلاش دوباره، دو بار زدن دکمه یا درخواست تکراری با 400 رد می‌شود و هیچ‌وقت دو بار پرداخت نمی‌شود.
- این 400 را به معنی «قبلاً پرداخت شده» بگیرید و حسابداری خودتان را تمام کنید، مثلاً جایزه را دریافت شده علامت بزنید. وگرنه کرشی بین واریز و ذخیره‌ی شما بازیکن را پشت یک خطا گیر می‌اندازد.
- شناسه‌ای که از ساعت ساخته شود ( <userId>\_<amount>\_<time> ) با هر بار صدا زدن دوباره واریز می‌کند؛ فقط برای پرداخت‌هایی که واقعاً تکرارشونده‌اند.

## قدم ۶ – خواندن موجودی (کلاینت)

چه کار می‌کند: موجودی گیتا کوین بازیکن را برمی‌گرداند. تنها درخواست پلتفرم است که کلاینت با توکن کاربر بازیکن می‌زند. درخواست از دامنه‌های دیگر مجاز است، پس بازی مرورگری می‌تواند مستقیم صدایش بزند.

کی صدایش بزنید: بعد از اینکه کلاینت توکن گرفت، وقتی یک صفحه‌ی بازیکن بسته می‌شود (شاید بازیکن سکه خرج کرده باشد)، وقتی بازی دوباره به جلو می‌آید، و بعد از اینکه سرور سکه داد.

HTTP

```
GET {api}/user/get-coin
Authorization: Bearer <user token>

→ 200 OK
{ "resourceId": 1, "amount": 2450 }
```

## خطاها در یک نگاه

| پاسخ                         | کجا            | معنی                              | کار شما                                                         |
|------------------------------|----------------|-----------------------------------|-----------------------------------------------------------------|
| 401 Unauthorized             | قدم‌های ۲ تا ۵ | توکن کلاینت نیست یا منقضی شده     | قدم ۱ را دوباره صدا بزنید و یک بار دیگر بفرستید                 |
| 200 با "isValid": false      | قدم ۴          | توکن واقعی نیست یا منقضی شده      | درخواست را رد کنید؛ کلاینت باید دوباره وارد شود                 |
| 400 با "add resource failed" | قدم ۵          | این transactionId قبلاً واریز شده | آن را پرداخت‌شده حساب کنید                                      |
| خطای شبکه / 5xx              | هر جا          | پلتفرم در دسترس نیست              | بازی را بدون بخش‌های بازیکن ادامه دهید و بعداً دوباره تلاش کنید |

## صفحه‌های بازی‌نس و پیام‌هایشان

صفحه‌های وبی که بازی شما داخل خودش نشان می‌دهد (ورود، پروفایل، فروشگاه جایزه و صفحه‌ی پیشنهاد بازی)، پارامترهایی که هر کدام لازم دارد و پیام‌هایی که برمی‌گردانند.

### این صفحه‌ها چه هستند

بعضی بخش‌های بازی‌نس **صفحه‌ی وب** هستند که بازی شما **داخل خودش** نشان می‌دهد؛ مثلاً صفحه‌ی ورود و فروشگاه جایزه‌ی «انتخاب جایزه». آدرس صفحه را در یک وب‌ویوی داخل بازی باز می‌کنید: روی وب یک `iframe`، در موتور بازی یک کامپوننت `WebView`. صفحه با **پیام** با بازی حرف می‌زند، مثلاً «بازیکن وارد شد» یا «مرا ببند».

همه‌ی صفحه‌ها روی یک سایت هستند:

| محیط  | آدرس پایه‌ی صفحه‌ها {web}                                |
|-------|----------------------------------------------------------|
| توسعه | <code>https://platform-web-dev.karizmastudios.org</code> |
| اصلی  | <code>https://baziness.com</code>                        |

ساختن نمای داخل بازی (شیتی که از پایین بالا می‌آید و با کشیدن به پایین بسته می‌شود) در **شیت پایین** / **وب‌ویو آمده** است.

### پارامترها

#### مهم

`linkCode` (کد یک‌بارمصرف از سرور شما با `v2/client/link-code/{userId}`) جای `linkedToken` را می‌گیرد. در دوره‌ی مهاجرت صفحه‌ها هر دو را قبول می‌کنند. **مهاجرت توکن** را ببینید.

| پارامتر     | کی                          | مقدار                         | چرا                                                                                                            |
|-------------|-----------------------------|-------------------------------|----------------------------------------------------------------------------------------------------------------|
| packageName | همیشه                       | نام بسته‌ی پلتفرم شما         | بازی شما را معرفی می‌کند                                                                                       |
| sdkType     | همیشه                       | js                            | به صفحه می‌گوید داخل بازی است تا برایتان پیام بفرستد. بدون آن حتی دکمه‌ی بستن خود صفحه هم نمای شما را نمی‌بندد |
| linkedToken | هر وقت دارید                | توکن کاربر بازیکن             | صفحه به نام همین بازیکن کار می‌کند و ورود داخل صفحه به او وصل می‌شود                                           |
| redirect    | پروفایل، جایزه              | / profile<br>shop             | صفحه بعد از ورود داخلش به کجا برگردد                                                                           |
| allowGuest  | ورود، پروفایل، پیشنهاد بازی | false                         | «ادامه به عنوان مهمان» خود صفحه را پنهان می‌کند؛ بازی شما این گزینه را دارد                                    |
| game_name   | جایزه، پیشنهاد بازی         | نام بازی شما، -URL encode شده | جایزه‌ها و پیشنهاد بازی شما را انتخاب می‌کند                                                                   |
| buildStore  | فقط بیلدهای فروشگاه‌های     | / CafeBazaar<br>Myket         | صفحه لینک فروشگاه‌های دیگر را پنهان می‌کند                                                                     |

همه‌ی مقدارها را encode کنید، چون game\_name ممکن است فارسی باشد و توکن‌ها ممکن است + یا / داشته باشند.

| صفحه                         | مسیر        | کی بازش کنید                                                               | پارامترها                                                                                                                                                       |
|------------------------------|-------------|----------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ورود                         | /           | بازیکن «ورود با حساب بازیکن» را می‌زند، یا برای دریافت جایزه باید وارد شود | <code>packageName</code> ، <code>sdkType=js</code><br><code>allowGuest=false</code> ، <code>linkedToken</code>                                                  |
| پروفایل                      | /profile    | تنظیمات ← اتصال یا مدیریت حساب                                             | <code>packageName</code> ، <code>sdkType=js</code><br><code>allowGuest=false</code> ، <code>linkedToken</code><br><code>redirect=profile</code>                 |
| فروشگاه جایزه «انتخاب جایزه» | /shop       | بازیکن دکمه‌ی جایزه را می‌زند                                              | <code>packageName</code> ، <code>game_name</code><br><code>allowGuest</code> ، <code>linkedToken</code> ، <code>redirect=shop</code><br><code>sdkType=js</code> |
| پیشنهاد بازی                 | /game-offer | یک بار، وسط آموزش (جزئیات)                                                 | <code>packageName</code> ، <code>game_name</code><br><code>allowGuest=false</code> ، <code>linkedToken</code> ، <code>sdkType=js</code>                         |

## Text

```

Login      https://baziness.com/?
packageName=com.karizmastudio.example&sdkType=js&linkedToken=eyJ...&allowGuest=false
Profile    https://baziness.com/profile?
packageName=com.karizmastudio.example&sdkType=js&linkedToken=eyJ...
&redirect=profile&allowGuest=false
Rewards    https://baziness.com/shop?
game_name=example&packageName=com.karizmastudio.example&sdkType=js&linkedToken=eyJ...
&redirect=shop
Game offer https://baziness.com/game-offer?
game_name=example&packageName=com.karizmastudio.example&sdkType=js&linkedToken=eyJ...
&allowGuest=false

```

## مهم

برای فروشگاه جایزه `/shop` را باز کنید، نه ریشه‌ی سایت (`/`). ریشه بازیکنی را که داخل بازی است به صفحه‌ی ورود می‌برد؛ `/shop` از توکنی که بازیکن دارد استفاده می‌کند.

## پیام‌های صفحه‌ها

صفحه با `postMessage` پیام می‌فرستد. هر پیام JSON است، به شکل رشته یا شیء، با قالب `{ "type": "...", "data": ... }`

| کار بازی شما                                                                                                                                             | معنی                                              | data                                         | type   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|----------------------------------------------|--------|
| نما را ببندید                                                                                                                                            | دکمه‌ی بستن صفحه زده شد                           | —                                            | close  |
| توکن را به <b>سرور خودتان</b> بفرستید تا <b>بررسی‌اش کند</b> و حساب را وصل کند یا به حساب دیگر بازیکن جابه‌جا شود؛ بعد توکن را تازه کنید و نما را ببندید | بازیکن وارد یک حساب شد                            | توکن: "<token>"<br>یا { "token": "<token>" } | login  |
| بازیکن را متصل علامت بزنیید و توکن را تازه کنید                                                                                                          | حساب فعلی بازیکن در همان جا وصل شد                | (بدون داده)                                  | login  |
| <b>نادیده بگیرید؛</b> ورود نیست                                                                                                                          | دست‌دادن بی‌صدایی که صفحه هنگام بارگذاری می‌فرستد | توکن                                         | token  |
| نما را ببندید، هر چه بازی به‌صورت محلی ذخیره کرده پاک کنید و مثل یک بازدیدکننده‌ی کاملاً جدید از اول شروع کنید                                           | بازیکن داخل صفحه خارج شد                          | —                                            | logout |

#### هشدار

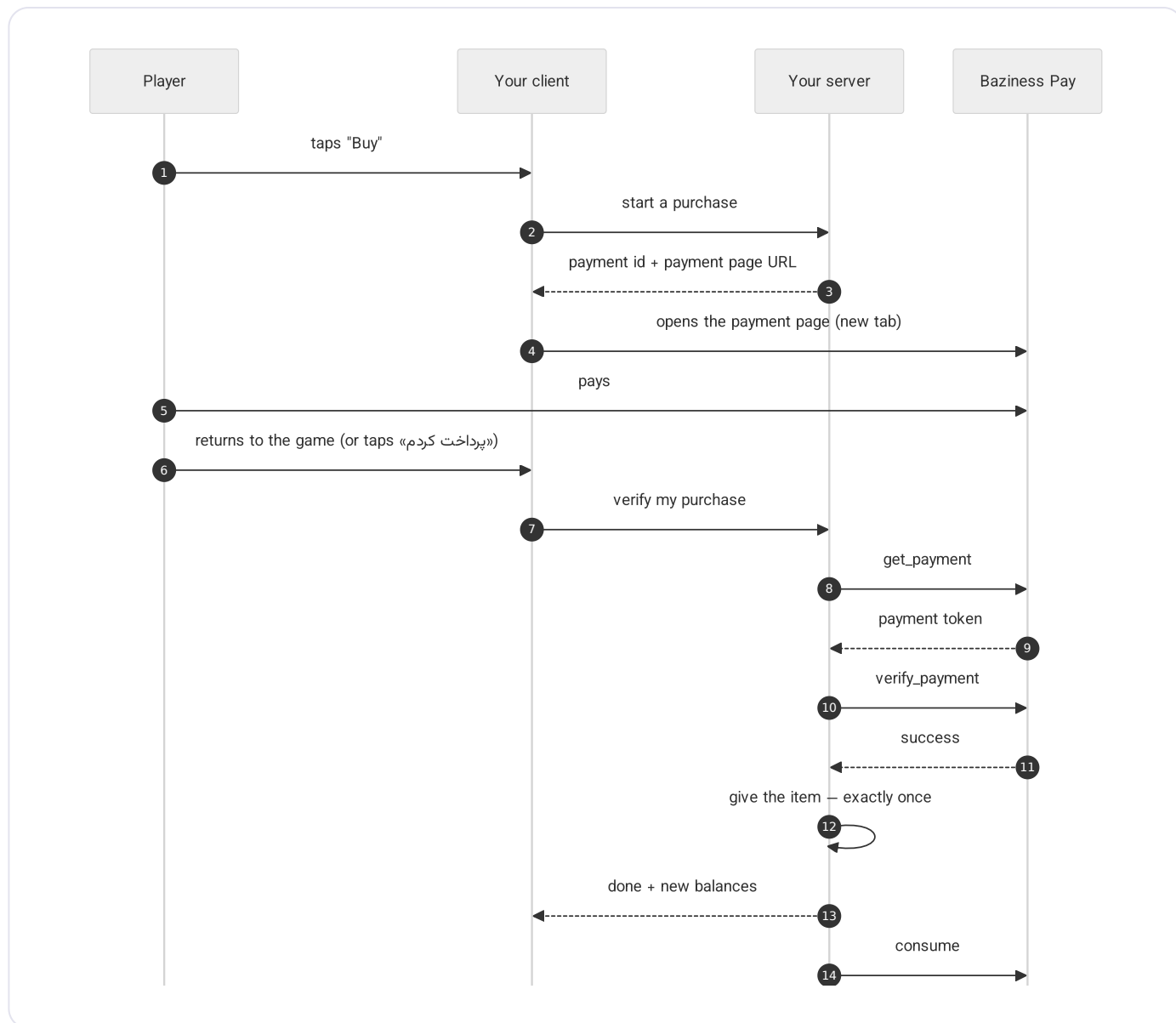
- فقط پیام‌هایی را بپذیرید که origin آن‌ها صفحه‌ای است که باز کرده‌اید ( {web} ) و بقیه را نادیده بگیرید.
- هیچ‌وقت token را ورود حساب نکنید؛ وگرنه فروشگاه جایزه به محض باز شدن بسته می‌شود.

## پرداخت بازی‌نس، قدم به قدم

فروش آیتم با پول واقعی، به ترتیب: صفحه‌ی پرداخت را باز کنید، پرداخت را پیدا و تأیید کنید، آیتم را دقیقاً یک بار بدهید و خرید را مصرف کنید تا دوباره قابل خرید باشد.

### پیش از شروع

|                    |                                                                                                                                                                  |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| آدرس پایه<br>{pay} | توسعه: <code>https://pay-dev.karizmastudios.org</code><br>اصلی: <code>https://pay.baziness.com</code>                                                            |
| چه کسی صدا می‌زند  | سرور شما. کلاینت فقط صفحه‌ی پرداخت را باز می‌کند و بقیه‌ی کارها را با سرور خود شما انجام می‌دهد                                                                  |
| pname              | نام بسته‌ی پرداخت شما؛ ممکن است با نام بسته‌ی پلتفرم فرق کند                                                                                                     |
| sku                | شناسه‌ی محصولی که در داشبورد بازی‌نس ثبت شده (مثلاً <code>example_pack</code> )                                                                                  |
| uid                | شناسه‌ی ثابت بازیکن در بازی شما، و باید عددی باشد (به شکل رشته بفرستید، مثلاً <code>"1042"</code> ).<br>uid غیرعددی با <code>no_required_params</code> رد می‌شود |



## قدم ۱ – باز کردن صفحه‌ی پرداخت

**چه کار می‌کند:** صفحه‌ی پرداخت بازی‌نس را برای یک محصول به بازیکن نشان می‌دهد.

**چطور:** وقتی بازیکن «خرید» را می‌زند، کلاینت از **سرور شما** می‌خواهد خرید را شروع کند. سرور یک سفارش ذخیره می‌کند (قدم ۴ را ببینید)، آدرس صفحه را می‌سازد و برمی‌گرداند. کلاینت آن را در **تب جدید** (وب) یا مرورگر سیستم (موبایل) باز می‌کند.

### HTTP

```
GET {pay}/buy?sku=example_pack&uid=1042&pname=com.karizmastudio.example
```

→ the hosted payment page (opened in a new tab / the system browser)

مرورگرها تبی را که **مستقیم** با ضربه‌ی بازیکن باز نشده باشد مسدود می‌کنند. اول در همان رویداد ضربه یک تب خالی باز کنید، بعد وقتی سرور جواب داد آن را به آدرس ببرید (**کد کلاینت**).

## قدم ۲ – بررسی اینکه بازیکن پرداخت کرده یا نه

چه کار می‌کند: از بازیکن می‌پرسد این بازیکن برای این محصول پرداخت کامل شده و هنوز مصرف‌نشده‌ای دارد یا نه، و اگر دارد توکن پرداخت را برمی‌گرداند.

کی صدایش بزنید: وقتی بازیکن به بازی برمی‌گردد (پنجره یا اپ دوباره فوکوس می‌گیرد)، وقتی «پرداخت کردم» را می‌زند، و وقتی فروشگاه باز می‌شود، برای پرداخت‌هایی که وقتی بازی بسته بود تمام شده‌اند.

### HTTP

```
POST {pay}/get_payment
Content-Type: application/json

{ "uid": "1042", "sku": "example_pack", "pname": "com.karizmastudio.example" }
```

→ 200 OK (paid, not consumed yet)

```
{ "uid": "1042", "sku": "example_pack", "token": "TEST-633...", "time": 1791366000000,
  "success": true }
```

→ 200 OK (nothing to verify yet – the player hasn't paid)

```
{ "error": "payment_is_not_valid", "success": false }
```

یعنی «هنوز پرداختی نیست»: بی‌صدا منتظر بمانید. `payment_is_not_valid`

## قدم ۳ – تأیید پرداخت

چه کار می‌کند: از بازیکن تأیید می‌گیرد که توکن پرداخت واقعی است. این همان درخواستی است که از شما محافظت می‌کند: هیچ‌وقت بر اساس حرف کلاینت چیزی ندهید.

### HTTP

```
POST {pay}/verify_payment
Content-Type: application/x-www-form-urlencoded

uid=1042&token=TEST-633...&sku=example_pack&pname=com.karizmastudio.example
```

→ 200 OK

```
{ "success": true }
```

`verify_payment` بدنه را به شکل فرم (`application/x-www-form-urlencoded`) می‌گیرد؛ بقیه‌ی درخواست‌ها JSON. فرستادن JSON در این‌جا رایج‌ترین دلیل شکست تأیید است.

## قدم ۴ – دادن آیتم، دقیقاً یک بار

بازینس به شما می‌گوید پرداخت واقعی است؛ اینکه بازیکن آیتم را یک بار بگیرد کار سرور شماست. یک جدول پرداخت‌ها در دیتابیس نگه دارید:

| ستون                                                                                   | معنی                                                                                                                                    |
|----------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <code>id</code>                                                                        | شناسه‌ی سفارش شما (در قدم ۱ به کلاینت برمی‌گردد)                                                                                        |
| <code>user_id</code> ، <code>product_id</code> ، <code>sku</code> ، <code>price</code> | چه کسی چه چیزی می‌خرد                                                                                                                   |
| <code>status</code>                                                                    | <code>REDIRECTED</code> (صفحه‌ی پرداخت باز شد) ← <code>FULFILLED</code> (آیتم داده شد)، یا <code>FAILED</code> / <code>CANCELLED</code> |
| <code>reward_snapshot</code>                                                           | دقیقاً همان چیزی که هنگام زدن «خرید» به بازیکن وعده داده شد                                                                             |
| <code>provider_token</code>                                                            | توکن پرداخت از قدم ۲، با محدودیت یکتا                                                                                                   |

بعد از قدم ۳ موفق، در یک تراکنش دیتابیس:

۱. ردیف سفارش را قفل کنید و ببینید از قبل `FULFILLED` نباشد. اگر هست، بدون دادن دوباره‌ی آیتم «انجام شد» را برگردانید.
۲. اگر همین توکن پرداخت قبلاً سفارش دیگری را تحویل داده، این یکی را `FAILED` کنید و متوقف شوید.
۳. آیتم‌های `reward_snapshot` را بدهید، نه تعریف فعلی محصول را. مدیری که وسط پرداخت بازیکن یک بسته را ویرایش می‌کند نباید سفارش او را عوض کند.
۴. توکن را ذخیره کنید و سفارش را `FULFILLED` کنید.

## قدم ۵ – مصرف کردن خرید

**چه کار می‌کند:** خرید را مصرف‌شده علامت می‌زند تا همان محصول دوباره قابل خرید باشد. تا وقتی مصرف نشده، قدم ۲ همچنان آن را برمی‌گرداند و خرید دوم آن SKU مسدود است.

**کی:** درست بعد از ثبت نهایی قدم ۴. اگر شکست خورد چند بار دوباره تلاش کنید؛ آیتم قبلاً داده شده، پس شکست این قدم هیچ‌وقت بازیکن را معطل نمی‌کند.

HTTP

```
POST {pay}/consume
Content-Type: application/json

{ "uid": "1042", "sku": "example_pack", "pname": "com.karizmastudio.example" }
```

→ 200 OK

```
{ "success": true }
```

## چیزی که بازیکن می بیند

| لحظه                                  | کلاينت شما نشان می دهد                                                                                                  |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| بعد از زدن «خرید»                     | یک پنجره‌ی کوچک: «صفحه‌ی پرداخت بازی‌س باز شد، بعد از پرداخت به بازی برگرد»، با دکمه‌ی «پرداخت کردم»                    |
| برگشت به بازی، هنوز پرداختی پیدا نشده | چیز تازه‌ای نشان نمی دهد (بررسی بی صدا)؛ فقط زدن «پرداخت کردم» می گوید «پرداختی پیدا نشد، چند لحظه بعد دوباره بررسی کن» |
| پرداخت تأیید شد                       | «خرید موفق» و موجودی‌های به روز                                                                                         |
| پنجره پیش از پرداخت بسته شد           | چیزی لغو نمی شود؛ دفعه‌ی بعد که فروشگاه باز شود دوباره بررسی می شود                                                     |

## تست

- روی **pay-dev** صفحه‌ی پرداخت گزینه‌ی «خرید آزمایشی» دارد: کل جریان، از قدم ۱ تا ۵، بدون پول واقعی اجرا می شود.
- اگر به هیچ درگاهی دسترسی ندارید، از **درگاه آزمایشی محلی** بازی نمونه استفاده کنید: همان پروتکل را دارد، با دکمه‌ی «پرداخت موفق» ([تنظیمات](#)).

## بیلدهای فروشگاه‌ی (بازار / مایکت)

بیلدی که در کافه بازار یا مایکت منتشر می شود باید از **پرداخت درون برنامه‌ای خود آن فروشگاه** استفاده کند، نه پرداخت بازی‌س. نام SKUها را در هر دو یکی نگه دارید تا یک فهرست محصول برای همه‌ی بیلدها کافی باشد، و در آن بیلدها **buildStore** را روی لینک صفحه‌های بازی‌س بفرستید ([پارامترها](#)).

## ورود و اتصال حساب بازیکن

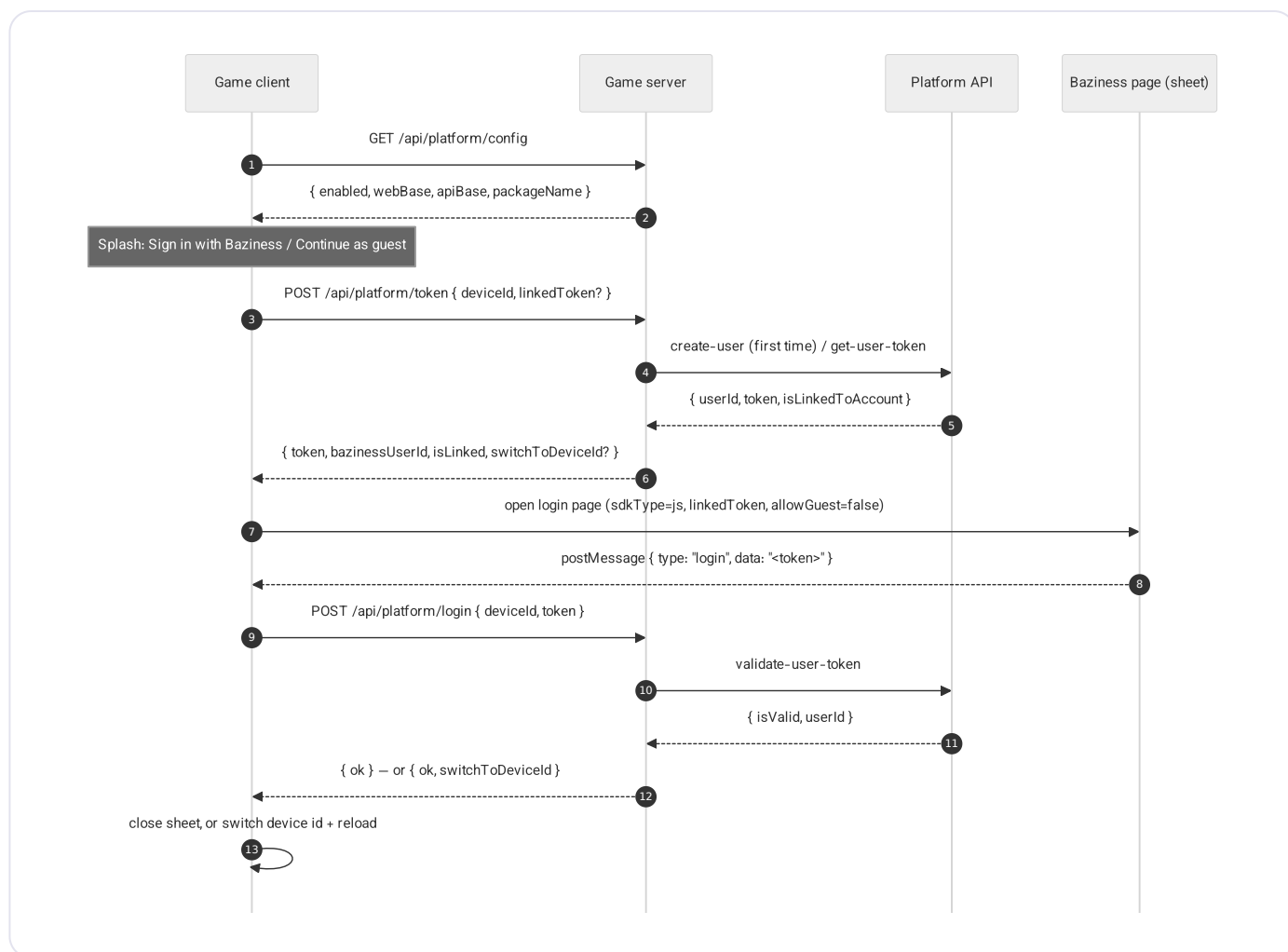
بازیکن چطور هویت بازیکن می‌گیرد، از اسپلش یا تنظیمات وارد می‌شود، و بازی چطور حساب‌ها را بی‌خطر جابه‌جا می‌کند.

### مدل هویت

بازی شما حساب‌های بازیکن خود را دارد، مثلاً با کلید شناسه‌ی دستگاهی که در اولین اجرا ساخته می‌شود. بازیکن روی همین حساب سوار می‌شود:

- اولین باری که پلتفرم روشن است، سرور شما یک کاربر ناشناس بازیکن می‌سازد ( `create-user` ) و `userId` آن را روی رکورد بازیکن ذخیره می‌کند.
- بازیکن تا وقتی با شماره موبایلش در یکی از صفحه‌های بازیکن وارد نشود ناشناس می‌ماند ( `isLinkedToAccount: false` ). بعد از ورود حساب متصل است.
- اگر حساب بازیکنی که بازیکن واردش می‌شود از قبل مال بازیکن دیگری از بازیکنان شما باشد (همان شخص روی گوشی دیگر)، بازی به آن بازیکن جابه‌جا می‌شود و حساب را دو بار وصل نمی‌کند.

| وضعیت        | <code>userId</code> بازیکن ذخیره شده؟ | <code>isLinkedToAccount</code> | چیزی که بازیکن می‌بیند                                         |
|--------------|---------------------------------------|--------------------------------|----------------------------------------------------------------|
| بازیکن خاموش | —                                     | —                              | هیچ بخش بازیکنی                                                |
| مهمان        | بله (کاربر ناشناس)                    | <code>false</code>             | موجودی سکه و دکمه‌ی جایزه کار می‌کنند؛ «اتصال حساب» در تنظیمات |
| وارد شده     | بله                                   | <code>true</code>              | همه‌چیز، از جمله دریافت جایزه‌های گیتا کوین                    |



## چیزهایی که سرور شما فراهم می‌کند

کلاينت شما هيچ وقت خودش API سرور پلتفرم را صدا نمی‌زند. سه endpoint از سرور خودتان را صدا می‌زند:

| endpoint شما                             | کلاينت می‌فرستد                                       | سرور شما                                                                                                                                                                                         | پاسخ                                                                                                      |
|------------------------------------------|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| GET<br><code>/api/platform/config</code> | —                                                     | کاری نمی‌کند، فقط تنظیمات را می‌گوید                                                                                                                                                             | <code>{ enabled, webBase, apiBase, packageName, gameName, { gameOfferUrl } یا { enabled: false } }</code> |
| POST<br><code>/api/platform/token</code> | شناسه‌ی دستگاه،<br><code>linkedToken</code> و اگر هست | اگر هنوز کاربر بازی‌س <code>create-user</code> ندارد ← <code>user</code> ؛ وگرنه ← <code>get-user-token</code> اگر <code>token</code> حاصل مال <code>userId</code> بازیکن دیگری باشد وصل نمی‌کند | <code>{ token, userId, isLinkedToAccount } یا { switchToDeviceId }</code>                                 |
| POST<br><code>/api/platform/login</code> | شناسه‌ی دستگاه، توکنی که صفحه‌ی ورود داده             | <code>validate-user-token</code> ← این حساب مال کیست؟ مال بازیکن دیگری ← جابه‌جایی؛ وگرنه وصل به همین بازیکن                                                                                     | <code>{ ok: true } یا { ok: false } یا { ok: true, switchToDeviceId } یا { ok: false }</code>             |

اگر پلتفرم در دسترس نباشد، `/token` پاسخ `{ enabled: true, error: "platform_unavailable" }` را بدون توکن می‌دهد و بازی به هر حال بالا می‌آید.

## اجرای اول: مهمان یا ورود

وقتی بازی‌س روشن است و بازیکن نه هنوز انتخابی کرده و نه وارد شده، اسپلش به جای نوار بارگذاری دو دکمه نشان می‌دهد:

- «ورود با حساب بازی‌س»: صفحه‌ی ورود را باز می‌کند و منتظر می‌ماند.
- «ادامه به عنوان مهمان»: انتخاب را به خاطر می‌سپارد و ادامه می‌دهد.

ترتیب بالا آمدن بازی:

1. بازیکن را در سیستم حساب **خود بازی** وارد کنید.
2. از سرور `/api/platform/config` را بپرسید. اگر بازی‌س خاموش بود، از همه‌ی قدم‌های بعد بگذرید.
3. اگر انتخابی نشده و بازیکن وارد نیست، دو دکمه را نشان دهید و منتظر بمانید.
4. `/api/platform/token` را صدا بزنید. اگر `switchToDeviceId` برگشت، آن شناسه را بپذیرید و از اول شروع کنید.

5. اگر بازیکن همین الان وارد شد، پروفایلش را دوباره بخوانید تا صفحه حساب وارد شده را نشان دهد.

اگر بازیکن صفحه‌ی ورود را بدون وارد شدن بست، دو دکمه را دوباره نشان دهید.

## جابه‌جایی حساب، قدم به قدم

جابه‌جایی وقتی پیش می‌آید که حساب بازیکنی که بازیکن واردش می‌شود از قبل به بازیکن دیگری از بازیکنان شما وصل باشد، مثلاً بعد از نصب دوباره یا روی گوشی جدید:

1. سرور پاسخ می‌دهد "`<شناسه‌ی دستگاه آن بازیکن>`": `switchToDeviceId`.
2. کلاینت آن شناسه را به‌عنوان شناسه‌ی خودش ذخیره می‌کند و توکن بازیکن ذخیره‌شده را فراموش می‌کند.
3. بازی را از اول اجرا می‌کند، بدون `linkedToken` در آدرس.
4. اجرای بعدی بازیکن دیگر را با همه‌ی پیشرفت‌ش بالا می‌آورد.

## باز شدن بازی از اپ بازیکن

### مهم

اپ بازیکن حالا `?linkCode=` (یک بار مصرف) را هم می‌فرستد. به جای `linkedToken` آن را به سرورتان بدهید. **مهاجرت توکن را ببینید.**

اپ بازیکن بازی‌ها را با `<توکن>?linkedToken=` در آدرس باز می‌کند:

- آن توکن را به‌عنوان `linkedToken` به `/api/platform/token` بفرستید تا بازیکن در حساب درست بازیکن قرار بگیرد. در اجرای بعدی آن را از آدرس پاک کنید.
- اگر دستگاه از قبل توکن دیگری دارد، حساب بازیکن دیگری دارد می‌آید: پیش از ادامه داده‌های محلی را پاک کنید و از اول شروع کنید.
- ورود تعیین می‌کند کدام حساب بازی کند، هیچ‌وقت تعیین نمی‌کند بازیکن جدید است یا نه. بازیکن تازه‌واردی که این‌طور می‌آید هم باید آموزش و هدیه‌های خوش‌آمد شما را ببیند.

## ورود در آینده، از تنظیمات

مهمان هر وقت بخواهد می‌تواند حسابش را وصل کند: تنظیمات ← باز کردن صفحه‌ی پروفایل (پارامترها). وقتی نما بسته شد، دوباره `/api/platform/token` را بپرسید و موجودی سکه را تازه کنید.

## خروج

وقتی صفحه `logout` می‌فرستد، نما را ببندید، هر چه بازی به‌صورت محلی ذخیره کرده پاک کنید (هویت، توکن، انتخاب‌ها، پیشرفت آموزش) و مثل یک بازدیدکننده‌ی کاملاً جدید از اول شروع کنید.

## تست ورود روی پلتفرم توسعه

### پیشنهاد

روی پلتفرم توسعه (همان جایی که بازی‌های تستی به آن وصل‌اند) با هر شماره موبایلی و کد 111111 می‌توانید وارد شوید. هیچ پیامکی فرستاده نمی‌شود. با شماره‌های ساختگی هر چند حساب تستی که لازم دارید بسازید: یکی برای تست جابه‌جایی حساب، یکی برای یک مهمان تازه و غیره. محیط اصلی کد واقعی را پیامک می‌کند.

## اشتباه‌های رایج

| اشتباه                                              | نتیجه                                                                                 | راه‌حل                                                                                                          |
|-----------------------------------------------------|---------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| پیام <code>token</code> صفحه ورود حساب شود          | فروشگاه جایزه به محض باز شدن بسته می‌شود؛ ورود اول پیش از وارد شدن بازیکن تمام می‌شود | فقط <code>login</code> ورود است                                                                                 |
| وصل کردن توکن بدون <code>validate-user-token</code> | هر کسی می‌تواند هر حسابی را مال خودش کند                                              | همیشه در سرور بررسی کنید                                                                                        |
| باز کردن صفحه‌ی ورود بدون توکن بازیکن               | ورود اول از اسپلش کاربر ناشناسی ندارد که به آن وصل شود                                | مطمئن شوید <code>/api/platform/token</code> قبلاً اجرا شده و توکنش را به‌عنوان <code>linkedToken</code> بفرستید |
| نبودن <code>allowGuest=false</code> روی صفحه‌ی ورود | بازیکن یک «ادامه به عنوان مهمان» دوم می‌بیند                                          | روی صفحه‌های ورود، پروفایل و پیشنهاد اضافه‌اش کنید                                                              |
| پاداش اتصال حساب بیش از یک بار                      | خروج و ورود دوباره، پاداش را تکرار می‌کند                                             | هدیه‌ی «حساب وصل شد» را از سرور و یک بار برای هر بازیکن بدهید                                                   |

کد آماده: `endpoint`‌های سرور، کنترلر و پیام‌های کلاینت.

# گیتا کوین

بازیکن چطور گیتا کوین به دست می‌آورد و خرج می‌کند، بازی چطور موجودی را نشان می‌دهد، سکه را امن می‌دهد و با بازیکنانی که هنوز وارد نشده‌اند چه می‌کند.

## گیتا کوین چطور کار می‌کند

گیتا کوین واحد پول بازیکن است. بازیکن آن را در بازی شما به دست می‌آورد (برد مسابقه، رتبه‌ی تورنمنت، هدیه، خرید بسته) و در فروشگاه جایزه برای جایزه‌های واقعی خرج می‌کند. موجودی روی پلتفرم نگه داشته می‌شود، نه در دیتابیس شما:

| کار          | چه کسی     | چطور                                             |
|--------------|------------|--------------------------------------------------|
| نمایش موجودی | کلاینت شما | <code>user/get-coin</code> با توکن کاربر بازیکن  |
| دادن سکه     | سرور شما   | <code>client/add-coin</code> با توکن کلاینت      |
| خرج کردن سکه | بازیکن     | داخل صفحه‌ی فروشگاه جایزه؛ بازی شما کاری نمی‌کند |

### هشدار

هیچ‌وقت عددی را که به‌صورت محلی ذخیره شده موجودی حساب نکنید و هیچ‌وقت اجازه ندهید کلاینت سکه اضافه کند.

## نمایش موجودی

موجودی را تازه کنید:

- بلافاصله بعد از اینکه کلاینت توکنش را گرفت
- وقتی یک صفحه‌ی بازیکن بسته می‌شود، چون شاید بازیکن در فروشگاه جایزه سکه خرج کرده باشد
- وقتی بازی دوباره به جلو می‌آید
- بعد از اینکه سرور شما سکه داد

تا وقتی موجودی معلوم نیست یا درخواست شکست خورده، ... نشان دهید یا آن را پنهان کنید. هیچ‌وقت 0 نشان ندهید؛ بازیکن خیال می‌کند سکه‌هایش از دست رفته.

## دادن سکه به‌شکل امن

هر پرداخت یک `transactionId` یکتا لازم دارد که از ماهیت همان پرداخت ساخته شده، مثلاً `match:17` یا `reward_claim:42` (فرمول). در نتیجه:

- تلاش دوباره، دو بار زدن دکمه یا درخواست تکراری هیچ وقت دو بار پرداخت نمی کند: پلتفرم تکرار را با 400 رد می کند
- آن 400 را به معنی «قبلاً پرداخت شده» بگیرید و حسابداری خودتان را تمام کنید
- هر پرداخت را در دفتر حساب خودتان هم ثبت کنید (به چه کسی، چقدر، چرا)، برای پشتیبانی و گزارش

## بازیکنانی که هنوز وارد نشده اند

مهمان هم یک کاربر ناشناس بازیکن دارد، پس می شود به او سکه داد. ولی خیلی از بازی ها مرز را وارد شدن می گذارند:

| نوع پرداخت                                                                                | بازیکن وارد نشده                                                                                                                             | چرا                                                              |
|-------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|
| <b>دریافت ها:</b> جایزه ای که بازیکن برای گرفتنش دکمه می زند (برد، جایزه ی تورنمنت، هدیه) | با خطای «اول وارد شو» رد کنید؛ کلاینت صفحه ی ورود را باز می کند و بعد از ورود موفق دوباره تلاش می کند                                        | سکه ای که با آن جایزه ی واقعی می خرد باید مال یک حساب واقعی باشد |
| <b>پرداخت هایی که نباید شکست بخورند:</b> خرید پرداخت شده ای که گیتا کوین دارد             | سکه ها را در یک موجودی امانی محلی روی سرور نگه دارید                                                                                         | سفارش پرداخت شده هیچ وقت نباید شکست بخورد                        |
| وقتی بازیکن وارد شد                                                                       | کل موجودی امانی را با یک <code>add-coin</code> (با <code>transactionId</code> مخصوص خودش) به بازیکن منتقل کنید و بعد موجودی محلی را صفر کنید | سکه ها دقیقاً یک بار منتقل می شوند                               |

وقتی بازیکن **خاموش** است هم جایزه های گیتا کوین را در همان موجودی محلی نگه دارید تا اقتصاد بازی بدون کلید هم قابل تست باشد.

## ارز را صادقانه نشان دهید

- همه جا آن را «گیتا کوین» بنامید و آیکون مخصوصش را بگذارید.
- بیلدی که بدون بازیکن منتشر می شود نباید جایزه ی گیتا کوین تبلیغ کند.

کد آماده: [دادن سکه، نمایش موجودی](#).

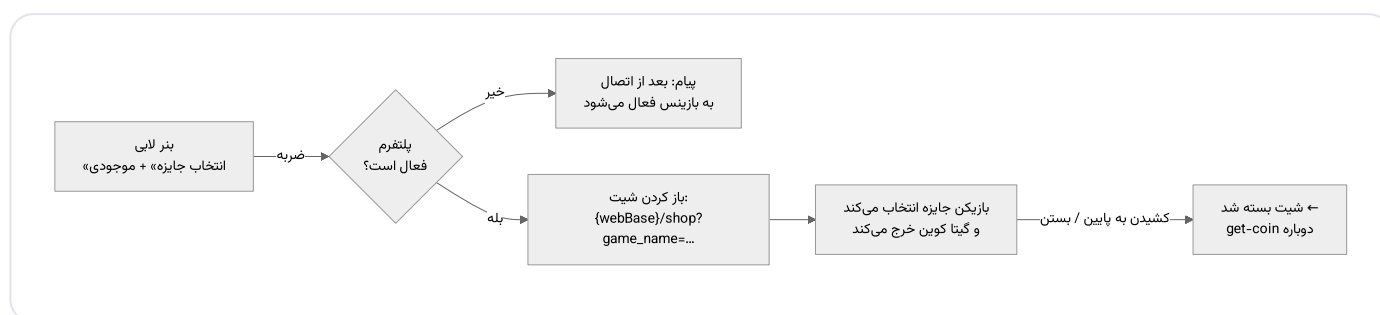
## صفحه‌ی جایزه («انتخاب جایزه»)

بگذارید بازیکن گیتا کوین‌هایش را با جایزه‌ی واقعی عوض کند: دکمه‌ای در بازی که فروشگاه جایزه‌ی بازیکن را باز می‌کند.

### این صفحه چیست

«انتخاب جایزه» فروشگاه جایزه‌ی بازیکن است: صفحه‌ای که بازیکن در آن گیتا کوین را برای جایزه‌های واقعی خرج می‌کند. بازی شما سه کار می‌کند:

1. برای بازیکن سکه به دست می‌آورد (گیتا کوین)
2. موجودی را روی یک دکمه یا بنر نشان می‌دهد
3. وقتی دکمه زده شد، صفحه‌ی جایزه را در وب‌ویوی داخل بازی باز می‌کند



### آدرس

#### Text

توکن=<linkedToken&sdkType=js>&packageName=<بسته> نام بازی=<game\_name>/{web}/shop?game\_name=<بسته> و گیتا کوین خرج می‌کند  
&redirect=shop>کاربر

پارامترها در صفحه‌های بازیکن توضیح داده شده‌اند. دو قانون این‌جا مهم‌ترند:

- `/shop` را باز کنید، نه ریشه‌ی سایت را؛ ریشه بازیکن داخل بازی را به صفحه‌ی ورود می‌برد.
- `allowGuest=false` را اضافه نکنید؛ فروشگاه مقصد است، نه صفحه‌ی ورود.

### دکمه در بازی شما

- آن را جایی بگذارید که بازیکن می‌بیند: لابی یا منوی اصلی. **موجودی زنده** را با آیکون گیتا کوین روی آن نشان دهید.
- وقتی بازیکن خاموش است، یا پنهانش کنید یا با زدنش بگویید «بعد از اتصال به بازیکن فعال می‌شود».
- وسط آموزش، ضربه‌ی اول می‌تواند به جایش صفحه‌ی پیشنهاد بازی را باز کند.

## بعد از بسته شدن صفحه

شاید بازیکن سکه خرج کرده باشد، پس به محض بسته شدن نما و دوباره وقتی بازی فوکوس می‌گیرد موجودی را تازه کنید.

## پنهان کردن نوار پایین پلتفرم (اختیاری)

سایت بازیکنس نوار ناوبری پایین خودش را دارد (فروشگاه / بازی‌ها / خانه / ماموریت‌ها / بازیکنس من). داخل بازی شاید بخواهید پنهانش کنید. صفحه مال سایت دیگری است، پس CSS و اسکریپت شما به داخلش دسترسی ندارند. به جایش:

1. وب‌ویو را به اندازه‌ی ارتفاع نوار بلندتر از ظرفش کنید.

2. بگذارید ظرف بخش اضافه را برش بدهد.

نوار پایین صفحه است، پس بیرون محدوده‌ی دیده‌شده می‌افتد و بقیه‌ی صفحه مثل قبل اسکرول می‌شود. ارتفاع را روی گوشه‌های واقعی تنظیم کنید. [کد](#).

## بیلدهای بدون بازیکنس

بیلدی که بدون بازیکنس منتشر می‌شود (مثلاً بعضی بیلدهای فروشگاه‌ها) نباید دکمه را نشان دهد و نباید جایزه‌ی گیتا کوین تبلیغ کند. بگذارید سرور تصمیم بگیرد: برای آن بیلد `{ enabled: false }` برگرداند و کلاینت هیچ‌کدام را نکشد.

## چک‌لیست

- ☐ دکمه موجودی زنده را نشان می‌دهد (هنگام بارگذاری یا خطا ...).
- ☐ `/shop` را با `game_name`، `packageName`، `sdktType=js`، `linkedToken` و `redirect=shop` باز می‌کند.
- ☐ نه `allowGuest=false` دارد و نه ریشه‌ی سایت را باز می‌کند.
- ☐ موجودی بعد از بسته شدن نما و بعد از فوکوس دوباره‌ی بازی تازه می‌شود.
- ☐ وقتی بازیکنس خاموش است، دکمه پنهان است یا توضیح می‌دهد چرا در دسترس نیست.

## صفحه‌ی پیشنهاد آموزش (game-offer)

صفحه‌ای نمایشی که مخصوص بازی شما ساخته شده و یک بار وسط آموزش باز می‌شود تا به بازیکن تازه‌وارد نشان دهد با گیتا کوین چه چیزهایی می‌شود گرفت و یک بسته‌ی شروع رایگان به او پیشنهاد کند.

### این صفحه چیست

هر بازی می‌تواند یک **صفحه‌ی پیشنهاد** روی بازیکن داشته باشد که مخصوص همان بازی است و با `game_name` انتخاب می‌شود:

Text

```
{web}/game-offer?game_name=<نام بازی>
```

این صفحه جایزه‌هایی را که با گیتا کوین می‌شود گرفت نشان می‌دهد، به‌علاوه‌ی یک بسته‌ی رایگان مخصوص بازی شما؛ مثلاً صفحه‌ی بازی نمونه «بسته ویژه بازی example – رایگان» را با دکمه‌ی «خرید رایگان» نشان می‌دهد. این صفحه نمایشی است: یک بار و وسط آموزش باز می‌کنید تا انتظار بازیکن شکل بگیرد. بعد از آموزش، دکمه‌ی جایزه‌ی شما **فروشگاه جایزه‌ی واقعی** را باز می‌کند.

#### نکته

از تیم پلتفرم بخواهید صفحه‌ی پیشنهاد بازی شما را راه بیندازد، و از همان `game_name` صفحه‌ی جایزه استفاده کنید.

### آدرس

| محیط  | آدرس                                                                                           |
|-------|------------------------------------------------------------------------------------------------|
| توسعه | <code>https://platform-web-dev.karizmastudios.org/game-offer?game_name=&lt;نام بازی&gt;</code> |
| اصلی  | <code>https://baziness.com/game-offer?game_name=&lt;نام بازی&gt;</code>                        |

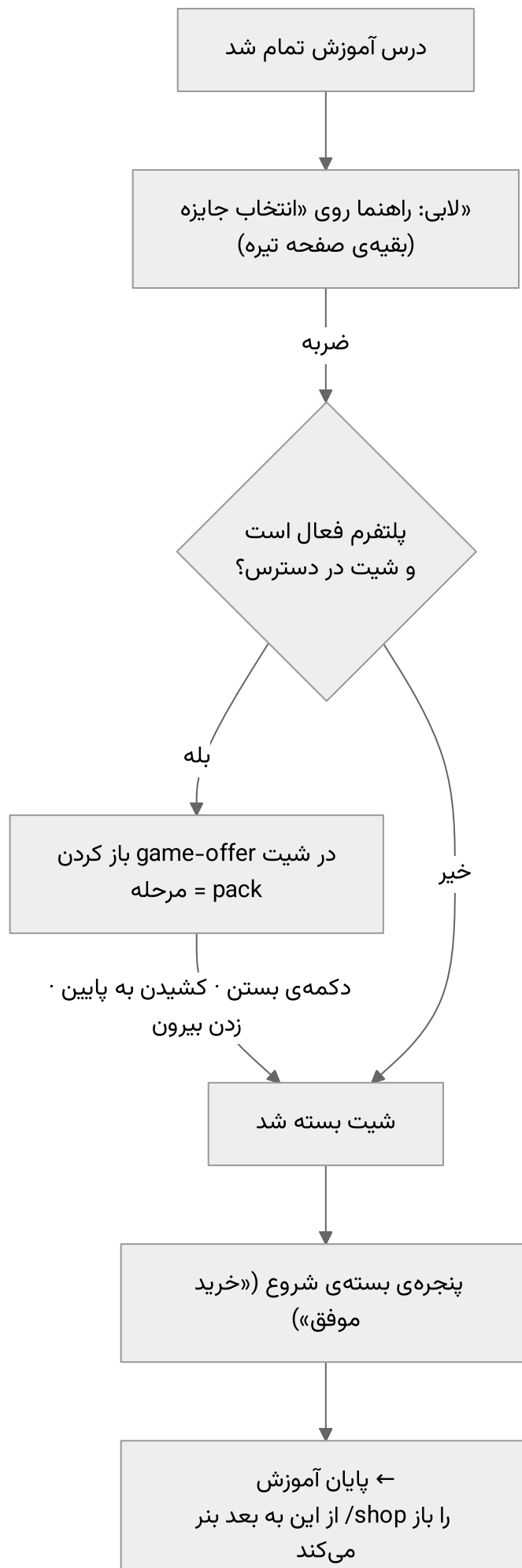
مثل همه‌ی صفحه‌های بازیکن، `packageName` و `sdkType=js` و همچنین `linkedToken` و `allowGuest=false` لازم دارد (پارامترها).

#### هشدار

بدون `sdkType=js` صفحه‌ی پیشنهاد هیچ‌وقت پیام `close` را نمی‌فرستد، پس دکمه‌اش نمی‌تواند نمای شما را ببندد.







## قانون‌ها

1. پیش از باز کردن صفحه، قدم آموزش را ذخیره کنید. اگر بازیکن وقتی صفحه باز است بازی را دوباره بارگذاری کند، با بسته‌ی شروع ادامه می‌دهد و پیشنهاد را دو بار نمی‌بیند.
2. به هر شکلی که صفحه بسته شد ادامه دهید: دکمه‌ی خودش، کشیدن به پایین یا زدن بیرون آن.
3. آموزش را هیچ‌وقت معطل بازی‌نس نکنید. اگر خاموش یا در دسترس نیست، مستقیم به بسته‌ی شروع بروید.
4. بسته‌ی شروع را از سرور و یک بار برای هر بازیکن بدهید. آن را مثل یک خرید موفق («خرید موفق») نشان دهید، هرچند رایگان است.
5. بازیکنانی که از اپ بازی‌نس می‌آیند هم آموزش را می‌بینند. ورود، بازیکن را «قدیمی» نمی‌کند.
6. بعد از آموزش [/shop](#) را باز کنید، دیگر هرگز صفحه‌ی پیشنهاد را.

## تست کنید

1. داده‌های سایت را پاک کنید (یا پنجره‌ی خصوصی) و بازی را باز کنید.
2. در اسپلش یکی از دو گزینه را انتخاب کنید و درس اول آموزش را تمام کنید.
3. فقط دکمه‌ی جایزه روشن است. آن را بزنید: صفحه‌ی پیشنهاد از پایین بالا می‌آید.
4. آن را هر بار به یک روش ببندید (دکمه‌اش، کشیدن دستگیره، زدن بیرون). هر روش باید بسته‌ی شروع را بیاورد.
5. دوباره دکمه‌ی جایزه را بزنید: حالا [/shop](#) واقعی باز می‌شود.

کد آماده: [کلاینت](#). نسخه‌ی زنده را در [بازی نمونه](#) ببینید.

# لینک‌ها و پارامترهای UTM

سه نوع لینک در یکپارچه‌سازی بازاریابی، و اینکه لینک‌هایی را که بازیکن به بازی شما می‌آورند چطور برچسب بزنید تا هر بازیکن به کمپین درستش نسبت داده شود.

## سه نوع لینک

| لینک                                                                      | جهت                           | سازنده                                        | جزئیات                |
|---------------------------------------------------------------------------|-------------------------------|-----------------------------------------------|-----------------------|
| لینک صفحه‌های بازاریابی (ورود، پروفایل،<br>( /game-offer ، /shop )        | بازی ← بازاریابی              | کلاینت شما، با تنظیماتی<br>که از سرور می‌گیرد | صفحه‌های<br>بازاریابی |
| لینک صفحه‌ی پرداخت ( /buy )                                               | بازی ←<br>پرداخت<br>بازاریابی | فقط سرور شما                                  | پرداخت<br>بازاریابی   |
| لینک‌هایی که به بازی شما می‌آیند (?)<br>( ?linkedToken= ، ?tag= ، utm_... | بیرون ← بازی                  | بازاریابی، تبلیغات، اپ<br>بازاریابی           | همین صفحه             |

## لینک‌هایی که به بازی شما می‌آیند

هر لینکی که بازیکن به بازی شما می‌آورد باید بگوید او از کجا آمده:

Text

```
https://yourgame.example.com/?
utm_source=baziness&utm_medium=inapp&utm_campaign=shahrivar_1405&utm_content=banner_home
https://yourgame.example.com/?tag=telegram
```

| پارامتر      | معنی                                               | مثال                                 |
|--------------|----------------------------------------------------|--------------------------------------|
| utm_source   | چه کسی ترافیک را می‌فرستد                          | telegram ، baziness ، instagram      |
| utm_medium   | نوع کانال                                          | social ، cpc ، banner ، inapp ، push |
| utm_campaign | نام کمپین                                          | nowruz_1405 ، shahrivar_1405         |
| utm_content  | طرح یا جایگاه تبلیغ                                | story_2 ، banner_home                |
| tag          | برچسب کوتاه کانال (جایگزین ساده‌تر)                | google ، telegram                    |
| linkedToken  | اپ بازی‌س اضافه می‌کند وقتی بازی شما را باز می‌کند | توکن کاربر بازیکن (ورود)             |

#### مهم

مقدار UTM باید ASCII ساده باشد: حروف کوچک انگلیسی، رقم، . ، \_ ، - و حداکثر ۶۴ کاراکتر. نام فارسی کمپین گم می‌شود، پس بنویسید shahrivar\_1405 ، نه شهر یور .

## بازی چطور باید آن‌ها را بخواند

- پیش از هر چیز بخوانیدشان، پیش از اینکه چیزی رندر شود. نام‌ها را بدون حساسیت به بزرگی و کوچکی حروف تطبیق دهید.
- از نوار آدرس پاکشان کنید. بازیکن آن‌ها را نمی‌بیند و ویرایش نمی‌کند، و بازیکنی که آدرسش را با دوستانش به اشتراک می‌گذارد کمپین شما را به نام آن‌ها ثبت نمی‌کند.
- بار اول روی دستگاه ذخیره‌شان کنید («اولین تماس»)، و فقط اگر هنوز چیزی ذخیره نشده. ورود بازیکن یا جابه‌جایی حساب بازی را از اول اجرا می‌کند و بدون ذخیره‌ی محلی برچسب‌ها گم می‌شوند.
- با اولین درخواستی که بازیکن را می‌سازد بفرستیدشان، هم ورود معمولی خودتان و هم `/api/platform/token` ، هر کدام که زودتر اجرا شود.
- در سرور فقط هنگام ساخت بازیکن ذخیره‌شان کنید و بعداً هرگز بازنویسی نکنید.
- چهار فیلد جدا نگه دارید ( `utm_source` ، `utm_medium` ، `utm_campaign` ، `utm_content` )، نه یک رشته‌ی چسبیده. نام کمپین‌ها خودشان زیرخط دارند، پس مقدار چسبیده را نمی‌شود دوباره جدا کرد.
- به `linkedToken` دست نزنید. برچسب کمپین نیست: منطق ورود از آن استفاده می‌کند و در اجرای بعدی پاکش می‌کند.

## ساختن لینک کمپین

به تیم بازاریابی یک دستور ثابت یا یک سازنده‌ی لینک کوچک بدهید تا هر لینک **هر چهار** پارامتر را با املای یکسان داشته باشد: آدرس بازی شما به‌علاوه‌ی `utm_source`، `utm_medium`، `utm_campaign` و (اختیاری) `utm_content`، همه با حروف کوچک ASCII.

## لینک‌هایتان را تست کنید

- یک لینک کمپین را در پنجره‌ی خصوصی باز کنید. نوار آدرس بعد از بارگذاری باید تمیز باشد.
- با بازینس وارد شوید یا یک جابه‌جایی حساب راه بیندازید. بازیکن جدید هم باید همان کمپین را داشته باشد.
- همان لینک را با یک بازیکن **موجود** باز کنید. کمپین ذخیره‌شده‌اش نباید عوض شود.

کد آماده: خواندن و ذخیره‌ی برچسب‌ها، ساختن لینک‌ها.

## مهاجرت به کد اتصال و توکن‌های کوتاه‌عمر

پلتفرم توکن‌های بلندعمر کاربر را با توکن‌های ۱۵ دقیقه‌ای و کدهای اتصال یک‌بارمصرف جایگزین می‌کند. چه چیزی در سرور و کلاینت شما عوض می‌شود، قدم‌به‌قدم، و تا کی.

### چه چیزی عوض می‌شود و چرا

تا الان توکن کاربر بازینس ۱۰ سال اعتبار داشت. توی آدرس‌ها هم جابه‌جا می‌شد:

- بازی شما صفحه‌های بازینس را با `?linkedToken=<token>` باز می‌کرد؛
- اپ بازینس بازی شما را با `?linkedToken=<token>` باز می‌کرد.

آدرس‌ها در لاگ سرورها، ابزارهای آنالیتیکس، تاریخچه‌ی مرورگر و هدر Referer ثبت می‌شوند. توکنی که از آن‌جا پیدا می‌شد، تا ده سال به سکه‌ها و حساب بازیکن دسترسی می‌داد و خروج از حساب هم جلوییش را نمی‌گرفت.

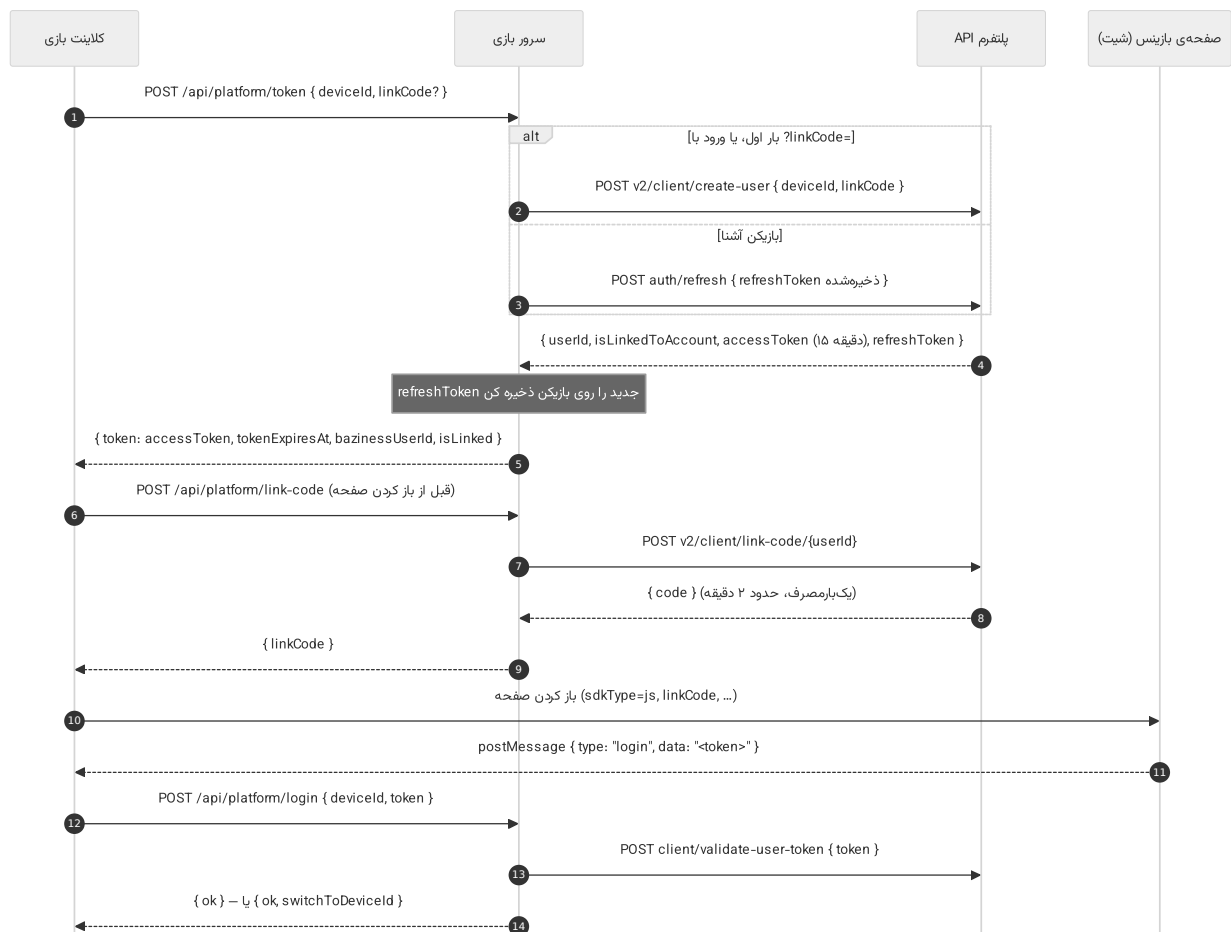
مدل جدید:

| قبلاً                                                                                                                              | حالا                                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| توکن دسترسی (چیزی که کلاینت به‌عنوان Bearer می‌فرستد)                                                                              | ۱۰ سال اعتبار                                                                                                    |
| توکن تمدید (refresh token)                                                                                                         | —                                                                                                                |
| توکن دسترسی را تمدید می‌کند؛ روی سرور شما نگه داشته می‌شود، هیچ‌وقت روی دستگاه؛ یک‌بارمصرف است و هر تمدید یک توکن تازه برمی‌گرداند |                                                                                                                  |
| جابه‌جا کردن بازیکن بین اپ‌ها                                                                                                      | <code>?linkCode=&lt;code&gt;</code> : یک کد یک‌بارمصرف با حدود ۲ دقیقه اعتبار که بعد از اولین استفاده بی‌اثر است |
|                                                                                                                                    | <code>?linkedToken=&lt;token&gt;</code>                                                                          |

### مهم

در دوره‌ی مهاجرت هر دو مدل کنار هم کار می‌کنند. endpoint های قدیمی و `?linkedToken=` کار می‌کنند و اپ بازینس هم `?linkCode=` و هم `?linkedToken=` را برای بازی‌ها می‌فرستد. تیم پلتفرم یک تاریخ پایان اعلام می‌کند: بعد از آن، توکن‌های قدیمی و `linkedToken` دیگر کار نمی‌کنند. مهاجرت را قبل از آن تاریخ انجام دهید.

جریان بعد از مهاجرت:



## تغییرات در یک نگاه

| امروز                                                                            | بعد از مهاجرت                                                                                                                     | کجا              |
|----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|------------------|
| با <code>client/create-user</code> با <code>linkedToken</code>                   | با <code>v2/client/create-user</code> با <code>linkCode</code> ؛<br><code>tokens.refreshToken</code> را ذخیره کنید                | سرور             |
| <code>client/get-user-token/{id}</code> در هر بار اجرا                           | با <code>auth/refresh</code> با توکن تمدید ذخیره شده؛<br><code>v2/client/get-user-token/{id}</code> فقط وقتی نشستی (session) نیست | سرور             |
| باز کردن صفحه‌ها با <code>linkedToken=&lt;token&gt;</code>                       | باز کردن صفحه‌ها با <code>linkCode=&lt;code&gt;</code> ؛ برای هر صفحه یک کد تازه از <code>v2/client/link-code/{id}</code>         | سرور +<br>کلاینت |
| اپ بازی‌ها را با <code>linkCode=?</code> و <code>linkedToken=?</code> باز می‌کند | <code>linkCode=?</code> : مثل توکن قبلی آن را به سرور تان بدهید                                                                   | کلاینت           |
| <code>GET client/validate-user-token/{token}</code>                              | <code>POST client/validate-user-token</code> با <code>{ token }</code> (توکن دیگر در آدرس نیست)                                   | سرور             |
| کلاینت توکنش را سال‌ها نگه می‌دارد                                               | کلاینت فقط توکن دسترسی را نگه می‌دارد؛ وقتی منقضی شد دوباره از سرور خودتان می‌گیرد                                                | کلاینت           |

باقی چیزها عوض نمی‌شوند: پیام‌های صفحه (`login`، `token`، `close`، `logout`)، سکه‌ها، جایزه‌ها و پرداخت مثل قبل کار می‌کنند.

## سرور، قدم به قدم

### ۱. برای هر بازیکن یک توکن تمدید نگه دارید

یک ستون به رکورد بازیکن اضافه کنید، مثلاً `business_refresh_token`. با آن مثل رمز عبور رفتار کنید:

- هیچ وقت لاگش نکنید؛
- هیچ وقت به کلاینت نفرستید؛
- اگر رازهای دیگر را رمزنگاری می‌کنید، این را هم رمزنگاری کنید.

همیشه با جدیدترین توکن جایگزینش کنید. هر `create-user`، `get-user-token` و `refresh` یک توکن تمدید تازه برمی‌گرداند و توکن قبلی دیگر کار نمی‌کند.

### ۲. `/api/platform/token`: به جای `get-user-token` تمدید کنید

این endpoint است که کلاینت در شروع اجرا صدا می‌زند. کلاینت حالا به جای `linkCode`، `linkedToken` می‌فرستد.

1. کلاینت `linkCode` فرستاده، یا بازیکن هنوز کاربر بازیکن ندارد: `v2/client/create-user` را با

`deviceId` و `linkCode` صدا بزنید. بعد `userId` برگشتی را مثل امروز بررسی کنید:

◦ `userId` مال بازیکن دیگری از بازیکنان شماست ← `switchToDeviceId` برگردانید (همان آدم روی دستگاه دیگر).

◦ بازیکن از قبل `userId` دیگری دارد ← یک حساب بازیکن دیگر رسیده است. آن را به عنوان بازیکن جدا شروع کنید و `switchToDeviceId` برگردانید (همان قاعده‌ی قدیمی «داده‌های محلی را پاک کن و از نو شروع کن»).

◦ در غیر این صورت ← `userId` و توکن تمدید را روی همین بازیکن ذخیره کنید.

◦ اگر کد منقضی یا مصرف‌شده بود ( `400 invalid_link_code` )، طوری ادامه دهید که انگار کدی نیامده.

2. بازیکن آشنا: `auth/refresh` را با توکن تمدید ذخیره‌شده صدا بزنید و توکن تازه را ذخیره کنید. جواب

`isLinkedToAccount` را هم دارد، پس همچنان می‌فهمید بازیکن کی وارد حسابش شده.

◦ `401 refresh_token_rotated` : درخواست دیگری از همین بازیکن لحظه‌ای قبل تمدید کرده. توکن تمدید را دوباره از دیتابیس بخوانید و یک بار دیگر امتحان کنید.

◦ هر `401` دیگر ( `session_revoked` ، `refresh_token_expired` ، `invalid_refresh_token` ،

`refresh_token_reused` ) : نشست تمام شده. با `v2/client/get-user-token/{userId}`

نشست تازه بگیرید و توکن تمدیدش را ذخیره کنید.

◦ `403 account_suspended` : حساب مسدود است.

3. به کلاینت توکن دسترسی و زمان انقضایش را بدهید: `{ token, tokenExpiresAt, businessUserId,`

`isLinkedToAccount }` توکن تمدید را هرگز نفرستید.

### پیشنهاد

بازیکن‌هایی که از قبل دارید هنوز توکن تمدید ندارند. قدم ۲ خودش آن‌ها را درست می‌کند: توکن تمدید ذخیره‌شده‌ای نیست ← یک بار `v2/client/get-user-token` ، و از آن به بعد `auth/refresh` .

### ۳. endpoint جدید `/api/platform/link-code` : یک کد برای هر صفحه

وقتی کلاینت می‌خواهد یکی از صفحه‌های بازیکن را باز کند، از سرور شما کد می‌خواهد. سرور شما `v2/client/link-code/{userId}` را صدا می‌زند و `{ linkCode }` را برمی‌گرداند.

- هر کد یک بار و حدود دو دقیقه کار می‌کند: برای هر صفحه‌ای که باز می‌کنید کد تازه بگیرید.
- فقط صفحه‌های بازیکن می‌توانند از آن استفاده کنند. حتی اگر لو بروند، نمی‌شود با آن وارد بازی شما شد.
- این endpoint را مثل بقیه‌ی endpointهای بازیکن محافظت کنید: فقط خود بازیکن می‌تواند کد خودش را بگیرد.

### ۴. `/api/platform/login` : بررسی با POST

منطق عوض نمی‌شود. فقط نوع درخواست عوض می‌شود: `POST client/validate-user-token` با `{ token }` در بدنه.

توکنی که در پیام `login` می‌رسد حالا ۱۵ دقیقه‌ای است. همان لحظه بررسی‌اش کنید، همان‌طور که امروز هم می‌کنید.

اگر توکن مال کاربر بازیکنی غیر از کاربر فعلی بازیکن بود، توکن تمدید ذخیره‌شده‌ی بازیکن را پاک کنید؛ فراخوانی بعدی `/token` نشست تازه شروع می‌کند.

## ۵. endpoint جدید /api/platform/logout

وقتی صفحه `logout` می‌فرستد، کلاینت قبل از شروع دوباره این را صدا می‌زند. سرور شما نشست را می‌بندد (`auth/logout`) و توکن تمدید ذخیره‌شده را پاک می‌کند.

### کد سرور

اضافه‌ها به کلاینت پلتفرم شما (کنار `createUser` / `getUserAccessToken`):

#### TypeScript (Node)

```
// platform/baziness.ts – ADD these next to createUser / getUserAccessToken (they reuse
API_BASE and platformFetch).

/** A player's session: the access token goes to your client, the refresh token stays in your
database. */
export interface PlatformSession {
  userId: number;
  isLinkedToAccount: boolean;
  accessToken: string;
  accessTokenExpiresAt: string;
  refreshToken: string;
}

const readSession = (j: Record<string, any>): PlatformSession => {
  const t = j.tokens ?? j; // v2 create-user / get-user-token nest the pair in `tokens`;
  auth/refresh returns it flat
  return {
    userId: Number(j.userId),
    isLinkedToAccount: Boolean(j.isLinkedToAccount),
    accessToken: String(t.accessToken),
    accessTokenExpiresAt: String(t.accessTokenExpiresAt),
    refreshToken: String(t.refreshToken),
  };
};

/** First time, or when the client brought a ?linkCode=. Throws (status 400) for an expired
or used code. */
export const createUserSession = async (deviceId: string, platform: string, store: string,
linkCode: string | null) =>
  readSession(await platformFetch("POST", "/v2/client/create-user", { deviceId, platform,
store, linkCode }));

/** A new session for a player you know – only when you have no refresh token or the session
ended. */
export const startUserSession = async (bazinessUserId: number) =>
  readSession(await platformFetch("POST", `/v2/client/get-user-token/${bazinessUserId}`));

/**
 * Renews a session. Returns the new session, "rotated" when a parallel request refreshed
first
 * (reload the stored token and retry once), or null when the session is over (start a new
one).
 */
export async function refreshUserSession(refreshToken: string): Promise<PlatformSession |
"rotated" | null> {
  const res = await fetch(`${API_BASE}/auth/refresh`, { // no client token: the refresh
token is the credential
```

```

    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ refreshToken }),
  });
  if (res.ok) return readSession(await res.json());
  const error = ((await res.json().catch(() => ({}))) as { error?: string }).error;
  if (res.status === 401 && error === "refresh_token_rotated") return "rotated";
  if (res.status === 403) throw Object.assign(new Error("account_suspended"), { status: 403 });
  if (res.status === 401) return null;
  throw new Error(`auth/refresh failed (${res.status})`); // 5xx / network: keep the old session, try later
}

/** Ends a session (player signed out). Never throws. */
export async function endUserSession(refreshToken: string): Promise<void> {
  await fetch(`${API_BASE}/auth/logout`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ refreshToken }),
  }).catch(() => {});
}

/** One-time code for opening ONE Baziness page as this player (~2 minutes). */
export const createPageLinkCode = async (bazinessUserId: number) =>
  String((await platformFetch("POST", `/v2/client/link-code/${bazinessUserId}`).code));

/** REPLACES the old GET version: the token now goes in the body, not in the URL. */
export async function validateUserAccessToken(token: string): Promise<number> {
  const j = await platformFetch("POST", "/client/validate-user-token", { token });
  if (!(j.isValid ?? j.IsValid)) throw new Error("invalid user token");
  return Number(j.userId ?? j.UserId ?? 0);
}

```

## C# (.NET)

```

// BazinessPlatformClient.cs - ADD these members (they reuse settings, http, SendAsync and Json).

public record PlatformSession(long UserId, bool IsLinkedToAccount, string AccessToken,
    DateTimeOffset AccessTokenExpiresAt, string RefreshToken);

public enum RefreshOutcome { Ok, Rotated, Ended }

static PlatformSession ReadSession(JsonElement j)
{
    // v2 create-user / get-user-token nest the pair in "tokens"; auth/refresh returns it flat.
    var t = j.TryGetProperty("tokens", out var nested) ? nested : j;
    return new PlatformSession(
        j.GetProperty("userId").GetInt64(),
        j.GetProperty("isLinkedToAccount").GetBoolean(),
        t.GetProperty("accessToken").GetString()!,
        t.GetProperty("accessTokenExpiresAt").GetDateTimeOffset(),
        t.GetProperty("refreshToken").GetString()!);
}

/// <summary>First time, or when the client brought a ?linkCode=. 400 = expired or used code.
</summary>

```

```

public async Task<PlatformSession> CreateUserSessionAsync(string deviceId, string platform,
string store, string? linkCode) =>
    ReadSession(await SendAsync(HttpMethod.Post, "/v2/client/create-user", new { deviceId,
platform, store, linkCode }));

/// <summary>A new session for a known player – only when there is no refresh token or the
session ended.</summary>
public async Task<PlatformSession> StartUserSessionAsync(long bazinessUserId) =>
    ReadSession(await SendAsync(HttpMethod.Post, $"/v2/client/get-user-
token/{bazinessUserId}"));

/// <summary>Renews a session. Rotated = a parallel request refreshed first (reload, retry
once); Ended = start a new one.</summary>
public async Task<(RefreshOutcome Outcome, PlatformSession? Session)>
RefreshUserSessionAsync(string refreshToken)
{
    // No client token here: the refresh token itself is the credential.
    var res = await http.PostAsJsonAsync($"{settings.ApiBase}/auth/refresh", new {
refreshToken }, Json);
    if (res.IsSuccessStatusCode)
        return (RefreshOutcome.Ok, ReadSession(await
res.Content.ReadFromJsonAsync<JsonElement>(Json)));

    string? error = null;
    var text = await res.Content.ReadAsStringAsync();
    if (!string.IsNullOrEmpty(text))
    {
        try { error = JsonDocument.Parse(text).RootElement.TryGetProperty("error", out var e)
? e.GetString() : null; }
        catch (JsonException) { /* not JSON */ }
    }
    return res.StatusCode switch
    {
        HttpStatusCode.Unauthorized when error == "refresh_token_rotated" =>
(RefreshOutcome.Rotated, null),
        HttpStatusCode.Unauthorized => (RefreshOutcome.Ended, null),
        HttpStatusCode.Forbidden => throw new InvalidOperationException("account_suspended"),
        _ => throw new HttpRequestException($"auth/refresh failed ({(int)res.StatusCode}")
    };
    // keep the old session
};
}

/// <summary>Ends a session (player signed out). Never throws.</summary>
public async Task EndUserSessionAsync(string refreshToken)
{
    try { await http.PostAsJsonAsync($"{settings.ApiBase}/auth/logout", new { refreshToken },
Json); }
    catch { /* best effort */ }
}

/// <summary>One-time code for opening ONE Baziness page as this player (~2 minutes).
</summary>
public async Task<string> CreatePageLinkCodeAsync(long bazinessUserId) =>
    (await SendAsync(HttpMethod.Post, $"/v2/client/link-
code/{bazinessUserId}")).GetProperty("code").GetString()!;

/// <summary>REPLACES the old GET version: the token now goes in the body, not in the URL.
</summary>
public async Task<long> ValidateUserAccessTokenAsync(string token)
{

```

```

var j = await SendAsync(HttpMethod.Post, "/client/validate-user-token", new { token });
if (!j.GetProperty("isValid").GetBoolean()) throw new InvalidOperationException("invalid
user token");
return j.GetProperty("userId").GetInt64();
}

```

## Python

```

# baziness/platform.py - ADD these methods to BazinessPlatform (they reuse API_BASE,
self._http and self._call).

@staticmethod
def _session(j: dict) -> dict:
    t = j.get("tokens") or j # v2 create-user / get-user-token nest the pair;
    auth/refresh returns it flat
    return {"userId": int(j["userId"]), "isLinkedToAccount":
bool(j["isLinkedToAccount"]),
        "accessToken": t["accessToken"], "accessTokenExpiresAt":
t["accessTokenExpiresAt"],
        "refreshToken": t["refreshToken"]}

def create_user_session(self, device_id: str, platform: str, store: str, link_code: str |
None) -> dict:
    """First time, or when the client brought a ?linkCode=. HTTPError 400 = expired or
used code."""
    return self._session(self._call("POST", "/v2/client/create-user", {
        "deviceId": device_id, "platform": platform, "store": store, "linkCode":
link_code}))

def start_user_session(self, baziness_user_id: int) -> dict:
    """A new session for a known player - only when there is no refresh token or the
session ended."""
    return self._session(self._call("POST", f"/v2/client/get-user-
token/{baziness_user_id}"))

def refresh_user_session(self, refresh_token: str):
    """Returns the new session, "rotated" (reload the stored token, retry once) or None
(session over)."""
    r = self._http.post(f"{API_BASE}/auth/refresh", json={"refreshToken": refresh_token},
timeout=10)
    if r.ok:
        return self._session(r.json())
    error = (r.json() if r.content else {}).get("error")
    if r.status_code == 401:
        return "rotated" if error == "refresh_token_rotated" else None
    if r.status_code == 403:
        raise PermissionError("account_suspended")
    r.raise_for_status() # 5xx: keep the old session, try later

def end_user_session(self, refresh_token: str) -> None:
    """Ends a session (player signed out). Never raises."""
    try:
        self._http.post(f"{API_BASE}/auth/logout", json={"refreshToken": refresh_token},
timeout=10)
    except requests.RequestException:
        pass

def create_page_link_code(self, baziness_user_id: int) -> str:
    """One-time code for opening ONE Baziness page as this player (~2 minutes)."""

```

```

        return str(self._call("POST", f"/v2/client/link-code/{baziiness_user_id}")[ "code" ])

    def validate_user_access_token(self, token: str) -> int:
        """REPLACES the old GET version: the token now goes in the body, not in the URL."""
        j = self._call("POST", "/client/validate-user-token", {"token": token})
        if not _pick(j, "isValid", False):
            raise ValueError("invalid user token")
        return int(_pick(j, "userId", 0))

```

چهار endpoint:

#### TypeScript (Node)

```

import {
    createUserSession, startUserSession, refreshUserSession, endUserSession,
    createPageLinkCode,
    validateUserAccessToken, platformClientConfig, platformEnabled, type PlatformSession,
} from "./platform/baziiness";
import { db } from "./db"; // your own data layer. New column on the player:
baziiness_refresh_token (a secret)

// What the client gets. The refresh token never leaves your server.
const forClient = (s: PlatformSession) => ({
    token: s.accessToken, tokenExpiresAt: s.accessTokenExpiresAt,
    baziinessUserId: s.userId, isLinkedToAccount: s.isLinkedToAccount,
});

async function keep(playerId: number, s: PlatformSession) {
    await db.setRefreshToken(playerId, s.refreshToken); // always the NEWEST one: the old one
    stops working
    return s;
}

/** A valid session for a player who already has a Baziiness user. */
async function sessionFor(player: { id: number; baziinessUserId: number; baziinessRefreshToken:
string | null }) {
    if (player.baziinessRefreshToken) {
        let r = await refreshUserSession(player.baziinessRefreshToken);
        if (r === "rotated") { // another request of this player refreshed
a moment ago
            const latest = await db.getRefreshToken(player.id);
            r = latest ? await refreshUserSession(latest) : null;
            if (r === "rotated") r = null;
        }
        if (r) return keep(player.id, r);
    }
    // No refresh token yet (players from before the migration) or the session ended.
    return keep(player.id, await startUserSession(player.baziinessUserId));
}

app.post("/api/platform/token", async (req, res) => {
    const deviceId = String(req.body.deviceId ?? "").trim();
    const linkCode: string | null = req.body.linkCode ?? null; // replaces
linkedToken
    if (!deviceId) return res.status(400).json({ error: "deviceId required" });
    if (!platformEnabled()) return res.json({ enabled: false });

    const config = { enabled: true, ...platformClientConfig() };

```

```

const me = await db.ensureUserByDevice(deviceId, { utm: req.body.utm, tag: req.body.tag });
try {
  if (linkCode || !me.bazinessUserId) {
    // First time, or arrived from the Baziness app with ?linkCode=: which Baziness user is
    // this?
    const s = await createUserSession(deviceId, "Web", "Direct", linkCode)
      .catch((e) => (linkCode && e.status === 400 ? null : Promise.reject(e))); // expired
    // or used code
    if (s) {
      const owner = await db.findUserByBazinessId(s.userId);
      if (owner && owner.id !== me.id) { // the same person, another
        // player of yours
        await keep(owner.id, s);
        return res.json({ ...config, switchToDeviceId: owner.deviceId });
      }
      if (me.bazinessUserId && me.bazinessUserId !== s.userId) {
        // A different Baziness account arrived on a device that already has a player:
        // it becomes its own player (the old "clear local data and start again" rule).
        const fresh = await db.createPlayerForBaziness(s.userId, s.isLinkedToAccount);
        await keep(fresh.id, s);
        return res.json({ ...config, switchToDeviceId: fresh.deviceId });
      }
      await db.linkBaziness(me.id, s.userId, s.isLinkedToAccount);
      await keep(me.id, s);
      return res.json({ ...config, ...forClient(s) });
    }
    if (!me.bazinessUserId) { // stale code and no Baziness
      // user yet
      const s2 = await createUserSession(deviceId, "Web", "Direct", null);
      await db.linkBaziness(me.id, s2.userId, s2.isLinkedToAccount);
      await keep(me.id, s2);
      return res.json({ ...config, ...forClient(s2) });
    }
  }
  const s = await sessionFor(me);
  if (s.isLinkedToAccount !== me.isBazinessLinked) await db.setLinked(me.id,
s.isLinkedToAccount);
  return res.json({ ...config, ...forClient(s) });
} catch (err) {
  console.error("[platform] token failed:", (err as Error).message);
  const suspended = (err as { status?: number }).status === 403;
  return res.json({ ...config, error: suspended ? "account_suspended" :
"platform_unavailable" }); // never block the boot
}
});

// Protect it like your other player endpoints: only the player themself may ask for their
// code.
app.post("/api/platform/link-code", async (req, res) => {
  const me = await db.findUserByDevice(String(req.body.deviceId ?? ""));
  if (!me?.bazinessUserId) return res.status(409).json({ error: "call /api/platform/token
first" });
  try {
    return res.json({ linkCode: await createPageLinkCode(me.bazinessUserId) });
  } catch {
    return res.status(503).json({ error: "platform_unavailable" });
  }
});

app.post("/api/platform/login", async (req, res) => {

```

```

const deviceId = String(req.body.deviceId ?? "").trim();
const token = String(req.body.token ?? "").trim();
if (!deviceId || !token) return res.status(400).json({ error: "deviceId and token required"
});
if (!platformEnabled()) return res.json({ ok: false, reason: "disabled" });
try {
    const bazinessUserId = await validateUserAccessToken(token); // POST now; verify
right away (15-min token)
    const owner = await db.findUserByBazinessId(bazinessUserId);
    const me = await db.ensureUserByDevice(deviceId);
    if (owner && owner.id !== me.id) return res.json({ ok: true, switchToDeviceId:
owner.deviceId });
    if (me.bazinessUserId !== bazinessUserId) await db.setRefreshToken(me.id, null); //
belonged to the old user
    await db.linkBaziness(me.id, bazinessUserId, true);
    return res.json({ ok: true }); // the client then
calls /token again
} catch {
    return res.json({ ok: false, reason: "invalid" });
}
});

// The page sent `logout`: end the session before the client starts over as a new visitor.
app.post("/api/platform/logout", async (req, res) => {
    const me = await db.findUserByDevice(String(req.body.deviceId ?? ""));
    if (me?.bazinessRefreshToken) {
        await endUserSession(me.bazinessRefreshToken);
        await db.setRefreshToken(me.id, null);
    }
    return res.json({ ok: true });
});

```

## C# (.NET)

```

// IUserStore gains: SetRefreshTokenAsync(id, token?), GetRefreshTokenAsync(id),
FindByDeviceAsync(deviceId),
// CreatePlayerForBazinessAsync(bazinessUserId, isLinked). Store the
refresh token like a password.

// What the client gets. The refresh token never leaves your server.
static object ForClient(PlatformSettings pub, PlatformSession s) => new
{
    enabled = true, pub.WebBase, pub.ApiBase, pub.PackageName,
    token = s.AccessToken, tokenExpiresAt = s.AccessTokenExpiresAt,
    bazinessUserId = s.UserId, isLinkedToAccount = s.IsLinkedToAccount
};

static async Task<PlatformSession> SessionForAsync(Player player, BazinessPlatformClient
platform, IUserStore users)
{
    var refreshToken = player.BazinessRefreshToken;
    for (var attempt = 0; refreshToken is not null && attempt < 2; attempt++)
    {
        var (outcome, session) = await platform.RefreshUserSessionAsync(refreshToken);
        if (outcome == RefreshOutcome.Ok)
        {
            await users.SetRefreshTokenAsync(player.Id, session!.RefreshToken); // always the
newest one
            return session;
        }
    }
}

```

```

    }
    if (outcome == RefreshOutcome.Ended) break;
    refreshToken = await users.GetRefreshTokenAsync(player.Id);           // Rotated: a
parallel request won
    }
    // No refresh token yet (players from before the migration) or the session ended.
    var fresh = await platform.StartUserSessionAsync(player.BazinessUserId!.Value);
    await users.SetRefreshTokenAsync(player.Id, fresh.RefreshToken);
    return fresh;
}

app.MapPost("/api/platform/token", async (TokenRequest body, BazinessPlatformClient platform,
    IUserStore users, PlatformSettings pub, ILogger<Program> log) =>
{
    if (string.IsNullOrEmpty(body.DeviceId)) return Results.BadRequest(new { error =
"deviceId required" });
    if (!pub.IsOn) return Results.Ok(new { enabled = false });

    var me = await users.EnsureByDeviceAsync(body.DeviceId);
    try
    {
        if (body.LinkCode is not null || me.BazinessUserId is null)
        {
            PlatformSession? s = null;
            try { s = await platform.CreateUserSessionAsync(body.DeviceId, "Web", "Direct",
body.LinkCode); }
            catch (HttpRequestException e) when (body.LinkCode is not null && e.StatusCode ==
HttpStatusCode.BadRequest) { }

            if (s is not null)
            {
                var owner = await users.FindByBazinessUserIdAsync(s.UserId);
                if (owner is not null && owner.Id != me.Id)
                {
                    await users.SetRefreshTokenAsync(owner.Id, s.RefreshToken);
                    return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase,
pub.PackageName, switchToDeviceId = owner.DeviceId });
                }
                if (me.BazinessUserId is not null && me.BazinessUserId != s.UserId)
                {
                    var created = await users.CreatePlayerForBazinessAsync(s.UserId,
s.IsLinkedToAccount);
                    await users.SetRefreshTokenAsync(created.Id, s.RefreshToken);
                    return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase,
pub.PackageName, switchToDeviceId = created.DeviceId });
                }
                await users.LinkBazinessAsync(me.Id, s.UserId, s.IsLinkedToAccount);
                await users.SetRefreshTokenAsync(me.Id, s.RefreshToken);
                return Results.Ok(ForClient(pub, s));
            }
            if (me.BazinessUserId is null)                               // stale code and no Baziness
user yet
            {
                var s2 = await platform.CreateUserSessionAsync(body.DeviceId, "Web",
"Direct", null);
                await users.LinkBazinessAsync(me.Id, s2.UserId, s2.IsLinkedToAccount);
                await users.SetRefreshTokenAsync(me.Id, s2.RefreshToken);
                return Results.Ok(ForClient(pub, s2));
            }
        }
    }
}

```

```

        var session = await SessionForAsync(me, platform, users);
        if (session.IsLinkedToAccount != me.IsBazinessLinked) await
users.SetLinkedAsync(me.Id, session.IsLinkedToAccount);
        return Results.Ok(ForClient(pub, session));
    }
    catch (Exception ex)
    {
        log.LogError(ex, "platform token failed");
        var error = ex.Message == "account_suspended" ? "account_suspended" :
"platform_unavailable";
        return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase, pub.PackageName,
error }); // never block the boot
    }
});

// Protect it like your other player endpoints: only the player themself may ask for their
code.
app.MapPost("/api/platform/link-code", async (DeviceRequest body, BazinessPlatformClient
platform, IUserStore users) =>
{
    var me = await users.FindByDeviceAsync(body.DeviceId);
    if (me?.BazinessUserId is null) return Results.Conflict(new { error = "call
/api/platform/token first" });
    try { return Results.Ok(new { linkCode = await
platform.CreatePageLinkCodeAsync(me.BazinessUserId.Value) }); }
    catch { return Results.StatusCode(503); }
});

app.MapPost("/api/platform/login", async (LoginRequest body, BazinessPlatformClient platform,
IUserStore users, PlatformSettings pub) =>
{
    if (string.IsNullOrEmpty(body.DeviceId) || string.IsNullOrEmpty(body.Token))
        return Results.BadRequest(new { error = "deviceId and token required" });
    if (!pub.IsOn) return Results.Ok(new { ok = false, reason = "disabled" });
    try
    {
        long bazinessUserId = await platform.ValidateUserAccessTokenAsync(body.Token); //
POST now; verify right away
        var owner = await users.FindByBazinessUserIdAsync(bazinessUserId);
        var me = await users.EnsureByDeviceAsync(body.DeviceId);
        if (owner is not null && owner.Id != me.Id) return Results.Ok(new { ok = true,
switchToDeviceId = owner.DeviceId });
        if (me.BazinessUserId != bazinessUserId) await users.SetRefreshTokenAsync(me.Id,
null); // belonged to the old user
        await users.LinkBazinessAsync(me.Id, bazinessUserId, isLinked: true);
        return Results.Ok(new { ok = true }); //
the client then calls /token again
    }
    catch { return Results.Ok(new { ok = false, reason = "invalid" }); }
});

// The page sent `logout`: end the session before the client starts over as a new visitor.
app.MapPost("/api/platform/logout", async (DeviceRequest body, BazinessPlatformClient
platform, IUserStore users) =>
{
    var me = await users.FindByDeviceAsync(body.DeviceId);
    if (me?.BazinessRefreshToken is { } refreshToken)
    {
        await platform.EndUserSessionAsync(refreshToken);
    }
});

```

```

        await users.SetRefreshTokenAsync(me.Id, null);
    }
    return Results.Ok(new { ok = true });
});

public record TokenRequest(string DeviceId, string? LinkCode);
public record DeviceRequest(string DeviceId);
public record LoginRequest(string DeviceId, string Token);

```

## Python

```

from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from requests import HTTPError
from baziness.platform import platform, platform_client_config, platform_enabled
from app import db # your own data layer. New column on the player: baziness_refresh_token
(a secret)

app = FastAPI()

class TokenRequest(BaseModel):
    deviceId: str
    linkCode: str | None = None # replaces linkedToken

class DeviceRequest(BaseModel):
    deviceId: str

class LoginRequest(BaseModel):
    deviceId: str
    token: str

def for_client(s: dict) -> dict: # the refresh token never leaves your server
    return {"token": s["accessToken"], "tokenExpiresAt": s["accessTokenExpiresAt"],
            "bazinessUserId": s["userId"], "isLinkedToAccount": s["isLinkedToAccount"]}

def keep(player_id: int, s: dict) -> dict:
    db.set_refresh_token(player_id, s["refreshToken"]) # always the newest one
    return s

def session_for(player) -> dict:
    token = player.baziness_refresh_token
    for _ in range(2):
        if not token:
            break
        r = platform.refresh_user_session(token)
        if isinstance(r, dict):
            return keep(player.id, r)
        if r is None: # the session ended
            break
        token = db.get_refresh_token(player.id) # "rotated": a parallel request won
    # No refresh token yet (players from before the migration) or the session ended.
    return keep(player.id, platform.start_user_session(player.baziness_user_id))

```

```

@app.post("/api/platform/token")
def platform_token(body: TokenRequest):
    if not platform_enabled():
        return {"enabled": False}
    config = {"enabled": True, **platform_client_config()}
    me = db.ensure_user_by_device(body.deviceId)
    try:
        if body.linkCode or not me.baziness_user_id:
            try:
                s = platform.create_user_session(body.deviceId, "Web", "Direct",
body.linkCode)
            except HTTPError as e:
                if not (body.linkCode and e.response.status_code == 400):
                    raise
                s = None # expired or used code
            if s:
                owner = db.find_user_by_baziness_user_id(s["userId"])
                if owner and owner.id != me.id: # the same person, another player of
yours
                    keep(owner.id, s)
                    return {**config, "switchToDeviceId": owner.device_id}
                if me.baziness_user_id and me.baziness_user_id != s["userId"]:
                    fresh = db.create_player_for_baziness(s["userId"],
s["isLinkedToAccount"])
                    keep(fresh.id, s)
                    return {**config, "switchToDeviceId": fresh.device_id}
                db.link_baziness(me.id, s["userId"], s["isLinkedToAccount"])
                return {**config, **for_client(keep(me.id, s))}
            if not me.baziness_user_id: # stale code and no Baziness user yet
                s2 = platform.create_user_session(body.deviceId, "Web", "Direct", None)
                db.link_baziness(me.id, s2["userId"], s2["isLinkedToAccount"])
                return {**config, **for_client(keep(me.id, s2))}
        s = session_for(me)
        if s["isLinkedToAccount"] != me.is_baziness_linked:
            db.set_linked(me.id, s["isLinkedToAccount"])
        return {**config, **for_client(s)}
    except PermissionError:
        return {**config, "error": "account_suspended"}
    except Exception as err: # never block the boot
        print("[platform] token failed:", err)
        return {**config, "error": "platform_unavailable"}

@app.post("/api/platform/link-code") # protect it like your other player endpoints
def platform_link_code(body: DeviceRequest):
    me = db.find_user_by_device(body.deviceId)
    if not me or not me.baziness_user_id:
        raise HTTPException(409, "call /api/platform/token first")
    return {"linkCode": platform.create_page_link_code(me.baziness_user_id)}

@app.post("/api/platform/login")
def platform_login(body: LoginRequest):
    if not platform_enabled():
        return {"ok": False, "reason": "disabled"}
    try:
        baziness_user_id = platform.validate_user_access_token(body.token) # POST now;
verify right away
    except Exception:

```

```

        return {"ok": False, "reason": "invalid"}
    owner = db.find_user_by_baziness_user_id(baziness_user_id)
    me = db.ensure_user_by_device(body.deviceId)
    if owner and owner.id != me.id:
        return {"ok": True, "switchToDeviceId": owner.device_id}
    if me.baziness_user_id != baziness_user_id:
        db.set_refresh_token(me.id, None) # belonged to
the old user
    db.link_baziness(me.id, baziness_user_id, True)
    return {"ok": True} # the client
then calls /token again

@app.post("/api/platform/logout") # the page sent `logout`
def platform_logout(body: DeviceRequest):
    me = db.find_user_by_device(body.deviceId)
    if me and me.baziness_refresh_token:
        platform.end_user_session(me.baziness_refresh_token)
        db.set_refresh_token(me.id, None)
    return {"ok": True}

```

## کلاینت، قدم به قدم

1. در شروع اجرا، به جای `linkedToken`، `linkCode` را (از `?linkCode=` در آدرس یا از `deep link`) به `/api/platform/token` بفرستید. بعد از آن `linkCode` و `linkedToken` را از آدرس حذف کنید: هر کد یک بار کار می‌کند.
2. توکن را برای اجرای بعدی ذخیره نکنید (`localStorage`، `PlayerPrefs`). سرور شما نشست را نگه می‌دارد و با هر `/api/platform/token` یک توکن تازه می‌دهد.
3. قبل از استفاده از توکن (`user/get-coin`) `tokenExpiresAt` را نگاه کنید. اگر تا یک دقیقه‌ی دیگر منقضی می‌شود، یا درخواستی `401` گرفت، دوباره `/api/platform/token` را صدا بزنید.
4. برای باز کردن صفحه‌ی بازی، از `/api/platform/link-code` کد بگیرید و به جای `linkedToken`، `linkCode=<code>` را در آدرس صفحه بگذارید. بقیه‌ی پارامترها همان قبلی‌ها هستند (صفحه‌ها).
5. با پیام `logout` صفحه، `/api/platform/logout` را صدا بزنید و بعد مثل قبل از نو شروع کنید.

### TypeScript

```

// PlatformController – the parts that change.

// 1. Boot: forward the one-time ?linkCode= (it replaces ?linkedToken=). Don't keep the token
in
// localStorage any more: your server keeps the session and gives you a fresh token on
demand.
async init(): Promise<boolean> {
    const params = new URLSearchParams(location.search);
    const linkCode = params.get("linkCode");
    const res = await fetch("/api/platform/token", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ deviceId: this.deviceId, linkCode }),
    });
    const data = await res.json();
    if (data.switchToDeviceId) return this.switchTo(data.switchToDeviceId); // unchanged
}

```

```

    this.token = data.token ?? null;
    this.tokenExpiresAt = data.tokenExpiresAt ? Date.parse(data.tokenExpiresAt) : 0;
    if (linkCode || params.has("linkedToken")) { // a code works once
        params.delete("linkCode"); params.delete("linkedToken");
        const rest = params.toString();
        history.replaceState(null, "", rest ? `${location.pathname}?${rest}` : location.pathname);
    }
    return !!this.token;
}

// 2. The token lives 15 minutes: ask your server again shortly before it expires, or after a 401.
async accessToken(): Promise<string | null> {
    if (!this.token || this.tokenExpiresAt - Date.now() < 60_000) await this.init();
    return this.token;
}

async getCoin(): Promise<number> {
    const call = async () => fetch(`${this.apiBase}/user/get-coin`, {
        headers: { Authorization: `Bearer ${await this.accessToken()}` },
    });
    let res = await call();
    if (res.status === 401) { this.token = null; res = await call(); } // renew once, then give up
    return res.ok ? Number((await res.json()).amount ?? 0) : 0;
}

// 3. Pages: a fresh one-time code for every page you open, instead of the token.
async pageUrl(path: string, o: PageOptions = {}): Promise<string> {
    const res = await fetch("/api/platform/link-code", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ deviceId: this.deviceId }),
    });
    const { linkCode } = await res.json();
    const url = new URL(bazinessPageUrl(this.webBase, path, this.packageName, null, o)); // no token any more
    url.searchParams.set("linkCode", linkCode);
    return url.toString();
}

// 4. The page sent `logout`: tell your server first, then start over as before.
async onPageLogout() {
    await fetch("/api/platform/logout", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ deviceId: this.deviceId }),
    }).catch(() => {});
    this.resetAndRestart(); // unchanged: delete local data, new visitor
}

```

## C# (Unity)

```

// BazinessLogin - the parts that change.
public DateTimeOffset TokenExpiresAt { get; private set; }

[Serializable] class LinkCodeResponse { public string linkCode; }

```

```

// 1. Boot: forward the one-time ?linkCode= / deep-link code (it replaces linkedToken).
// Stop saving the token in PlayerPrefs: your server keeps the session.
public IEnumerator Init(string deviceId, string linkCode = null)
{
    var body = $"{{\"deviceId\": \"{deviceId}\", \"linkCode\": {Quote(linkCode)}}}";
    using var req = Post("/api/platform/token", body);
    yield return req.SendWebRequest();
    var r = JsonUtility.FromJson<TokenResponse>(req.downloadHandler.text); // TokenResponse
    gains: string tokenExpiresAt
    if (!string.IsNullOrEmpty(r.switchToDeviceId)) { SwitchTo(r.switchToDeviceId); yield
break; } // unchanged
    Token = r.token;
    TokenExpiresAt = DateTimeOffset.TryParse(r.tokenExpiresAt, out var t) ? t :
DateTimeOffset.MinValue;
}

// 2. The token lives 15 minutes: renew it through your server before using it.
public IEnumerator EnsureFreshToken(string deviceId)
{
    if (string.IsNullOrEmpty(Token) || TokenExpiresAt - DateTimeOffset.UtcNow <
(TimeSpan.FromMinutes(1)))
        yield return Init(deviceId);
}

// 3. Pages: a fresh one-time code for every page you open, instead of the token.
public IEnumerator OpenPage(string deviceId, string path, Action<string> open,
    string redirect = null, bool allowGuest = true, string extra = null)
{
    using var req = Post("/api/platform/link-code", $"{{\"deviceId\": \"{deviceId}\", \"linkCode\": \"{linkCode}\"}}");
    yield return req.SendWebRequest();
    var linkCode = JsonUtility.FromJson<LinkCodeResponse>(req.downloadHandler.text).linkCode;
    open(PageUrl(path, linkCode, redirect, allowGuest, extra));
}

public string PageUrl(string path, string linkCode, string redirect = null, bool allowGuest =
true, string extra = null)
{
    var url = $"{webBase.TrimEnd('/')}{path}?packageName=
{Uri.EscapeDataString(packageName)}&sdkType=js";
    if (!string.IsNullOrEmpty(linkCode)) url += "&linkCode=
{Uri.EscapeDataString(linkCode)}"; // was linkedToken
    if (!string.IsNullOrEmpty(redirect)) url += "&redirect={redirect}";
    if (!allowGuest) url += "&allowGuest=false";
    if (!string.IsNullOrEmpty(extra)) url += "&{extra}";
    return url;
}

```

## endpointهای جدید پلتفرم

همه زیر همان `{api}` همیشگی هستند. endpointهای `client` مثل قبل توکن کلاینت لازم دارند.

## ساخت کاربر همراه با نشست

HTTP

```

POST {api}/v2/client/create-user
Authorization: Bearer <client token>
Content-Type: application/json

{ "deviceId": "8f1c2a...", "platform": "Android", "store": "CafeBazaar", "linkCode": null }

→ 200 OK
{
  "userId": 239236,
  "token": "eyJhbGciOiJIUzI1NiIs...", // the access token (same as
tokens.accessToken)
  "isLinkedToAccount": false,
  "tokens": {
    "accessToken": "eyJhbGciOiJIUzI1NiIs...", // 15 minutes – the client uses this
    "accessTokenExpiresAt": "2026-10-22T10:15:00+00:00",
    "refreshToken": "rt1.dkUoqaDDG8YB15KAQN01q77FE...", // keep on YOUR server only
    "refreshTokenExpiresAt": "2026-11-21T10:00:00+00:00"
  }
}

→ 400 { "success": false, "error": "invalid_link_code" } // the code expired or was already
used

```

## تمدید نشست

توکن کلاینت لازم نیست: خود توکن تمدید مدرک است. این را از سرور خودتان صدا بزنید.

### HTTP

```

POST {api}/auth/refresh
Content-Type: application/json

{ "refreshToken": "rt1.dkUoqaDDG8YB15KAQN01q77FE..." }

→ 200 OK
{
  "userId": 239236,
  "isLinkedToAccount": true,
  "accessToken": "eyJhbGciOiJIUzI1NiIs...",
  "accessTokenExpiresAt": "2026-10-22T10:30:00+00:00",
  "refreshToken": "rt1.Wn0QhZ...", // store THIS one – the old one no longer
works
  "refreshTokenExpiresAt": "2026-11-21T10:15:00+00:00"
}

→ 401 { "success": false, "error": "refresh_token_rotated" } // another request refreshed
first: reload yours, retry once
→ 401 { "success": false, "error": "session_revoked" } // also invalid_refresh_token,
refresh_token_expired,
// refresh_token_reused: get a
new session
→ 403 { "success": false, "error": "account_suspended" }

```

HTTP

```
POST {api}/v2/client/get-user-token/239236
```

Authorization: Bearer <client token>

→ 200 OK

```
{ "userId": 239236, "token": "eyJhbGciOiJIUzI1NiIs... ", "isLinkedToAccount": true,
  "tokens": { "accessToken": "...", "accessTokenExpiresAt": "...", "refreshToken": "rt1...",
"refreshTokenExpiresAt": "..."} }
```

```
→ 403 { "success": false, "error": "access denied" } // the user belongs to another game
```

## کد اتصال برای باز کردن صفحه

HTTP

POST {api}/v2/client/link-code/239236

Authorization: Bearer <client token>

→ 200 OK

```
{ "code": "q3Jc9yX2tL0...", "expiresAt": "2026-10-22T10:02:00+00:00" } // one page, once, ~2
minutes
```

```
→ 403 { "success": false, "error": "access denied" } // the user belongs to another game
```

### بررسی توکن (در بدنه به جای آدرس)

HTTP

POST {api}/client/validate-user-token

**Authorization:** Bearer <client token>

Content-Type: application/json

```
{ "token": "eyJhbGciOiJIUzI1NiIs..." }
```

→ 200 OK

```
{ "userId": 239236, "isValid": true }
```

```
→ 200 OK    (invalid or expired: not an HTTP error)
```

```
{ "userId": 0, "isValid": false }
```

## بستن نشست

POST {api}/auth/logout با { "refreshToken": "rt1...." }. همیشه 200 جواب می‌دهد.

| پاسخ                                                                                             | کجا                                                       | یعنی                                          | چه کنید                                                |
|--------------------------------------------------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------|--------------------------------------------------------|
| 401<br>refresh_token_rotated                                                                     | auth/refresh                                              | درخواست دیگری از همین بازیکن زودتر تمدید کرده | توکن تمدید را دوباره بخوانید و یک بار دیگر امتحان کنید |
| 401<br>session_revoked<br>refresh_token_expired<br>invalid_refresh_token<br>refresh_token_reused | auth/refresh                                              | نشست تمام شده                                 | v2/client/get-user-token/{id} و ذخیره توکن تمدید تازه  |
| 403<br>account_suspended                                                                         | auth/refresh                                              | حساب مسدود است                                | پیام مسدودی را نشان دهید                               |
| 400<br>invalid_link_code                                                                         | v2/client/create-user                                     | کد منقضی یا مصرف شده، یا برای بازی شما نبوده  | بدون کد ادامه دهید                                     |
| 403                                                                                              | v2/client/link-code/{id}<br>v2/client/get-user-token/{id} | کاربر مال بازی دیگری است                      | userId ذخیره شده را بررسی کنید                         |
| 401                                                                                              | user/get-coin (کلاینت)                                    | توکن ۱۵ دقیقه‌ای منقضی شده                    | دوباره /api/platform/token و یک بار تکرار              |

## چک لیست

روی پلتفرم توسعه تست کنید (هر شماره‌ای، کد 111111):

- ☐ بازیکن جدید بازی را باز می‌کند: کلاینت توکن می‌گیرد و دیتابیس شما برایش توکن تمدید دارد.

- اجرای دوباره: `auth/refresh` استفاده می‌شود (نه `get-user-token`) و توکن تمدید ذخیره شده عوض شده. ☐
- دو اجرای هم‌زمان یک بازیکن: هر دو موفق‌اند (یکی با تکرار بعد از `rotated`). ☐
- بازیکنی از قبل از مهاجرت یک بار بازی را باز می‌کند و توکن تمدید می‌گیرد. ☐
- همه‌ی صفحه‌های بازی‌نس با `linkCode=...` و بدون `linkedToken` باز می‌شوند؛ صفحه‌ی ورود، پروفایل، فروشگاه جایزه و صفحه‌ی پیشنهاد همه کار می‌کنند. ☐
- ورود در صفحه با شماره‌ای که مال بازیکن دیگری از بازیکنان شماست، همچنان به آن بازیکن جابه‌جا می‌شود. ☐
- باز کردن بازی از اپ بازی‌نس با `?linkCode=` به حساب درست می‌رسد؛ همان لینک را دو بار باز کنید ← بار دوم کد نادیده گرفته می‌شود و بازی باز هم بالا می‌آید. ☐
- بعد از ۱۵ دقیقه بازی، موجودی سکه هنوز بارگذاری می‌شود (توکن از طریق سرور شما تمدید شده). ☐
- از صفحه نشست را می‌بندد؛ اجرای بعدی یک بازدیدکننده‌ی جدید است. `logout` ☐
- در لاگ‌های شما توکن تمدید نیست و هیچ توکنی در آدرس‌ها نیست. ☐

#### نکته

تا تاریخ پایان، اپ بازی‌نس `?linkedToken=` را هم می‌فرستد. وقتی بازی شما `linkCode` را می‌خواند، `linkedToken` را نادیده بگیرید. بعد از تاریخ پایان دیگر فرستاده نمی‌شود و توکن‌های قدیمی هم کار نمی‌کنند.

## کد سرور

کد آماده‌ی سرور برای همه‌ی قدم‌ها (کلاینت پلتفرم، endpointهای خودتان، پرداخت گیتا کوین، پرداخت بازی‌نس و برچسب‌های کمپین) به C، TypeScript و Python.

این نمونه‌ها **قدم‌های API** و **راهنماها** را پیاده می‌کنند. هر کدام به سه زبان **C# (.NET)**، **Python** و **TypeScript (Node)** آمده است؛ یک زبانه را انتخاب کنید و انتخابتان در همه‌ی صفحه‌ها به خاطر سپرده می‌شود. از Express، ASP.NET، minimal API و FastAPI استفاده شده، ولی منطق در هر فریم‌ورکی همین است. هر جا **db...** صدا زده شده، دیتابیس خودتان را بگذارید.

## تنظیمات

.env (server)

```
# --- Baziness platform: accounts, Gita coins, reward pages -----
BAZINESS_PLATFORM_ENABLED=false          # master switch: nothing shows until "true"
BAZINESS_PLATFORM_USE_PRODUCTION=false    # selects the default API + web URLs below
BAZINESS_CLIENT_ID=
BAZINESS_CLIENT_SECRET=
BAZINESS_PLATFORM_PACKAGE_NAME=com.yourstudio.yourgame
# Optional overrides (derived from USE_PRODUCTION when blank):
# dev api https://platform-dev.karizmastudios.org/api    web https://platform-web-dev.karizmastudios.org
# prod api https://platform.karizmastudio.com/api        web https://baziness.com
BAZINESS_PLATFORM_API_URL=
BAZINESS_PLATFORM_WEB_URL=
BAZINESS_GAME_NAME=yourgame              # ?game_name= of your pages on the platform
(rewards, game offer)
# Optional: the tutorial offer page (blank = {web}/game-offer?game_name=<BAZINESS_GAME_NAME>)
BAZINESS_GAME_OFFER_URL=

# --- Baziness Pay: real-money purchases -----
BAZINESS_PAY_ENABLED=false               # buy buttons stay disabled until "true"
# dev https://pay-dev.karizmastudios.org    prod https://pay.baziness.com
BAZINESS_PAY_BASE_URL=https://pay-dev.karizmastudios.org
BAZINESS_PAY_PACKAGE_NAME=com.yourstudio.yourgame # the Pay app's package (pname) - may
differ from BAZINESS_PLATFORM_PACKAGE_NAME
BAZINESS_SKU_PREFIX=                    # optional prefix: SKU = prefix + product id
DEV_FAKE_BAZINESS_PAY=false              # local development only (see the Pay page)
```

## کلاینت پلتفرم

یک کلاس کوچک که **قدم‌های ۱ تا ۵** را انجام می‌دهد: با اولین استفاده توکن کلاینت را می‌گیرد، بعد از **401** دوباره می‌گیرد و شناسه‌ی پرداخت را از کلید پرداخت می‌سازد.

TypeScript (Node)

```

// platform/baziness.ts - all credentials stay on the server.
import { createHash } from "node:crypto";

const DEV_API = "https://platform-dev.karizmastudios.org/api";
const PROD_API = "https://platform.karizmastudio.com/api";
const DEV_WEB = "https://platform-web-dev.karizmastudios.org";
const PROD_WEB = "https://baziness.com";

const env = process.env;
const ENABLED = env.BAZINESS_PLATFORM_ENABLED === "true";
const USE_PROD = env.BAZINESS_PLATFORM_USE_PRODUCTION === "true";
const CLIENT_ID = env.BAZINESS_CLIENT_ID ?? "";
const CLIENT_SECRET = env.BAZINESS_CLIENT_SECRET ?? "";
const PACKAGE_NAME = env.BAZINESS_PLATFORM_PACKAGE_NAME ?? "";
const trim = (u: string) => u.replace(/\/+$/, "");
export const API_BASE = trim(env.BAZINESS_PLATFORM_API_URL || (USE_PROD ? PROD_API : DEV_API));
export const WEB_BASE = trim(env.BAZINESS_PLATFORM_WEB_URL || (USE_PROD ? PROD_WEB : DEV_WEB));

export const platformEnabled = () => ENABLED && CLIENT_ID !== "" && CLIENT_SECRET !== "";

/** Public, non-secret config for the browser. */
export const platformClientConfig = () => ({ webBase: WEB_BASE, apiBase: API_BASE,
packageName: PACKAGE_NAME });

export interface PlatformUser { userId: number; token: string; isLinkedToAccount: boolean }

let bearer: string | null = null;

async function authenticate(): Promise<void> {
  const res = await fetch(`${API_BASE}/client/auth`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ clientId: CLIENT_ID, clientSecret: CLIENT_SECRET }),
  });
  if (!res.ok) throw new Error(`client/auth failed (${res.status})`);
  const json = (await res.json()) as Record<string, unknown>;
  bearer = String(json.token ?? json.Token ?? "");
  if (!bearer) throw new Error("client/auth returned no token");
}

async function platformFetch(method: "GET" | "POST", path: string, body?: unknown) {
  if (!bearer) await authenticate();
  const send = () =>
    fetch(`${API_BASE}${path}`, {
      method,
      headers: { Authorization: `Bearer ${bearer}`, ...(body ? { "Content-Type":
"application/json" } : {}) },
      body: body ? JSON.stringify(body) : undefined,
    });
  let res = await send();
  if (res.status === 401) { await authenticate(); res = await send(); } // bearer expired
  if (!res.ok) throw Object.assign(new Error(`${path} failed (${res.status})`), { status:
res.status }); // 400 from add-coin = already paid
  return (await res.json()) as Record<string, unknown>;
}

const readUser = (j: Record<string, unknown>): PlatformUser => ({

```

```

    userId: Number(j.userId ?? j.UserId ?? 0),
    token: String(j.token ?? j.Token ?? ""),
    isLinkedToAccount: Boolean(j.isLinkedToAccount ?? j.IsLinkedToAccount ?? false),
  });

export const createUser = async (deviceId: string, platform: string, store: string,
linkedToken: string | null) =>
  readUser(await platformFetch("POST", "/client/create-user", { deviceId, platform, store,
linkedToken }));

export const getUserAccessToken = async (businessUserId: number) =>
  readUser(await platformFetch("GET", `/client/get-user-token/${businessUserId}`));

export async function validateUserAccessToken(token: string): Promise<number> {
  const j = await platformFetch("GET", `/client/validate-user-
token/${encodeURIComponent(token)}`);
  if (!(j.isValid ?? j.IsValid)) throw new Error("invalid user token");
  return Number(j.userId ?? j.UserId ?? 0);
}

/** Credit Gita coins. `grantKey` makes the transaction id deterministic: a repeat answers
400 (= already paid). */
export async function addCoin(businessUserId: number, amount: number, grantKey: string,
metaData?: object) {
  const hash = createHash("sha1").update(grantKey).digest("hex").slice(0, 24);
  const j = await platformFetch("POST", "/client/add-coin", {
    userId: businessUserId, amount, transactionId: `${businessUserId}_${amount}_${hash}`,
    metaData: metaData ?? null,
  });
  return Number(j.amount ?? j.Amount ?? 0); // the player's new balance
}

```

## C# (.NET)

```

// BazinessPlatformClient.cs - all credentials stay on the server.
// Program.cs: builder.Services.AddSingleton<PlatformSettings>(...);
// builder.Services.AddHttpClient<BazinessPlatformClient>();
using System.Net;
using System.Net.Http.Headers;
using System.Net.Http.Json;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;

public record PlatformSettings(bool Enabled, bool UseProduction, long ClientId, string
ClientSecret, string PackageName)
{
  public string ApiBase => UseProduction ? "https://platform.karizmastudio.com/api" :
"https://platform-dev.karizmastudios.org/api";
  public string WebBase => UseProduction ? "https://baziness.com" : "https://platform-web-
dev.karizmastudios.org";
  public bool IsOn => Enabled && ClientId != 0 && !string.IsNullOrEmpty(ClientSecret);
}

public record PlatformUser(long UserId, string Token, bool IsLinkedToAccount);

public sealed class BazinessPlatformClient(HttpClient http, PlatformSettings settings)
{
  static readonly JsonSerializerOptions Json = new(JsonSerializerDefaults.Web); //

```

```

camelCase + case-insensitive
    static string? bearer; // the client token, shared by
every request
    static readonly SemaphoreSlim authLock = new(1, 1);

    async Task AuthenticateAsync()
    {
        await authLock.WaitAsync();
        try
        {
            var res = await http.PostAsJsonAsync($"{settings.ApiBase}/client/auth",
                new { clientId = settings.ClientId, clientSecret = settings.ClientSecret });
            res.EnsureSuccessStatusCode();
            bearer = (await res.Content.ReadFromJsonAsync<JsonElement>
                ()).GetProperty("token").GetString()
                ?? throw new InvalidOperationException("client/auth returned no token");
        }
        finally { authLock.Release(); }
    }

    async Task<JsonElement> SendAsync(HttpMethod method, string path, object? body = null)
    {
        if (bearer is null) await AuthenticateAsync();
        async Task<HttpResponseMessage> Send()
        {
            var req = new HttpRequestMessage(method, $"{settings.ApiBase}{path}");
            req.Headers.Authorization = new AuthenticationHeaderValue("Bearer", bearer);
            if (body is not null) req.Content = JsonContent.Create(body, options: Json);
            return await http.SendAsync(req);
        }
        var res = await Send();
        if (res.StatusCode == HttpStatusCode.Unauthorized) { await AuthenticateAsync(); res =
await Send(); } // token expired
        if (!res.IsSuccessStatusCode) throw new HttpRequestException($"{path} failed
({(int)res.StatusCode})", null, res.StatusCode);
        return await res.Content.ReadFromJsonAsync<JsonElement>(Json);
    }

    static PlatformUser ReadUser(JsonElement j) => new(
        j.GetProperty("userId").GetInt64(), j.GetProperty("token").GetString()!,
        j.GetProperty("isLinkedToAccount").GetBoolean());

    public async Task<PlatformUser> CreateUserAsync(string deviceId, string platform, string
store, string? linkedToken) =>
        ReadUser(await SendAsync(HttpMethod.Post, "/client/create-user", new { deviceId,
platform, store, linkedToken }));

    public async Task<PlatformUser> GetUserAccessTokenAsync(long businessUserId) =>
        ReadUser(await SendAsync(HttpMethod.Get, $"/client/get-user-
token/{businessUserId}"));

    /// <summary>The server-side gate for every token a client sends. Throws when the token
is invalid.</summary>
    public async Task<long> ValidateUserAccessTokenAsync(string token)
    {
        var j = await SendAsync(HttpMethod.Get, $"/client/validate-user-
token/{Uri.EscapeDataString(token)}");
        if (!j.GetProperty("isValid").GetBoolean()) throw new
InvalidOperationException("invalid user token");
        return j.GetProperty("userId").GetInt64();
    }

```

```

    }

    /// <summary>
    /// Credit Gita coins. The transaction id comes from grantKey (what the payout IS, e.g.
    "match:17"),
    /// so a repeat is refused with HTTP 400 – treat that 400 as "already paid".
    /// </summary>
    public async Task<long> AddCoinAsync(long baziinessUserId, long amount, string grantKey,
    object? metaData = null)
    {
        var hash = Convert.ToHexString(SHA1.HashData(Encoding.UTF8.GetBytes(grantKey)))
       [..24].ToLowerInvariant();
        var j = await SendAsync(HttpMethod.Post, "/client/add-coin",
            new { userId = baziinessUserId, amount, transactionId = $"
{baziinessUserId}_{amount}_{hash}", metaData });
        return j.GetProperty("amount").GetInt64(); // the player's new balance
    }
}

```

## Python

```

# baziiness/platform.py – all credentials stay on the server.
import hashlib
import os
import requests

DEV_API, PROD_API = "https://platform-dev.karizmastudios.org/api",
"https://platform.karizmastudio.com/api"
DEV_WEB, PROD_WEB = "https://platform-web-dev.karizmastudios.org", "https://baziiness.com"

ENABLED = os.getenv("BAZINESS_PLATFORM_ENABLED", "false").lower() == "true"
USE_PROD = os.getenv("BAZINESS_PLATFORM_USE_PRODUCTION", "false").lower() == "true"
CLIENT_ID = os.getenv("BAZINESS_CLIENT_ID", "")
CLIENT_SECRET = os.getenv("BAZINESS_CLIENT_SECRET", "")
PACKAGE_NAME = os.getenv("BAZINESS_PLATFORM_PACKAGE_NAME", "")
API_BASE = (os.getenv("BAZINESS_PLATFORM_API_URL") or (PROD_API if USE_PROD else
DEV_API)).rstrip("/")
WEB_BASE = (os.getenv("BAZINESS_PLATFORM_WEB_URL") or (PROD_WEB if USE_PROD else
DEV_WEB)).rstrip("/")

def platform_enabled() -> bool:
    return ENABLED and bool(CLIENT_ID) and bool(CLIENT_SECRET)

def platform_client_config() -> dict:
    """Public, non-secret config for the browser."""
    return {"webBase": WEB_BASE, "apiBase": API_BASE, "packageName": PACKAGE_NAME}

def _pick(j: dict, name: str, default=None):
    # The platform is .NET: accept camelCase and PascalCase.
    return j.get(name, j.get(name[0].upper() + name[1:], default))

class BaziinessPlatform:
    def __init__(self) -> None:
        self._bearer: str | None = None
        self._http = requests.Session()

```

```

def _authenticate(self) -> None:
    r = self._http.post(f"{API_BASE}/client/auth",
                        json={"clientId": int(CLIENT_ID), "clientSecret": CLIENT_SECRET},
                        timeout=10)
    r.raise_for_status()
    self._bearer = _pick(r.json(), "token")
    if not self._bearer:
        raise RuntimeError("client/auth returned no token")

def _call(self, method: str, path: str, body: dict | None = None) -> dict:
    if not self._bearer:
        self._authenticate()
    send = lambda: self._http.request(method, f"{API_BASE}{path}", json=body, timeout=10,
                                      headers={"Authorization": f"Bearer {self._bearer}"})
    r = send()
    if r.status_code == 401: # bearer expired
        self._authenticate()
        r = send()
    r.raise_for_status()
    return r.json()

@staticmethod
def _user(j: dict) -> dict:
    return {"userId": int(_pick(j, "userId", 0)), "token": str(_pick(j, "token", "")),
           "isLinkedToAccount": bool(_pick(j, "isLinkedToAccount", False))}

def create_user(self, device_id: str, platform: str, store: str, linked_token: str | None) -> dict:
    return self._user(self._call("POST", "/client/create-user", {
        "deviceId": device_id, "platform": platform, "store": store, "linkedToken": linked_token}))

def get_user_access_token(self, baziness_user_id: int) -> dict:
    return self._user(self._call("GET", f"/client/get-user-token/{baziness_user_id}"))

def validate_user_access_token(self, token: str) -> int:
    j = self._call("GET", f"/client/validate-user-token/{requests.utils.quote(token, safe='')}")
    if not _pick(j, "isValid", False):
        raise ValueError("invalid user token")
    return int(_pick(j, "userId", 0))

def add_coin(self, baziness_user_id: int, amount: int, grant_key: str, meta: dict | None = None) -> int:
    digest = hashlib.shal(grant_key.encode()).hexdigest()[0:24]
    j = self._call("POST", "/client/add-coin", {
        "userId": baziness_user_id, "amount": amount,
        "transactionId": f"{baziness_user_id}_{amount}_{digest}", "metaData": meta})
    return int(_pick(j, "amount", 0)) # the player's new balance

platform = BazinessPlatform()

```

کلید محرمانه را برنمی‌گرداند. `GET /api/platform/config` : بازیس روشن است یا نه، و صفحه‌ها و API کجا هستند. اثر جانبی ندارد و هرگز

## TypeScript (Node)

```
import express from "express";
import { platformClientConfig, platformEnabled, WEB_BASE } from "./platform/baziness";

const GAME_NAME = process.env.BAZINESS_GAME_NAME ?? "example"; // your game's name on the platform
const GAME_OFFER_URL = process.env.BAZINESS_GAME_OFFER_URL || `${WEB_BASE}/game-offer?game_name=${encodeURIComponent(GAME_NAME)}`;

const app = express();
app.use(express.json());

// Side-effect free: tells the client whether Baziness is on and where its pages live. Never the secret.
app.get("/api/platform/config", (_req, res) => {
  res.json(platformEnabled()
    ? { enabled: true, ...platformClientConfig(), gameId: GAME_NAME, gameIdUrl: GAME_OFFER_URL }
    : { enabled: false });
});
```

## C# (.NET)

```
// Side-effect free: tells the client whether Baziness is on and where its pages live. Never the secret.
app.MapGet("/api/platform/config", (PlatformSettings s, IConfiguration cfg) =>
{
  if (!s.IsOn) return Results.Ok(new { enabled = false });
  var gameId = cfg["Baziness:GameName"] ?? "example";
  var gameIdUrl = cfg["Baziness:GameOfferUrl"] is { Length: > 0 } custom
    ? custom
    : $"{s.WebBase}/game-offer?game_name={Uri.EscapeDataString(gameId)}";
  return Results.Ok(new { enabled = true, webBase = s.WebBase, apiBase = s.ApiBase,
    packageName = s.PackageName, gameId, gameIdUrl });
});
```

## Python

```
import os
from urllib.parse import quote
from fastapi import FastAPI
from baziness.platform import WEB_BASE, platform_client_config, platform_enabled

GAME_NAME = os.getenv("BAZINESS_GAME_NAME", "example") # your game's name on the platform
GAME_OFFER_URL = os.getenv("BAZINESS_GAME_OFFER_URL") or f"{WEB_BASE}/game-offer?game_name={quote(GAME_NAME)}"

app = FastAPI()

@app.get("/api/platform/config")
```

```
def platform_config():
    """Side-effect free: tells the client whether Baziness is on and where its pages live.
    Never the secret."""
    if not platform_enabled():
        return {"enabled": False}
    return {"enabled": True, **platform_client_config(), "gameName": GAME_NAME,
            "gameOfferUrl": GAME_OFFER_URL}
```

## endpointهای ورود

POST /api/platform/token و POST /api/platform/login ، همان طور که در ورود ← چیزهایی که سرور فراهم می کند آمده.

### TypeScript (Node)

```
import { createUser, getUserAccessToken, platformClientConfig, platformEnabled,
validateUserAccessToken } from "../platform/baziness";
import { db } from "../db"; // your own data layer

app.post("/api/platform/token", async (req, res) => {
    const deviceId = String(req.body.deviceId ?? "").trim();
    const linkedToken: string | null = req.body.linkedToken ?? null;
    if (!deviceId) return res.status(400).json({ error: "deviceId required" });
    if (!platformEnabled()) return res.json({ enabled: false });

    const config = { enabled: true, ...platformClientConfig() };
    // Pass first-touch attribution so a player who signs in before your normal login is still
    credited.
    const user = await db.ensureUserByDevice(deviceId, { utm: req.body.utm, tag: req.body.tag
});
    try {
        if (!user.bazinessUserId) {
            const created = await createUser(deviceId, "Web", "Direct", linkedToken);
            const owner = await db.findUserByBazinessId(created.userId);
            if (owner && owner.id !== user.id) {
                return res.json({ ...config, token: created.token, bazinessUserId: created.userId,
                    isLinkedToAccount: created.isLinkedToAccount, switchToDeviceId: owner.deviceId });
            }
            await db.linkBaziness(user.id, created.userId, created.isLinkedToAccount);
            return res.json({ ...config, token: created.token, bazinessUserId: created.userId,
isLinkedToAccount: created.isLinkedToAccount });
        }
        const fresh = await getUserAccessToken(user.bazinessUserId);
        if (fresh.isLinkedToAccount !== user.isBazinessLinked) await db.setLinked(user.id,
fresh.isLinkedToAccount);
        return res.json({ ...config, token: fresh.token, bazinessUserId: fresh.userId,
isLinkedToAccount: fresh.isLinkedToAccount });
    } catch (err) {
        console.error("[platform] token failed:", (err as Error).message);
        return res.json({ ...config, error: "platform_unavailable" }); // never block the boot
    }
});

app.post("/api/platform/login", async (req, res) => {
    const deviceId = String(req.body.deviceId ?? "").trim();
    const token = String(req.body.token ?? "").trim();
```

```

    if (!deviceId || !token) return res.status(400).json({ error: "deviceId and token required"
});
    if (!platformEnabled()) return res.json({ ok: false, reason: "disabled" });
    try {
        const bazinessUserId = await validateUserAccessToken(token); // the server-side gate
        const owner = await db.findUserByBazinessId(bazinessUserId);
        const me = await db.ensureUserByDevice(deviceId);
        if (owner && owner.id !== me.id) return res.json({ ok: true, switchToDeviceId:
owner.deviceId });
        await db.linkBaziness(me.id, bazinessUserId, true);
        return res.json({ ok: true });
    } catch {
        return res.json({ ok: false, reason: "invalid" });
    }
});

```

#### C# (.NET)

```

app.MapPost("/api/platform/token", async (TokenRequest body, BazinessPlatformClient platform,
    IUserStore users, PlatformSettings pub, ILogger<Program> log) =>
{
    if (string.IsNullOrWhiteSpace(body.DeviceId)) return Results.BadRequest(new { error =
"deviceId required" });
    if (!pub.IsOn) return Results.Ok(new { enabled = false });

    var user = await users.EnsureByDeviceAsync(body.DeviceId);
    try
    {
        PlatformUser auth;
        if (user.BazinessUserId is null)
        {
            auth = await platform.CreateUserAsync(body.DeviceId, "Web", "Direct",
body.LinkedToken);
            var owner = await users.FindByBazinessUserIdAsync(auth.UserId);
            if (owner is not null && owner.Id != user.Id)
                return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase,
pub.PackageName,
                    token = auth.Token, bazinessUserId = auth.UserId, auth.IsLinkedToAccount,
                    switchToDeviceId = owner.DeviceId });
            await users.LinkBazinessAsync(user.Id, auth.UserId, auth.IsLinkedToAccount);
        }
        else
        {
            auth = await platform.GetUserAccessTokenAsync(user.BazinessUserId.Value);
            if (auth.IsLinkedToAccount != user.IsBazinessLinked)
                await users.SetLinkedAsync(user.Id, auth.IsLinkedToAccount);
        }
        return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase, pub.PackageName,
            token = auth.Token, bazinessUserId = auth.UserId, auth.IsLinkedToAccount });
    }
    catch (Exception ex)
    {
        log.LogError(ex, "platform token failed");
        return Results.Ok(new { enabled = true, pub.WebBase, pub.ApiBase, pub.PackageName,
            error = "platform_unavailable" }); // never block the boot
    }
});

app.MapPost("/api/platform/login", async (LoginRequest body, BazinessPlatformClient platform,

```

```

IUserStore users, PlatformSettings pub) =>
{
    if (string.IsNullOrEmpty(body.DeviceId) || string.IsNullOrEmpty(body.Token))
        return Results.BadRequest(new { error = "deviceId and token required" });
    if (!pub.IsOn) return Results.Ok(new { ok = false, reason = "disabled" });
    try
    {
        long bazinessUserId = await platform.ValidateUserAccessTokenAsync(body.Token); //
throws when invalid
        var owner = await users.FindByBazinessUserIdAsync(bazinessUserId);
        var me = await users.EnsureByDeviceAsync(body.DeviceId);
        if (owner is not null && owner.Id != me.Id)
            return Results.Ok(new { ok = true, switchToDeviceId = owner.DeviceId });
        await users.LinkBazinessAsync(me.Id, bazinessUserId, isLinked: true);
        return Results.Ok(new { ok = true });
    }
    catch
    {
        return Results.Ok(new { ok = false, reason = "invalid" });
    }
});

public record TokenRequest(string DeviceId, string? LinkedToken);
public record LoginRequest(string DeviceId, string Token);

```

## Python

```

from fastapi import FastAPI
from pydantic import BaseModel
from baziness.platform import platform, platform_client_config, platform_enabled
from app import db # your own data layer

app = FastAPI()

class TokenRequest(BaseModel):
    deviceId: str
    linkedToken: str | None = None

class LoginRequest(BaseModel):
    deviceId: str
    token: str

@app.post("/api/platform/token")
def platform_token(body: TokenRequest):
    if not platform_enabled():
        return {"enabled": False}
    config = {"enabled": True, **platform_client_config()}
    user = db.ensure_user_by_device(body.deviceId)
    try:
        if not user.baziness_user_id:
            created = platform.create_user(body.deviceId, "Web", "Direct", body.linkedToken)
            owner = db.find_user_by_baziness_user_id(created["userId"])
            if owner and owner.id != user.id:
                return {**config, "token": created["token"], "bazinessUserId":
created["userId"],
                        "isLinkedToAccount": created["isLinkedToAccount"],

```

```

"switchToDeviceId": owner.device_id}
    db.link_baziness(user.id, created["userId"], created["isLinkedToAccount"])
    auth = created
  else:
    auth = platform.get_user_access_token(user.baziness_user_id)
    if auth["isLinkedToAccount"] != user.is_baziness_linked:
      db.set_linked(user.id, auth["isLinkedToAccount"])
    return {**config, "token": auth["token"], "bazinessUserId": auth["userId"],
            "isLinkedToAccount": auth["isLinkedToAccount"]}
  except Exception as exc: # platform down: never block the boot
    print("[platform] token failed:", exc)
    return {**config, "error": "platform_unavailable"}

@app.post("/api/platform/login")
def platform_login(body: LoginRequest):
  if not platform_enabled():
    return {"ok": False, "reason": "disabled"}
  try:
    baziness_user_id = platform.validate_user_access_token(body.token) # the server-side
    gate
  except Exception:
    return {"ok": False, "reason": "invalid"}
  owner = db.find_user_by_baziness_user_id(baziness_user_id)
  me = db.ensure_user_by_device(body.deviceId)
  if owner and owner.id != me.id:
    return {"ok": True, "switchToDeviceId": owner.device_id}
  db.link_baziness(me.id, baziness_user_id, True)
  return {"ok": True}

```

## دادن گیتا کوین

پرداختی که هیچ وقت دو بار انجام نمی شود: کلید پرداخت `transactionId` را ثابت می کند و 400 یعنی قبلاً پرداخت شده.

### TypeScript (Node)

```

// grants.ts - a payout that can never be paid twice.
import { addCoin } from "../platform/baziness";

export async function payTournamentReward(claimId: number, bazinessUserId: number, amount:
number) {
  try {
    // Same claim → same key → same transactionId → the platform refuses a second credit.
    await addCoin(bazinessUserId, amount, `reward_claim:${claimId}`, { source: "tournament",
claimId });
  } catch (err) {
    // 400 = this transactionId was already credited (e.g. we crashed after the platform paid
but
    // before we saved): the coins are there, so carry on instead of failing forever.
    if ((err as { status?: number }).status !== 400) throw err;
  }
  await db.claims.markPaid(claimId); // your own bookkeeping
  await db.ledger.insert({ claimId, currency: "GITA_COIN", amount, external: true });
}

```

## C# (.NET)

```
// Grants.cs - a payout that can never be paid twice.
public sealed class Grants(BazinessPlatformClient platform, AppDb db)
{
    public async Task PayTournamentRewardAsync(long claimId, long bazinessUserId, long
amount)
    {
        try
        {
            // Same claim → same key → same transactionId → the platform refuses a second
            credit.
            await platform.AddCoinAsync(bazinessUserId, amount, $"reward_claim:{claimId}",
new { source = "tournament", claimId });
        }
        catch (HttpRequestException e) when (e.StatusCode ==
System.Net.HttpStatusCode.BadRequest)
        {
            // 400 = already credited by an earlier attempt: the coins are there, carry on.
        }
        await db.Claims.MarkPaidAsync(claimId); // your own bookkeeping
        await db.Ledger.InsertAsync(new LedgerRow(claimId, "GITA_COIN", amount, External:
true));
    }
}
```

## Python

```
# grants.py - a payout that can never be paid twice.
import requests
from baziness.platform import platform

def pay_tournament_reward(claim_id: int, baziness_user_id: int, amount: int) -> None:
    try:
        # Same claim → same key → same transactionId → the platform refuses a second credit.
        platform.add_coin(baziness_user_id, amount, f"reward_claim:{claim_id}", {"source":
"tournament", "claimId": claim_id})
    except requests.HTTPError as err:
        if err.response is None or err.response.status_code != 400:
            raise
        # 400 = already credited by an earlier attempt: the coins are there, carry on.
    db.claims.mark_paid(claim_id) # your own bookkeeping
    db.ledger.insert(claim_id=claim_id, currency="GITA_COIN", amount=amount, external=True)
```

## دریافت‌ها، سکه‌های امانی و انتقالشان هنگام ورود

قانون‌های گیتا کوین ← بازیکنانی که هنوز وارد نشده‌اند: دریافت جایزه بازیکن وارد شده می‌خواهد، خرید پرداخت‌شده هیچ‌وقت شکست نمی‌خورد، و سکه‌های امانی هنگام ورود بازیکن به بازیکن منتقل می‌شوند.

## TypeScript (Node)

```
// grants.ts: Gita coins go to the platform; COIN/GEM stay local.
export async function grantCurrency(tx: Tx, change: WalletChange & { idempotencyKey: string
},
    opts: { requireLink: boolean; metaData?: object }) {
```

```

    if (change.currency !== "GITA_COIN" || !platformEnabled() || change.amount < 0) {
        return applyWalletChange(tx, change); // local wallet
    }
    const { baziness_user_id: bazinessUserId } = await tx.one("select baziness_user_id from
users where id = $1", [change.userId]);
    if (!bazinessUserId) {
        if (opts.requireLink) throw new ApiError("PLATFORM_NOT_LINKED", "برای دریافت گیتا کوین
وارد حساب بازیس شوید");
        return applyWalletChange(tx, { ...change, note: "held until a Baziness account is linked"
}); // escrow
    }
    try {
        await addCoin(bazinessUserId, change.amount, change.idempotencyKey, { source:
change.type, ...opts.metaData });
    } catch (err) {
        throw new ApiError("PLATFORM_UNAVAILABLE", "ارتباط با بازیس برقرار نشد، دوباره تلاش کنید");
    }
    await recordExternalCredit(tx, change); // ledger row, balance
    unchanged locally
}

/** Called right after a successful link (POST /api/platform/login or /token). */
export async function flushGitaEscrow(userId: number) {
    await withTx(async (tx) => {
        const user = await tx.one("select baziness_user_id from users where id = $1", [userId]);
        const wallet = await tx.oneOrNull(
            "select balance from user_wallets where user_id = $1 and currency = 'GITA_COIN' for
update", [userId]);
        if (!user.baziness_user_id || !wallet || wallet.balance <= 0) return;
        const last = await tx.one("select max(id) as id from wallet_transactions where user_id =
$1 and currency = 'GITA_COIN'", [userId]);
        const key = `escrow:${userId}:${last.id ?? 0}`;
        await addCoin(user.baziness_user_id, wallet.balance, key, { source: "escrow" });
        await applyWalletChange(tx, { userId, currency: "GITA_COIN", amount: -wallet.balance,
            type: "REWARD_SELECTION", idempotencyKey: key, note: "moved to the Baziness account"
        });
    });
}

```

#### C# (.NET)

```

// Inside a service with: BazinessPlatformClient platform, PlatformSettings settings,
IUserStore users, IWalletStore wallets.
public async Task GrantAsync(DbTransaction tx, WalletChange change, bool requireLink)
{
    if (change.Currency != "GITA_COIN" || !settings.IsOn || change.Amount < 0)
    { await wallets.ApplyAsync(tx, change); return; } // local wallet

    long? bazinessUserId = await users.GetBazinessUserIdAsync(tx, change.UserId);
    if (bazinessUserId is null)
    {
        if (requireLink) throw new ApiException("PLATFORM_NOT_LINKED", "برای دریافت گیتا کوین
وارد حساب بازیس شوید");
        await wallets.ApplyAsync(tx, change with { Note = "held until a Baziness account is
linked" }); // escrow
        return;
    }
    await platform.AddCoinAsync(bazinessUserId.Value, change.Amount, change.IdempotencyKey,
new { source = change.Type });
}

```

```

        await wallets.RecordExternalCreditAsync(tx, change);
    }

    public async Task FlushEscrowAsync(long userId)
    {
        await using var tx = await db.BeginTransactionAsync();
        var bazinessUserId = await users.GetBazinessUserIdAsync(tx, userId);
        var balance = await wallets.LockBalanceAsync(tx, userId, "GITA_COIN"); // SELECT ... FOR
        UPDATE
        if (bazinessUserId is null || balance <= 0) return;
        var key = $"escrow:{userId}:{await wallets.LastTransactionIdAsync(tx, userId,
"GITA_COIN")}";
        await platform.AddCoinAsync(bazinessUserId.Value, balance, key, new { source = "escrow"
});
        await wallets.ApplyAsync(tx, new WalletChange(userId, "GITA_COIN", -balance,
"REWARD_SELECTION", key));
        await tx.CommitAsync();
    }
}

```

## Python

```

def grant_currency(tx, change: dict, require_link: bool, meta: dict | None = None) -> None:
    if change["currency"] != "GITA_COIN" or not platform_enabled() or change["amount"] < 0:
        return apply_wallet_change(tx, change) # local wallet
    baziness_user_id = tx.scalar("select baziness_user_id from users where id = %s",
[change["userId"]])
    if not baziness_user_id:
        if require_link:
            raise ApiError("PLATFORM_NOT_LINKED", "برای دریافت گیتا کوین وارد حساب بازینس
شود")
        return apply_wallet_change(tx, {**change, "note": "held until a Baziness account is
linked"}) # escrow
    platform.add_coin(baziness_user_id, change["amount"], change["idempotencyKey"],
{"source": change["type"], **(meta or {})})
    record_external_credit(tx, change)

def flush_gita_escrow(user_id: int) -> None:
    with db.transaction() as tx:
        baziness_user_id = tx.scalar("select baziness_user_id from users where id = %s",
[user_id])
        balance = tx.scalar("select balance from user_wallets where user_id = %s and currency
= 'GITA_COIN' "
                             "for update", [user_id]) or 0
        if not baziness_user_id or balance <= 0:
            return
        last = tx.scalar("select max(id) from wallet_transactions where user_id = %s and
currency = 'GITA_COIN'",
                         [user_id]) or 0
        key = f"escrow:{user_id}:{last}"
        platform.add_coin(baziness_user_id, balance, key, {"source": "escrow"})
        apply_wallet_change(tx, {"userId": user_id, "currency": "GITA_COIN", "amount": -
balance,
                                "type": "REWARD_SELECTION", "idempotencyKey": key})

```

قدم‌های ۱ تا ۵ پرداخت بازینس: آدرس صفحه‌ی پرداخت، `get_payment`، `verify_payment` (بدنه‌ی فرم) و `.consume`

## TypeScript (Node)

```
// platform/baziness-pay.ts
const BASE = (process.env.BAZINESS_PAY_BASE_URL ??
"https://pay.baziness.com").replace(/\/+$/, "");
const PNAME = process.env.BAZINESS_PAY_PACKAGE_NAME ?? "";
const PREFIX = process.env.BAZINESS_SKU_PREFIX ?? "";
export const payEnabled = () => process.env.BAZINESS_PAY_ENABLED === "true";

export const skuFor = (productSku: string, override?: string | null) => override ||
`${PREFIX}${productSku}`;

export const buildBuyUrl = (uid: string, sku: string) =>
`${BASE}/buy?${new URLSearchParams({ sku, uid, pname: PNAME })}`;

/** The verification token of a completed, unconsumed payment – or null while unpaid. */
export async function getPayment(uid: string, sku: string): Promise<{ token: string; sku:
string } | null> {
  try {
    const res = await fetch(`${BASE}/get_payment`, {
      method: "POST", headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ uid, sku, pname: PNAME }),
    });
    if (!res.ok) return null;
    const j = (await res.json()) as { success?: boolean; token?: string; sku?: string };
    return j.success && j.token ? { token: j.token, sku: j.sku ?? sku } : null;
  } catch { return null; }
}

/** The authoritative check. Never trust a token from the client. */
export async function verifyPayment(uid: string, token: string, sku: string):
Promise<boolean> {
  try {
    const res = await fetch(`${BASE}/verify_payment`, {
      method: "POST", headers: { "Content-Type": "application/x-www-form-urlencoded" }, //
form, not JSON
      body: new URLSearchParams({ uid, token, sku, pname: PNAME }).toString(),
    });
    if (!res.ok) return false;
    const j = (await res.json()) as { success?: boolean; error?: string };
    if (!j.success && j.error) console.warn("[pay] verify_payment rejected:", j.error);
    return j.success === true;
  } catch { return false; }
}

/** Let the SKU be bought again. Best effort, 3 attempts; never blocks an already-granted
purchase. */
export async function consumePayment(uid: string, sku: string): Promise<boolean> {
  for (let attempt = 1; attempt <= 3; attempt++) {
    try {
      const res = await fetch(`${BASE}/consume`, {
        method: "POST", headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ uid, sku, pname: PNAME }),
      });
    }
  }
}
```

```

        if (res.ok && ((await res.json()) as { success?: boolean }).success) return true;
    } catch { /* retry */ }
}
console.error("[pay] consume failed uid=%s sku=%s", uid, sku);
return false;
}

```

## C# (.NET)

```

// BazinessPay.cs - register with builder.Services.AddHttpClient<BazinessPay>();
using System.Net.Http.Json;

public sealed class BazinessPay(HttpClient http, IConfiguration cfg, ILogger<BazinessPay>
log)
{
    string Base => (cfg["BazinessPay:BaseUrl"] ?? "https://pay.baziness.com").TrimEnd('/');
    string PName => cfg["BazinessPay:PackageName"]!;
    public bool Enabled => cfg.GetValue<bool>("BazinessPay:Enabled");
    public string SkuFor(string productSku, string? overrideSku = null)
        => string.IsNullOrEmpty(overrideSku) ? $"{cfg["BazinessPay:SkuPrefix"]}{productSku}"
: overrideSku;

    public string BuildBuyUrl(string uid, string sku) =>
        $"{Base}/buy?sku={Uri.EscapeDataString(sku)}&uid={Uri.EscapeDataString(uid)}&pname=
{Uri.EscapeDataString(PName)}";

    record PaymentDto(bool Success, string? Token, string? Sku);
    record ResultDto(bool Success, string? Error);

    public async Task<(string Token, string Sku)?> GetPaymentAsync(string uid, string sku)
    {
        try
        {
            var res = await http.PostAsJsonAsync($"{Base}/get_payment", new { uid, sku, pname
= PName });
            if (!res.IsSuccessStatusCode) return null;
            var j = await res.Content.ReadFromJsonAsync<PaymentDto>();
            return j is { Success: true, Token: { Length: > 0 } t } ? (t, j.Sku ?? sku) :
null;
        }
        catch (Exception ex) { log.LogError(ex, "get_payment failed"); return null; }
    }

    public async Task<bool> VerifyPaymentAsync(string uid, string token, string sku)
    {
        try
        {
            var form = new FormUrlEncodedContent(new Dictionary<string, string> // form,
not JSON
            { ["uid"] = uid, ["token"] = token, ["sku"] = sku, ["pname"] = PName });
            var res = await http.PostAsync($"{Base}/verify_payment", form);
            if (!res.IsSuccessStatusCode) return false;
            var j = await res.Content.ReadFromJsonAsync<ResultDto>();
            if (j is { Success: false, Error: not null }) log.LogWarning("verify_payment
rejected: {Error}", j.Error);
            return j?.Success == true;
        }
        catch (Exception ex) { log.LogError(ex, "verify_payment failed"); return false; }
    }
}

```

```

public async Task<bool> ConsumeAsync(string uid, string sku)
{
    for (var attempt = 1; attempt <= 3; attempt++)
    {
        try
        {
            var res = await http.PostAsJsonAsync($"{Base}/consume", new { uid, sku, pname = PName });
            if (res.IsSuccessStatusCode && (await res.Content.ReadFromJsonAsync<ResultDto>())?.Success == true) return true;
        }
        catch (Exception ex) { log.LogWarning(ex, "consume attempt {Attempt} failed", attempt); }
    }
    log.LogError("consume failed uid={Uid} sku={Sku}", uid, sku);
    return false;
}
}

```

## Python

```

# baziness/pay.py
import os
import requests
from urllib.parse import urlencode

BASE = os.getenv("BAZINESS_PAY_BASE_URL", "https://pay.baziness.com").rstrip("/")
PNAME = os.getenv("BAZINESS_PAY_PACKAGE_NAME", "")
PREFIX = os.getenv("BAZINESS_SKU_PREFIX", "")

def pay_enabled() -> bool:
    return os.getenv("BAZINESS_PAY_ENABLED", "false").lower() == "true"

def sku_for(product_sku: str, override: str | None = None) -> str:
    return override or f"{PREFIX}{product_sku}"

def build_buy_url(uid: str, sku: str) -> str:
    return f"{BASE}/buy?{urlencode({'sku': sku, 'uid': uid, 'pname': PNAME})}"

def get_payment(uid: str, sku: str) -> dict | None:
    try:
        r = requests.post(f"{BASE}/get_payment", json={"uid": uid, "sku": sku, "pname": PNAME}, timeout=10)
        j = r.json() if r.ok else {}
        return {"token": j["token"], "sku": j.get("sku", sku)} if j.get("success") and j.get("token") else None
    except requests.RequestException:
        return None

def verify_payment(uid: str, token: str, sku: str) -> bool:
    try: # form-encoded, not JSON
        r = requests.post(f"{BASE}/verify_payment", data={"uid": uid, "token": token, "sku": sku, "pname": PNAME},

```

```

        timeout=10)
    return r.ok and r.json().get("success") is True
except requests.RequestException:
    return False

def consume_payment(uid: str, sku: str) -> bool:
    for _ in range(3):
        try:
            r = requests.post(f"{BASE}/consume", json={"uid": uid, "sku": sku, "pname":
PNAME}, timeout=10)
            if r.ok and r.json().get("success"):
                return True
        except requests.RequestException:
            pass
    return False

```

## جدول پرداخت‌ها

Schema (PostgreSQL)

```

create table payments (
    id                bigserial primary key,
    user_id           bigint not null references users(id),
    shop_product_id   bigint not null references shop_products(id),
    provider           text not null default 'BAZINESS',
    provider_sku       text not null,
    -- The Baziness payment token; unique so the same payment can never be fulfilled twice.
    provider_transaction_id text,
    amount            bigint not null,
    currency_code      text not null,
    status            text not null default 'CREATED' check (status in
('CREATED', 'REDIRECTED', 'PAID', 'VERIFIED', 'FAILED', 'CANCELLED', 'FULFILLED')),
    reward_snapshot    jsonb not null,          -- what the player was promised when they
clicked
    failure_reason     text,
    created_at         timestamptz not null default now(),
    redirected_at      timestamptz,
    verified_at        timestamptz,
    fulfilled_at       timestamptz,
    updated_at         timestamptz not null default now(),
    unique (provider, provider_transaction_id)
);
create index payments_user on payments(user_id, created_at desc);

```

## endpointهای خرید

تأیید کنید و آیتم را دقیقاً یک بار بدهید. `POST /api/shop/purchase/start` و `POST /api/shop/purchase/verify` : سفارش را ذخیره کنید، بعد

TypeScript (Node)

```

// services/shop.ts
export async function startPurchase(userId: number, productId: number) {
  if (!payEnabled()) throw new ApiError("PAYMENT_DISABLED", "پرداخت به‌زودی فعال می‌شود");
  const product = await db.one("select * from shop_products where id = $1 and enabled",
    [productId]);
  const sku = skuFor(product.sku, product.baziness_product_id);
  const payment = await db.one(
    `insert into payments (user_id, shop_product_id, provider_sku, amount, currency_code,
    status, reward_snapshot, redirected_at)
    values ($1, $2, $3, $4, $5, 'REDIRECTED', $6, now()) returning id`,
    [userId, product.id, sku, product.price, product.currency_code,
    JSON.stringify(product.grants(product))]);
  return { paymentId: String(payment.id), url: buildBuyUrl(String(userId), sku) };
}

export async function verifyPurchase(userId: number, paymentId: number) {
  const p = await db.oneOrNull("select * from payments where id = $1", [paymentId]);
  if (!p || p.user_id !== userId) throw notFound("payment");
  if (p.status === "FULFILLED" || p.status === "FAILED" || p.status === "CANCELLED") {
    return { status: p.status, wallets: await getBalances(userId), granted: [] };
  }
  const uid = String(userId);
  const found = await getPayment(uid, p.provider_sku);
  if (!found) return { status: p.status, wallets: await getBalances(userId), granted: [] };
  // not paid yet
  if (!(await verifyPayment(uid, found.token, p.provider_sku))) {
    await db.none("update payments set failure_reason = 'verify_payment rejected' where id =
    $1 and status <> 'FULFILLED'", [paymentId]);
    throw new ApiError("PAYMENT_FAILED", "پرداخت تأیید نشد");
  }
  const granted = await fulfilPayment(paymentId, found.token);
  void consumePayment(uid, p.provider_sku); // after commit, best effort
  return { status: "FULFILLED", wallets: await getBalances(userId), granted };
}

export function fulfilPayment(paymentId: number, providerToken: string) {
  return withTx(async (tx) => {
    const p = await tx.one("select * from payments where id = $1 for update", [paymentId]);
    if (p.status === "FULFILLED") return [];
    const dup = await tx.oneOrNull(
      "select id from payments where provider = 'BAZINESS' and provider_transaction_id = $1
      and id <> $2", [providerToken, paymentId]);
    if (dup) {
      await tx.none("update payments set status = 'FAILED', failure_reason = 'duplicate
      provider token' where id = $1", [paymentId]);
      return [];
    }
    await tx.none("update payments set status = 'VERIFIED', provider_transaction_id = $2,
    verified_at = now() where id = $1",
    [paymentId, providerToken]);
    for (const g of p.reward_snapshot) {
      await grantCurrency(tx, { user_id: p.user_id, currency: g.currency, amount: g.amount,
      type: "SHOP_PURCHASE",
      idempotencyKey: `payment:${paymentId}:${g.currency}` }, { requireLink: false }); //
      Gita coins: escrow if not linked
    }
    await tx.none("update payments set status = 'FULFILLED', fulfilled_at = now() where id =
    $1", [paymentId]);
  });
}

```

```

    return p.reward_snapshot;
});
}

```

## C# (.NET)

```

app.MapPost("/api/shop/purchase/start", async (StartRequest body, ClaimsPrincipal me,
BazinessPay pay, ShopDb db) =>
{
    if (!pay.Enabled) return Results.Ok(new { error = "PAYMENT_DISABLED" });
    var userId = me.UserId();
    var product = await db.GetEnabledProductAsync(body.ProductId);
    if (product is null) return Results.NotFound();
    var sku = pay.SkuFor(product.Sku, product.BazinessProductId);
    var paymentId = await db.InsertPaymentAsync(userId, product, sku, status: "REDIRECTED");
    return Results.Ok(new { paymentId = paymentId.ToString(), url =
pay.BuildBuyUrl(userId.ToString(), sku) });
});

app.MapPost("/api/shop/purchase/verify", async (VerifyRequest body, ClaimsPrincipal me,
BazinessPay pay, ShopDb db) =>
{
    var userId = me.UserId();
    var p = await db.GetPaymentAsync(body.PaymentId);
    if (p is null || p.UserId != userId) return Results.NotFound();
    if (p.Status is "FULFILLED" or "FAILED" or "CANCELLED")
        return Results.Ok(new { status = p.Status, wallets = await db.BalancesAsync(userId),
granted = Array.Empty<object>() });

    var uid = userId.ToString();
    var found = await pay.GetPaymentAsync(uid, p.ProviderSku);
    if (found is null) // not paid yet - keep
waiting
        return Results.Ok(new { status = p.Status, wallets = await db.BalancesAsync(userId),
granted = Array.Empty<object>() });
    if (!await pay.VerifyPaymentAsync(uid, found.Value.Token, p.ProviderSku))
    {
        await db.SetFailureReasonAsync(p.Id, "verify_payment_rejected");
        return Results.Ok(new { error = "PAYMENT_FAILED" });
    }

    var granted = await db.FulfilAsync(p.Id, found.Value.Token); // one tx: FOR UPDATE,
dup-token check, grants, FULFILLED
    _ = pay.ConsumeAsync(uid, p.ProviderSku); // after commit, best
effort
    return Results.Ok(new { status = "FULFILLED", wallets = await db.BalancesAsync(userId),
granted });
});

public record StartRequest(long ProductId);
public record VerifyRequest(long PaymentId);

```

## Python

```

from fastapi import Depends, HTTPException
from baziness import pay

@app.post("/api/shop/purchase/start")

```

```

def start_purchase(body: dict, user=Depends(current_user)):
    if not pay.pay_enabled():
        raise HTTPException(400, "PAYMENT_DISABLED")
    product = db.fetch_one("select * from shop_products where id = %s and enabled",
[body["productId"]])
    if not product:
        raise HTTPException(404)
    sku = pay.sku_for(product["sku"], product["business_product_id"])
    payment_id = db.scalar(
        """insert into payments (user_id, shop_product_id, provider_sku, amount,
currency_code, status,
                                reward_snapshot, redirected_at)
        values (%s, %s, %s, %s, %s, 'REDIRECTED', %s, now()) returning id""",
        [user.id, product["id"], sku, product["price"], product["currency_code"],
json.dumps(product_grants(product))])
    return {"paymentId": str(payment_id), "url": pay.build_buy_url(str(user.id), sku)}

@app.post("/api/shop/purchase/verify")
def verify_purchase(body: dict, user=Depends(current_user)):
    p = db.fetch_one("select * from payments where id = %s", [body["paymentId"]])
    if not p or p["user_id"] != user.id:
        raise HTTPException(404)
    if p["status"] in ("FULFILLED", "FAILED", "CANCELLED"):
        return {"status": p["status"], "wallets": balances(user.id), "granted": []}
    uid = str(user.id)
    found = pay.get_payment(uid, p["provider_sku"])
    if not found: # not paid yet
        return {"status": p["status"], "wallets": balances(user.id), "granted": []}
    if not pay.verify_payment(uid, found["token"], p["provider_sku"]):
        db.execute("update payments set failure_reason = 'verify_payment rejected' where id =
%s "
                    "and status <> 'FULFILLED'", [p["id"]])
        raise HTTPException(400, "PAYMENT_FAILED")
    granted = fulfil_payment(p["id"], found["token"]) # one tx: FOR UPDATE, dup-token
    check, grants, FULFILLED
    threading.Thread(target=pay.consume_payment, args=(uid, p["provider_sku"]),
daemon=True).start()
    return {"status": "FULFILLED", "wallets": balances(user.id), "granted": granted}

```

## درگاه آزمایشی محلی

بازی نمونه یک جایگزین برای پرداخت بازی‌س دارد که همان پروتکل را حرف می‌زند، برای کار بدون دسترسی به درگاه. با این تنظیمات روشن می‌شود، و هرگز در محیط اصلی:

.env (local only)

```

DEV_FAKE_BAZINESS_PAY=true
BAZINESS_PAY_ENABLED=true
BAZINESS_PAY_BASE_URL=http://localhost:3390/dev/baziness

```

قانون‌های لینک‌ها و UTM. زبانه‌ی اول سمت کلاینت است (خواندن برچسب‌ها و پاک کردنشان از آدرس)؛ بقیه آن‌ها را هنگام ساخت بازیکن در سرور ذخیره می‌کنند.

### TypeScript (client)

```
// api/client.ts - the FIRST module the app imports.
const UTM_KEY = "mygame_utm";
const TAG_KEY = "mygame_tag";
const UTM_FIELDS = ["source", "medium", "campaign", "content"] as const;
type Utm = Partial<Record<(typeof UTM_FIELDS)[number], string>>;

const clean = (raw: unknown): string | null => {
  if (typeof raw !== "string") return null;
  const v = raw.trim().toLowerCase().replace(/\\s+/g, "_").slice(0, 64);
  return /^[a-z0-9][a-z0-9._-]*$/i.test(v) ? v : null; // ASCII only
};

export const FIRST_TOUCH_UTM: Utm = (() => {
  let stored: Utm = {};
  try { stored = JSON.parse(localStorage.getItem(UTM_KEY) ?? "{}"); } catch {}
  try {
    const query = new URLSearchParams(location.search);
    const keys = [...query.keys()].filter((k) => k.toLowerCase().startsWith("utm_"));
    const fromUrl: Utm = {};
    for (const k of keys) {
      const field = k.toLowerCase().slice(4) as keyof Utm;
      const v = clean(query.get(k));
      if (UTM_FIELDS.includes(field as never) && v) fromUrl[field] = v;
    }
    if (keys.length) { // take them out of the address
      keys.forEach((k) => query.delete(k));
      const search = query.toString();
      history.replaceState(history.state, "", `${location.pathname}${search ? `?${search}` : ''}`);
    }
    if (Object.keys(fromUrl).length && !Object.keys(stored).length) { // first touch wins
      stored = fromUrl;
      localStorage.setItem(UTM_KEY, JSON.stringify(stored));
    }
  } catch { /* storage blocked: the visit is simply untagged */ }
  return stored;
})();

export function firstTouchTag(): string | null {
  try {
    const fromUrl = clean(new URLSearchParams(location.search).get("tag"));
    if (fromUrl && !localStorage.getItem(TAG_KEY)) localStorage.setItem(TAG_KEY, fromUrl);
    return localStorage.getItem(TAG_KEY) ?? fromUrl;
  } catch { return null; }
}

// Sent with the device login AND with /api/platform/token:
// { deviceId, tag: firstTouchTag(), utm: FIRST_TOUCH_UTM }
```

### C# (server)

```

using System.Text.RegularExpressions;

// Save attribution ONLY when the account is created (first touch). Four separate columns.
public async Task<User> EnsureUserByDeviceAsync(string deviceId, string? tag, UtmParams? utm)
{
    const string sql = """
        insert into users (device_id, acq_tag, utm_source, utm_medium, utm_campaign,
utm_content)
        values (@deviceId, @tag, @source, @medium, @campaign, @content)
        on conflict (device_id) do update set last_seen = now() -- existing account:
attribution untouched
        returning *;
        """;
    return await db.QuerySingleAsync<User>(sql, new {
        deviceId, tag = Utm.Clean(tag),
        source = Utm.Clean(utm?.Source), medium = Utm.Clean(utm?.Medium),
        campaign = Utm.Clean(utm?.Campaign), content = Utm.Clean(utm?.Content),
    });
}

public static class Utm
{
    static readonly Regex Valid = new("^[a-z0-9][a-z0-9._-]*$", RegexOptions.Compiled);
    public static string? Clean(string? raw)
    {
        if (raw is null) return null;
        var v = Regex.Replace(raw.Trim().ToLowerInvariant(), @"\s+", "_");
        if (v.Length > 64) v = v[..64];
        return Valid.IsMatch(v) ? v : null;
    }
}

public record UtmParams(string? Source, string? Medium, string? Campaign, string? Content);

```

#### Python (server)

```

import re

_VALID = re.compile(r"^[a-z0-9][a-z0-9._-]*$")

def clean_utm(raw: str | None) -> str | None:
    if not isinstance(raw, str):
        return None
    v = re.sub(r"\s+", "_", raw.strip().lower())[:64]
    return v if _VALID.match(v) else None

def ensure_user_by_device(device_id: str, tag: str | None, utm: dict | None):
    """Attribution is saved ONLY when the account is created (first touch)."""
    utm = utm or {}
    return db.fetch_one(
        """
        insert into users (device_id, acq_tag, utm_source, utm_medium, utm_campaign,
utm_content)
        values (%s, %s, %s, %s, %s, %s)
        on conflict (device_id) do update set last_seen = now()
        returning *
        """,

```

```
[device_id, clean_utm(tag), *(clean_utm(utm.get(k) or utm.get(f"utm_{k}"))
    for k in ("source", "medium", "campaign", "content"))],
)
```

#### Schema (PostgreSQL)

```
alter table users
add column if not exists acq_tag      text,
add column if not exists utm_source  text,
add column if not exists utm_medium  text,
add column if not exists utm_campaign text,
add column if not exists utm_content text;

create index if not exists idx_users_utm_combo on users (utm_source, utm_medium,
utm_campaign, utm_content)
where utm_source is not null or utm_medium is not null or utm_campaign is not null or
utm_content is not null;
```

## ساختن لینک‌ها

لینک صفحه‌های بازی‌نِس (پارامترها):

#### TypeScript

```
type PageOptions = { redirect?: "shop" | "profile"; allowGuest?: boolean; gameId?: string;
buildStore?: string };

export function bazinessPageUrl(webBase: string, path: string, packageName: string, token:
string | null, o: PageOptions = {}) {
    const u = new URL(webBase.replace(/\/+$/, "") + (path || "/"));
    if (o.gameId) u.searchParams.set("game_id", o.gameId);
    u.searchParams.set("packageName", packageName);
    u.searchParams.set("sdkType", "js"); // never optional
    if (token) u.searchParams.set("linkedToken", token);
    if (o.redirect) u.searchParams.set("redirect", o.redirect);
    if (o.allowGuest === false) u.searchParams.set("allowGuest", "false");
    if (o.buildStore) u.searchParams.set("buildStore", o.buildStore);
    return u.toString();
}

// bazinessPageUrl(webBase, "/shop", pkg, token, { gameId: "example", redirect: "shop" })
// bazinessPageUrl(webBase, "/", pkg, token, { allowGuest: false })
```

#### C#

```
using System.Web; // or Microsoft.AspNetCore.WebUtilities.QueryHelpers

public static string BazinessPageUrl(string webBase, string path, string packageName, string?
token,
    string? redirect = null, bool allowGuest = true, string? gameId = null, string?
buildStore = null)
{
    var q = HttpUtility.ParseQueryString(string.Empty); // encodes values for you
    if (gameId != null) q["game_id"] = gameId;
```

```

q["packageName"] = packageName;
q["sdkType"] = "js"; // never optional
if (!string.IsNullOrEmpty(token)) q["linkedToken"] = token;
if (redirect != null) q["redirect"] = redirect;
if (!allowGuest) q["allowGuest"] = "false";
if (buildStore != null) q["buildStore"] = buildStore;
return $"{webBase.TrimEnd('/')}{(string.IsNullOrEmpty(path) ? "/" : path)}{q}";
}

// BazinessPageUrl(webBase, "/shop", pkg, token, redirect: "shop", gameName: "example")

```

## Python

```

from urllib.parse import urlencode

def baziness_page_url(web_base: str, path: str, package_name: str, token: str | None, *,
                      redirect: str | None = None, allow_guest: bool = True,
                      game_name: str | None = None, build_store: str | None = None) -> str:
    params: dict[str, str] = {}
    if game_name:
        params["game_name"] = game_name
    params["packageName"] = package_name
    params["sdkType"] = "js" # never optional
    if token:
        params["linkedToken"] = token
    if redirect:
        params["redirect"] = redirect
    if not allow_guest:
        params["allowGuest"] = "false"
    if build_store:
        params["buildStore"] = build_store
    return f"{web_base.rstrip('/')}{path or '/'}?{urlencode(params)}"

# baziness_page_url(web_base, "/shop", pkg, token, game_name="example", redirect="shop")

```

لینک کمپین برای بازاریابی:

## TypeScript

```

export function campaignLink(gameUrl: string, utm: { source: string; medium: string;
campaign: string; content?: string }) {
    const u = new URL(gameUrl);
    u.searchParams.set("utm_source", utm.source);
    u.searchParams.set("utm_medium", utm.medium);
    u.searchParams.set("utm_campaign", utm.campaign);
    if (utm.content) u.searchParams.set("utm_content", utm.content);
    return u.toString();
}

// campaignLink("https://yourgame.example.com/", { source: "baziness", medium: "inapp",
campaign: "shahrivar_1405", content: "banner_home" })

```

## C#

```

public static string CampaignLink(string gameUrl, string source, string medium, string
campaign, string? content = null)
{

```

```

var q = System.Web.HttpUtility.ParseQueryString(string.Empty);
q["utm_source"] = source; q["utm_medium"] = medium; q["utm_campaign"] = campaign;
if (content != null) q["utm_content"] = content;
return $"{gameUrl.TrimEnd('/')}/?{q}";
}

```

## Python

```

from urllib.parse import urlencode

def campaign_link(game_url: str, source: str, medium: str, campaign: str, content: str | None = None) -> str:
    params = {"utm_source": source, "utm_medium": medium, "utm_campaign": campaign}
    if content:
        params["utm_content"] = content
    return f"{game_url.rstrip('/')}/?{urlencode(params)}"

```

# کد کلاینت

کد آماده‌ی کلاینت (کنترلر پلتفرم، پیام‌های صفحه، موجودی سکه، دکمه‌ی جایزه، صفحه‌ی پیشنهاد آموزش و پنجره‌ی خرید) برای وب، Cocos Creator، React، و Unity.

این نمونه‌ها سمت کلاینت [راهنماها](#) هستند. بیشترشان زبانه‌های **TypeScript (وب)**، **React** و **C# (Unity)** دارند، و در انتها یک نسخه‌ی کامل **Cocos Creator** هست. وب‌ویوی داخل بازی (شیت پایین) [صفحه‌ی خودش](#) را دارد.

## کنترلر پلتفرم

قلب کلاینت: توکن بازیکن را از سرور شما می‌گیرد، لینک همه‌ی صفحه‌ها را می‌سازد، صفحه‌ها را باز می‌کند و انتخاب اجرای اول را در اسپلش انجام می‌دهد ([ورود](#)).

### TypeScript (web)

```
// baziness.ts - the client controller.
const TOKEN_KEY = "mygame_baziness_token";
const CHOICE_KEY = "mygame_platform_choice";

export class BazinessPlatform {
  enabled = false;
  isLinked = false;
  private token: string | null = null;
  private webBase = "";
  private apiBase = "";
  private packageName = "";
  private loginGate: ((ok: boolean) => void) | null = null;
  private loginSucceeded = false;
  onChange: (() => void) | null = null;

  constructor(private sheet: BazinessSheet, private deviceId: string) {
    sheet.onMessage = (m) => this.handleMessage(m);
    sheet.onClosed = () => this.handleClosed();
  }

  get choiceMade() { return localStorage.getItem(CHOICE_KEY) === "1"; }
  markChoiceMade() { localStorage.setItem(CHOICE_KEY, "1"); }

  async loadConfig(): Promise<boolean> {
    try {
      const cfg = await fetch("/api/platform/config").then((r) => r.json());
      this.enabled = cfg.enabled;
      if (cfg.enabled) this.apply(cfg);
    } catch { this.enabled = false; }
    return this.enabled;
  }

  /** Resolve the token. Returns false when the page is about to reload into another account.
  */
  async init(): Promise<boolean> {
    // The Baziness app opens games with ?linkedToken=... - forward it once, then it is
```

*stripped on reload.*

```
const linkedToken = new URLSearchParams(location.search).get("linkedToken") ||
localStorage.getItem(TOKEN_KEY);
const resp = await fetch("/api/platform/token", {
  method: "POST", headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ deviceId: this.deviceId, linkedToken }),
}).then((r) => r.json()).catch(() => null);
if (!resp?.enabled) { this.enabled = false; return true; }
this.enabled = true;
this.apply(resp);
if (resp.switchToDeviceId && resp.switchToDeviceId !== this.deviceId) {
  this.switchAccount(resp.switchToDeviceId);
  return false;
}
if (resp.token) {
  this.token = resp.token;
  this.isLinked = !!resp.isLinkedToAccount;
  localStorage.setItem(TOKEN_KEY, resp.token);
}
this.onChange?.();
return true;
}
```

*/\*\* Splash "Sign in with Baziness": resolves true once a login/link succeeded, false if closed. \*/*

```
async openLoginAndWait(): Promise<boolean> {
  if (!this.token) await this.init(); // the anonymous user's token becomes linkedToken
  if (!this.enabled || !this.webBase) return false;
  return new Promise((resolve) => {
    this.loginGate = resolve;
    this.loginSucceeded = false;
    this.sheet.open(this.pageUrl("/", { allowGuest: false }));
  });
}
```

*/\*\* Settings → "Connect your account": the profile page. \*/*

```
openAccount() { this.sheet.open(this.pageUrl("/profile", { redirect: "profile", allowGuest:
false })); }
```

```
pageUrl(path: string, o: { redirect?: string; allowGuest?: boolean; extra?: Record<string,
string> } = {}) {
```

```
  const u = new URL(this.webBase + path);
  for (const [k, v] of Object.entries(o.extra ?? {})) u.searchParams.set(k, v);
  u.searchParams.set("packageName", this.packageName);
  u.searchParams.set("sdkType", "js");
  if (this.token) u.searchParams.set("linkedToken", this.token);
  if (o.redirect) u.searchParams.set("redirect", o.redirect);
  if (o.allowGuest === false) u.searchParams.set("allowGuest", "false");
  return u.toString();
}
```

```
private apply(c: { webBase?: string; apiBase?: string; packageName?: string }) {
  this.webBase = c.webBase ?? ""; this.apiBase = c.apiBase ?? ""; this.packageName =
c.packageName ?? "";
}
```

*// handleMessage / handleClosed / switchAccount: see "Messages from the Baziness page" below.*

```
}
```

## React

```
// Splash.tsx - the first-time choice.
export function Splash({ onReady }: { onReady: (me: Me) => void }) {
  const [needsChoice, setNeedsChoice] = useState(false);
  const choice = useRef<() => void | null>(null);
  const loggedIn = useRef(false);

  useEffect(() => {
    (async () => {
      let me = (await authenticate()).me; // 1. your own account first
      if (await baziness.loadConfig()) { // 2. platform on?
        if (!baziness.choiceMade && !me.user.isBazinessLinked) {
          setNeedsChoice(true); // 3. ask once
          await new Promise<void>((r) => (choice.current = r));
          setNeedsChoice(false);
        }
        if (!(await baziness.init())) return; // 4. reloading into the linked
account
        if (loggedIn.current || baziness.isLinked !== me.user.isBazinessLinked) {
          me = await api<Me>("GET", "/api/me"); // 5. show the linked account
        }
      }
      onReady(me);
    })();
  }, [onReady]);

  const chooseGuest = () => { baziness.markChoiceMade(); choice.current?.(); };
  const chooseLogin = async () => {
    await baziness.init();
    if (!(await baziness.openLoginAndWait())) return; // closed without logging in → keep
the buttons
    loggedIn.current = true;
    choice.current?.();
  };

  return needsChoice ? (
    <div className="splash-choice">
      <button className="btn btn-green" onClick={chooseLogin}>ورود با حساب بازینس</button>
      <button className="btn btn-ghost" onClick={chooseGuest}>ادامه به عنوان مهمان</button>
    </div>
  ) : <ProgressBar />;
}
```

## C# (Unity)

```
// BazinessLogin.cs - the same protocol from a Unity client. Your SERVER still does every
// platform call; Unity only talks to your server and shows the page in a WebView.
using System;
using System.Collections;
using UnityEngine;
using UnityEngine.Networking;

public class BazinessLogin : MonoBehaviour
{
  const string TokenKey = "baziness_token", ChoiceKey = "baziness_choice";
  [SerializeField] string serverUrl = "https://api.yourgame.com";
  [SerializeField] BazinessSheet sheet; // see the Bottom sheet page
}
```

```

public bool Enabled { get; private set; }
public bool IsLinked { get; private set; }
public string Token { get; private set; }
public string PackageName => packageName;
public string GameOfferUrl { get; private set; } // from /api/platform/token
string webBase, apiBase, packageName;
Action<bool> loginGate;

[Serializable] class TokenResponse {
    public bool enabled; public string webBase, apiBase, packageName, token,
switchToDeviceId;
    public long bazinessUserId; public bool isLinkedToAccount;
}

public IEnumerator Init(string deviceId)
{
    var body = $"{{"deviceId":"{deviceId}"},"linkedToken":
{Quote(PlayerPrefs.GetString(TokenKey, null))}}";
    using var req = new UnityWebRequest($"{serverUrl}/api/platform/token", "POST");
    req.uploadHandler = new UploadHandlerRaw(System.Text.Encoding.UTF8.GetBytes(body));
    req.downloadHandler = new DownloadHandlerBuffer();
    req.SetRequestHeader("Content-Type", "application/json");
    yield return req.SendWebRequest();
    if (req.result != UnityWebRequest.Result.Success) { Enabled = false; yield break; }

    var r = JsonUtility.FromJson<TokenResponse>(req.downloadHandler.text);
    Enabled = r.enabled;
    if (!r.enabled) yield break;
    (webBase, apiBase, packageName) = (r.webBase, r.apiBase, r.packageName);
    if (!string.IsNullOrEmpty(r.switchToDeviceId) && r.switchToDeviceId != deviceId)
    {
        DeviceIdentity.Set(r.switchToDeviceId); // adopt the other account ...
        PlayerPrefs.DeleteKey(TokenKey);
        GameBoot.Restart(); // ... and boot again as that player
        yield break;
    }
    if (!string.IsNullOrEmpty(r.token)) { Token = r.token; IsLinked =
r.isLinkedToAccount; PlayerPrefs.SetString(TokenKey, r.token); }
}

public void OpenLogin(Action<bool> done)
{
    loginGate = done;
    sheet.Open(PageUrl("/", allowGuest: false), OnRawPageMessage, onClosed: () => {
loginGate?.Invoke(IsLinked); loginGate = null; });
}

public string PageUrl(string path, string redirect = null, bool allowGuest = true, string
extra = null)
{
    var url = $"{webBase}{path}?{extra}{(extra != null ? "&" : "")}packageName=
{Uri.EscapeDataString(packageName)}&sdkType=js";
    if (!string.IsNullOrEmpty(Token)) url += "&linkedToken=
{Uri.EscapeDataString(Token)}";
    if (redirect != null) url += "&redirect={redirect}";
    if (!allowGuest) url += "&allowGuest=false";
    return url;
}

public void OnRawPageMessage(string json) { /* see "Messages from the Baziness page"

```

```
below */ }
    static string Quote(string s) => string.IsNullOrEmpty(s) ? "null" : $"\"{s}\"";
}
```

## رسیدگی به پیام‌های صفحه

همان‌طور که در صفحه‌های بازی‌س ← پیام‌ها آمده. `logout` و `token` ، `login` ، `close`

### TypeScript (web)

```
// Inside BazinessPlatform (continued).
const extractToken = (data: unknown): string | undefined =>
    typeof data === "string" ? data
    : data && typeof (data as { token?: unknown }).token === "string" ? (data as { token:
string }).token
    : undefined;

handleMessage(msg: { type: string; data?: unknown }) {
    const token = extractToken(msg.data);
    switch (msg.type) {
        case "close":
            this.sheet.close();
            break;
        case "login":
            if (token) void this.applyNewLogin(token); // signed into an account → switch or link
            else void this.linkedInPlace(); // linked onto the current player
            break;
        case "token":
            break; // silent handshake – NOT a login
        case "logout":
            this.sheet.close();
            localStorage.clear(); // identity, token, choice, tutorial
            flags...
            this.reloadClean(); // next boot = brand-new visitor
            break;
    }
}

private async applyNewLogin(token: string) {
    const result = await fetch("/api/platform/login", {
        method: "POST", headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ deviceId: this.deviceId, token }),
    }).then((r) => r.json());
    if (result.switchToDeviceId && result.switchToDeviceId !== this.deviceId) {
        this.markChoiceMade();
        this.switchAccount(result.switchToDeviceId); // reloads
        return;
    }
    if (result.ok) {
        localStorage.setItem(TOKEN_KEY, token);
        await this.init(); // fresh token + linked flag
        this.loginSucceeded = true;
        this.markChoiceMade();
        this.sheet.close(); // handleClose resolves the login gate
    }
    // on failure keep the page open so the player can retry
}
```

```

private async linkedInPlace() {
    this.isLinked = true;
    this.loginSucceeded = true;
    this.markChoiceMade();
    if (this.LoginGate) { await this.init(); this.sheet.close(); }
    this.onChange?.();
}

private handleClosed() {
    const gate = this.LoginGate;
    this.LoginGate = null;
    gate?.(this.loginSucceeded); // true only if a login actually happened
    this.onChange?.(); // coin balance / link state may have
changed
}

private switchAccount(deviceId: string) {
    setDeviceId(deviceId); // your own identity store
    localStorage.removeItem(TOKEN_KEY);
    this.reloadClean();
}

private reloadClean() {
    const url = new URL(location.href);
    url.searchParams.delete("linkedToken"); // never re-apply a stale token
    url.searchParams.delete("device"); // a ?device= test override would undo the
switch
    location.replace(url.toString());
}

```

## React

```

// BazinessSheet.tsx – listening for page messages (only from the page's own origin).
function parse(data: unknown): { type: string; data?: unknown } | null {
    let value = data;
    if (typeof data === "string") {
        try { value = JSON.parse(data); } catch { return null; }
    }
    return value && typeof value === "object" && typeof (value as { type?: unknown }).type ===
"string"
        ? (value as { type: string; data?: unknown })
        : null;
}

useEffect(() => {
    if (!url) return;
    const origin = new URL(url).origin;
    const onMessage = (event: MessageEvent) => {
        if (event.origin !== origin) return; // ignore HMR, extensions, other frames
        const msg = parse(event.data);
        if (msg) baziness.handleMessage(msg);
    };
    window.addEventListener("message", onMessage);
    return () => window.removeEventListener("message", onMessage);
}, [url]);

```

## C# (Unity)

```

// Called by BazinessSheet with the raw JSON the page posted (see the Bottom sheet page).
// Uses Newtonsoft.Json (com.unity.nuget.newtonsoft-json) because `data` can be a string OR
an object.
using Newtonsoft.Json.Linq;

public void OnRawPageMessage(string json)
{
    JObject msg;
    try { msg = JObject.Parse(json); } catch { return; }
    var type = (string)msg["type"];
    var data = msg["data"];
    string token = data?.Type == JTokenType.String ? (string)data : (string)data?["token"];

    switch (type)
    {
        case "close": sheet.Close(); break;
        case "login":
            if (!string.IsNullOrEmpty(token)) StartCoroutine(ApplyNewLogin(token)); // POST
//api/platform/login
            else { IsLinked = true; StartCoroutine(Init(DeviceIdentity.Get()));
sheet.Close(); }
            break;
        case "token": break; // silent handshake - NOT a login
        case "logout":
            sheet.Close();
            PlayerPrefs.DeleteAll(); // forget identity, token and choice
            GameBoot.Restart(); // boot again as a new visitor
            break;
    }
}
}

```

## نمایش موجودی

### TypeScript (web)

```

// Inside BazinessPlatform: the player's Gita coin balance, straight from the platform. null
= unknown.
async getCoin(): Promise<number | null> {
    if (!this.enabled || !this.token || !this.apiBase) return null;
    try {
        const res = await fetch(`${this.apiBase}/user/get-coin`, {
            headers: { Authorization: `Bearer ${this.token}`, "Content-Type": "application/json" },
        });
        if (!res.ok) return null;
        const json = (await res.json()) as { amount?: number };
        return typeof json.amount === "number" ? json.amount : null;
    } catch {
        return null;
    }
}

// Wiring: refresh on every relevant moment.
baziness.onChange = () => void baziness.getCoin().then(renderCoinChip); // sheet closed,
login, link
window.addEventListener("focus", () => void baziness.getCoin().then(renderCoinChip));

```

## React

```
// session.tsx - keep the external balance in context and refresh it on the right events.
const [gita, setGita] = useState<number | null>(null);

const refreshGita = useCallback(async () => {
  if (baziness.enabled) setGita(await baziness.getCoin());
}, []);

useEffect(() => {
  baziness.onChange = () => void refreshGita(); // fires when the sheet closes / login changes
  void refreshGita();
  const onFocus = () => void refreshGita();
  window.addEventListener("focus", onFocus);
  return () => window.removeEventListener("focus", onFocus);
}, [refreshGita]);

// Anywhere in the UI:
<span className="gita-chip">
  <CurrencyIcon currency="GITA_COIN" />
  <bdi>{gita === null ? "..." : faNum(gita)}</bdi>
</span>
```

## C# (Unity)

```
using System;
using System.Collections;
using UnityEngine;
using UnityEngine.Networking;

[Serializable] class CoinResponse { public long amount; }

// apiBase and the user token come from /api/platform/token (see the Login page).
public IEnumerator GetCoin(Action<long?> done)
{
  if (!Enabled || string.IsNullOrEmpty(Token)) { done(null); yield break; }
  using var req = UnityWebRequest.Get($"{apiBase}/user/get-coin");
  req.SetRequestHeader("Authorization", $"Bearer {Token}");
  req.SetRequestHeader("Content-Type", "application/json");
  yield return req.SendWebRequest();
  if (req.result != UnityWebRequest.Result.Success) { done(null); yield break; }
  done(JsonUtility.FromJson<CoinResponse>(req.downloadHandler.text).amount);
}

// Refresh when the app regains focus and when the Baziness sheet closes.
void OnApplicationFocus(bool focus) { if (focus) StartCoroutine(GetCoin(UpdateCoinChip)); }
```

## دکمه‌ی جایزه

## TypeScript (web)

```
// Inside BazinessPlatform. GAME_NAME comes from /api/platform/config (or a constant).
openRewardSelection(): void {
  if (!this.enabled || !this.webBase) return;
  this.sheet.open(
```

```

    this.pageUrl("/shop", { redirect: "shop", extra: { game_name: this.gameName } })),
    BAZINESS_BOTTOM_BAR_CROP_PX, // optional: hide the platform's own tab bar (see below)
  );
}

// The lobby banner
rewardBanner.onclick = () => {
  if (baziness.enabled) baziness.openRewardSelection();
  else toast("انتخاب جایزه بعد از اتصال به بازی‌نس فعال می‌شود");
};

```

## React

```

// Home.tsx - the «انتخاب جایزه» banner with the live balance.
const { gita } = useSession(); // from get-coin, see the Gita coins page

const openRewards = () => {
  if (baziness.enabled) baziness.openRewardSelection();
  else toast("انتخاب جایزه بعد از اتصال به بازی‌نس فعال می‌شود", "info");
};

return (
  <button className="hero-banner reward-banner" onClick={openRewards}>
    <span className="hero-text">
      <h2>انتخاب جایزه</h2>
      <span className="hero-sub">گیتا کوین‌هایت را با جایزه‌ی واقعی عوض کن</span>
      <span className="gita-chip">
        <CurrencyIcon currency="GITA_COIN" size={30} />
        <bdi className="num">{gita === null ? "..." : faNum(gita)}</bdi>
      </span>
    </span>
    <img className="hero-art" src={art("ui/banner-rewards")} alt="" />
  </button>
);

```

## C# (Unity)

```

// RewardBanner.cs - attach to the lobby button.
public class RewardBanner : MonoBehaviour
{
  [SerializeField] BazinessLogin platform;
  [SerializeField] BazinessSheet sheet;
  [SerializeField] TMPro.TMP_Text balanceLabel;
  [SerializeField] string gameName = "example";

  void OnEnable() => StartCoroutine(platform.GetCoin(ShowBalance));

  public void OnClick()
  {
    if (!platform.Enabled) { Toast.Show("انتخاب جایزه بعد از اتصال به بازی‌نس فعال می‌شود");
    return; }

    var url = platform.PageUrl("/shop", redirect: "shop", extra: $"game_name={
    Uri.EscapeDataString(gameName)}");
    sheet.Open(url, platform.OnRawPageMessage, onClosed: () =>
    StartCoroutine(platform.GetCoin(ShowBalance)));
  }
}

```

```
void ShowBalance(long? amount) => balanceLabel.text = amount?.ToString("N0") ?? "...";
}
```

## پنهان کردن نوار پایین

روش [صفحه‌ی جایزه](#): فریم را بلندتر از نما کنید و بگذارید نما آن را برش بدهد.

### TypeScript (web)

```
/** Pixels cropped off the bottom of the embedded page. 0 = show the platform's tab bar. */
export const BAZINESS_BOTTOM_BAR_CROP_PX = 0;

// in BazinessSheet.open(url, cropBottomPx):
this.frame.style.marginBottom = cropBottomPx > 0 ? `${-cropBottomPx}px` : "";
```

## صفحه‌ی پیشنهاد آموزش

صفحه‌ی پیشنهاد را یک بار وسط آموزش باز کنید و وقتی بسته شد بسته‌ی شروع را بدهید ([صفحه‌ی پیشنهاد آموزش](#)).

### TypeScript (web)

```
// Inside BazinessPlatform.
gameOfferUrl = "";
private closeListeners = new Set<() => void>();

/** Run fn whenever the sheet closes; returns the unsubscribe function. */
onSheetClosed(fn: () => void) { this.closeListeners.add(fn); return () =>
this.closeListeners.delete(fn); }
// ...and in handleClosed(): for (const fn of [...this.closeListeners]) fn();

get canOpenSheet() { return this.enabled && !!this.webBase; }

/** Onboarding only. Adds the SDK parameters to the server-provided URL. */
openGameOffer() {
    if (!this.gameOfferUrl) return;
    const u = new URL(this.gameOfferUrl); // already has ?game_name=
    u.searchParams.set("packageName", this.packageName);
    u.searchParams.set("sdkType", "js"); // or its close button can't close
the sheet
    if (this.token) u.searchParams.set("linkedToken", this.token);
    u.searchParams.set("allowGuest", "false");
    this.sheet.open(u.toString());
}

// Lobby: the coach-marked banner
function onRewardBannerTap() {
    if (onboarding.stage === "rewards") {
        onboarding.setStage("pack"); // persist first: a reload resumes at
the pack
        if (baziness.canOpenSheet && baziness.gameOfferUrl) {
            const off = baziness.onSheetClosed(() => { off(); showStarterPack(); });
            baziness.openGameOffer();
        } else {
            showStarterPack(); // platform off: never block

```

```

onboarding
{
    return;
}
baziness.openRewardSelection(); // after onboarding: the real shop
}

```

## React

```

// Home.tsx: the onboarding branch of the «انتخاب جایزه» banner.
const openRewards = () => {
    if (coach === "rewards") {
        setOnboardingStage("pack"); // persisted in localStorage
        if (baziness.canOpenSheet && baziness.gameOfferUrl) {
            const off = baziness.onSheetClosed(() => {
                off();
                setCoach("pack"); // → StarterPackModal
            });
            setCoach("shop");
            baziness.openGameOffer();
        } else {
            setCoach("pack");
        }
        return;
    }
    if (baziness.enabled) baziness.openRewardSelection();
    else toast("انتخاب جایزه بعد از اتصال به بازینس فعال میشود", "info");
};

{coach === "rewards" && (
    <div className="coach-veil" aria-hidden="true">
        <div className="coach-bubble">
            <img src={asset("/assets/ui/tutorial-guide.png")} alt="" />
            <p>!بزن تا ببینی با گیتا کوین‌هایت چه جایزه‌هایی می‌توانی بگیری </b></p>
        </div>
    </div>
)}

```

## C# (Unity)

```

public void OnRewardBannerTap()
{
    if (Onboarding.Stage == OnboardingStage.Rewards)
    {
        Onboarding.SetStage(OnboardingStage.Pack); // persist before opening
        if (platform.Enabled && !string.IsNullOrEmpty(platform.GameOfferUrl))
        {
            var url = platform.GameOfferUrl
                + "&packageName={Uri.EscapeDataString(platform.PackageName)}&sdkType=js"
                + "&linkedToken={Uri.EscapeDataString(platform.Token ??
                "" )}&allowGuest=false";
            sheet.Open(url, platform.OnRawPageMessage, onClosed: StarterPack.Show);
        }
        else StarterPack.Show(); // never block onboarding
        return;
    }
    rewardBanner.OnClick(); // after onboarding: /shop
}

```

## پنجره‌ی خرید

تب را داخل همان ضربه باز کنید، خرید را در سرور شروع کنید، تب را به صفحه‌ی پرداخت ببرید و وقتی بازیکن برگشت بررسی کنید (پرداخت بازینس).

### TypeScript (web)

```
// purchase.ts - the purchase popup.
export function buy(productId: string, ui: PurchaseUi) {
  const tab = window.open("about:blank", "_blank"); // 1. synchronously, no
  "noopener"
  let paymentId: string | null = null;
  let started = false;
  let verifying = false;

  const verify = async (manual: boolean) => {
    if (!started || verifying || !paymentId) return;
    verifying = true;
    ui.showChecking();
    try {
      const r = await api("POST", "/api/shop/purchase/verify", { paymentId });
      if (r.status === "FULFILLED") { cleanup(); ui.success(r.granted); void
refreshWallets(); return; }
      if (manual) ui.toast("پرداختی پیدا نشد. اگر پرداخت کرده‌ای چند لحظه بعد دوباره بررسی کن.");
      ui.showWaiting();
    } catch (err) {
      if (manual) ui.error(err);
      ui.showWaiting();
    } finally { verifying = false; }
  };

  const onFocus = () => void verify(false);
  const onVisible = () => document.visibilityState === "visible" && void verify(false);
  const cleanup = () => { removeEventListener("focus", onFocus);
document.removeEventListener("visibilitychange", onVisible); };
  addEventListener("focus", onFocus);
  document.addEventListener("visibilitychange", onVisible);

  ui.showOpening(() => verify(true) /* «پرداخت کردم» */, () => { void verify(false); cleanup();
} /* close */);
  (async () => {
    try {
      const r = await api("POST", "/api/shop/purchase/start", { productId }); // 2.
      paymentId = r.paymentId;
      if (tab && !tab.closed) tab.location.href = r.url;
      else window.open(r.url, "_blank", "noopener");
      started = true;
      ui.showWaiting(); // 3.
    } catch (err) {
      tab?.close(); cleanup(); ui.error(err);
    }
  })();
}
```

### React

```

// Shop.tsx + PurchaseModal.tsx: open the tab in the click, then mount the modal.
export function openPayTab(): Window | null {
    return window.open("about:blank", "_blank"); // must run inside the click handler
}

// <button onClick={() => setBuying({ product, payTab: openPayTab() })}>...</button>

export function PurchaseModal({ product, payTab, onClose }: Props) {
    const { setWallets, toast, refresh } = useSession();
    const [phase, setPhase] = useState<"opening" | "waiting" | "checking" | "done">("opening");
    const paymentId = useRef<string | null>(null);
    const started = useRef(false);
    const verifying = useRef(false);

    const verify = useCallback(async (manual: boolean) => {
        if (!paymentId.current || verifying.current || !started.current) return;
        verifying.current = true;
        setPhase("checking");
        try {
            const res = await api<VerifyPurchaseResponse>("POST", "/api/shop/purchase/verify", {
                paymentId: paymentId.current });
            setWallets(res.wallets);
            if (res.status === "FULFILLED") { setPhase("done"); toast("خرید با موفقیت انجام شد", "success"); void refresh(); return; }
            if (manual) toast("پرداختی پیدا نشد. اگر پرداخت کرده‌ای چند لحظه بعد دوباره بررسی کن", "info");
            setPhase("waiting");
        } catch (err) {
            if (manual) toast(errorMessage(err), "error");
            setPhase("waiting");
        } finally { verifying.current = false; }
    }, [refresh, setWallets, toast]);

    useEffect(() => { // start once
        (async () => {
            try {
                const res = await api<StartPurchaseResponse>("POST", "/api/shop/purchase/start", {
                    productId: product.id });
                paymentId.current = res.paymentId;
                if (payTab && !payTab.closed) payTab.location.href = res.url;
                else window.open(res.url, "_blank", "noopener");
                started.current = true;
                setPhase("waiting");
            } catch (err) { payTab?.close(); toast(errorMessage(err), "error"); onClose(); }
        })();
    }, []);

    useEffect(() => { // verify when the player comes back
        const onFocus = () => void verify(false);
        const onVisible = () => document.visibilityState === "visible" && void verify(false);
        window.addEventListener("focus", onFocus);
        document.addEventListener("visibilitychange", onVisible);
        return () => { window.removeEventListener("focus", onFocus);
            document.removeEventListener("visibilitychange", onVisible); };
    }, [verify]);

    const close = () => { if (phase !== "done" && paymentId.current) void verify(false);

```

```

onClose(); };

return (
    <Modal open onClose={close}>
        <h3>{product.title}</h3>
        {phase === "done" ? <p>✓ به حسابت اضافه شد</p>
        : phase === "waiting" ? (<
            <p>صفحه‌ی پرداخت بازینس باز شد. بعد از پرداخت به بازی برگرد</p>
            <button onClick={() => void verify(true)}>پرداخت کردم</button>
        </>)}
        : <Spinner />}
    </Modal>
);
}

```

## C# (Unity)

```

// PurchaseFlow.cs – on mobile the payment page opens in the system browser, so there is no
// popup blocker. Verify when the app regains focus, plus a manual «پرداخت کردم» button.
public class PurchaseFlow : MonoBehaviour
{
    [SerializeField] GameApi api;           // your HTTP wrapper around /api/shop/purchase/*
    string paymentId;
    bool started, verifying;

    public IEnumerator Buy(string productId)
    {
        StartResponse r = null;
        yield return api.Post<StartResponse>("/api/shop/purchase/start", new { productId }, x
=> r = x);
        if (r == null) { Toast.Show("خطا در اتصال به درگاه"); yield break; }
        paymentId = r.paymentId;
        Application.OpenURL(r.url);         // {base}/buy?sku&uid&pname, built by YOUR server
        started = true;
        PurchasePanel.ShowWaiting(onPaid: () => StartCoroutine(Verify(manual: true)));
    }

    void OnApplicationFocus(bool focus)
    {
        if (focus && started) StartCoroutine(Verify(manual: false)); // back from the
browser
    }

    IEnumerator Verify(bool manual)
    {
        if (verifying || string.IsNullOrEmpty(paymentId)) yield break;
        verifying = true;
        VerifyResponse v = null;
        yield return api.Post<VerifyResponse>("/api/shop/purchase/verify", new { paymentId },
x => v = x);
        verifying = false;
        if (v?.status == "FULFILLED") { started = false; PurchasePanel.Success(v.granted);
Wallet.Set(v.wallets); }
        else if (manual) Toast.Show("پرداختی پیدا نشد. اگر پرداخت کرده‌ای چند لحظه بعد دوباره
کن بررسی کن.");
    }

    [System.Serializable] class StartResponse { public string paymentId, url; }
    [System.Serializable] class VerifyResponse { public string status; public Grant[]

```

```
granted; public Wallets wallets; }
}
```

## Cocos Creator

یک مجموعه‌ی کامل برای Cocos Creator 3.x در سه فایل:

| فایل                  | نقش                                                                                                                       |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------|
| BazinessSDK.ts        | یک کامپوننت ماندگار: صفحه‌ها را باز می‌کند، پیام‌هایشان را مسیریابی می‌کند و موجودی را می‌خواند                           |
| BazinessWebView.ts    | کامپوننت روی پری فب WebView: لایه‌ی بارگذاری، دکمه‌ی بستن، انیمیشن‌های لغزیدن، پیام‌ها                                    |
| BazinessController.ts | اتصال‌های زمان بالا آمدن: <code>linkedToken</code> از آدرس، <code>/api/platform/token</code> ، ورود، خروج، جابه‌جایی حساب |

پری فب یک `WebView`، یک نود لایه‌ی بارگذاری، یک دکمه‌ی بستن و یک `Animation` با کلیپ‌های `Open` و `Close` (لغزیدن به بالا و پایین) لازم دارد. در بیلد وب، `WebView` یک `iframe` است، پس پیام‌های صفحه به شکل رویداد `message` روی `window` می‌رسند.

### BazinessSDK.ts

```
// BazinessSDK.ts – a persistent component (put it on a node in the first scene).
// Opens Baziness pages in a WebView prefab, routes their messages and reads the Gita coin balance.
import { _decorator, Component, director, find, instantiate, Prefab, EventTarget } from "cc";
import { BazinessWebView } from "../BazinessWebView";
const { ccclass, property } = _decorator;

export const bazinessEvents = new EventTarget(); // "login" | "logout" | "closed"

const WEB = { dev: "https://platform-web-dev.karizmastudios.org", prod: "https://baziness.com" };
const API = { dev: "https://platform-dev.karizmastudios.org/api", prod: "https://platform.karizmastudio.com/api" };

@ccclass("BazinessSDK")
export class BazinessSDK extends Component {
    @property(Prefab) webViewPrefab: Prefab = null!;

    static instance: BazinessSDK;
    static packageName = "com.yourstudio.yourgame";
    static devMode = true;
    static token: string | null = null; // the player's user token, from YOUR server
    static isLinked = false;

    private view: BazinessWebView | null = null;

    onLoad() {
```

```

        BazinessSDK.instance = this;
        director.addPersistRootNode(this.node);
    }

    /** Call after your server answered /api/platform/token. */
    static initialize(token: string, isLinked: boolean) {
        this.token = token;
        this.isLinked = isLinked;
    }

    static openLogin() {
        this.instance.open(`${this.web()}/`, { allowGuest: false });
    }

    static openProfile() {
        this.instance.open(`${this.web()}/profile`, { redirect: "profile", allowGuest: false });
    }

    static openRewards(gameName: string) {
        this.instance.open(`${this.web()}/shop`, { redirect: "shop", extra: { game_name: gameName
    } });
    }

    /** The Gita coin balance, straight from the platform with the player's own token. */
    static async getCoin(): Promise<number | null> {
        if (!this.token) return null;
        try {
            const res = await fetch(`${this.devMode ? API.dev : API.prod}/user/get-coin`, {
                headers: { Authorization: `Bearer ${this.token}`, "Content-Type": "application/json"
            },
            });
            return res.ok ? ((await res.json()).amount ?? null) : null;
        } catch {
            return null;
        }
    }

    private static web() {
        return this.devMode ? WEB.dev : WEB.prod;
    }

    private open(base: string, o: { redirect?: string; allowGuest?: boolean; extra?:
Record<string, string> } = {}) {
        if (this.view) return; // one page at a time
        const q = new URLSearchParams(o.extra);
        q.set("packageName", BazinessSDK.packageName);
        q.set("sdkType", "js"); // the page posts close / login / logout / token
        if (BazinessSDK.token) q.set("linkedToken", BazinessSDK.token);
        if (o.redirect) q.set("redirect", o.redirect);
        if (o.allowGuest === false) q.set("allowGuest", "false");

        const node = instantiate(this.webViewPrefab);
        node.setParent(find("Canvas"));
        this.view = node.getComponent(BazinessWebView!);
        this.view.onMessage = (m) => this.onMessage(m);
        this.view.onClosed = () => {
            this.view = null;
            bazinessEvents.emit("closed");
        };
        this.view.open(`${base}?${q}`);
    }
}

```

```

private onMessage(msg: { type: string; data?: unknown }) {
  const token = typeof msg.data === "string" ? msg.data : (msg.data as { token?: string } |
undefined)?.token;
  switch (msg.type) {
    case "close":
      this.view?.close();
      break;
    case "login":
      if (token) {
        this.view?.close(true);
        bazinessEvents.emit("login", token); // → POST /api/platform/login on your server
      } else {
        BazinessSDK.isLinked = true; // linked in place
      }
      break;
    case "logout":
      this.view?.close(true);
      bazinessEvents.emit("logout");
      break;
    case "token":
      break; // the page's silent handshake – not a login
  }
}
}
}

```

#### BazinessWebView.ts

```

// BazinessWebView.ts – the component on the WebView prefab (WebView + loading overlay +
close button).
import { _decorator, Animation, Button, Component, Node, WebView } from "cc";
const { ccclass, property } = _decorator;

@ccclass("BazinessWebView")
export class BazinessWebView extends Component {
  @property(WebView) webView: WebView = null!;
  @property(Node) loadingOverlay: Node = null!;
  @property(Button) closeButton: Button = null!;
  @property(Animation) animation: Animation = null!; // "Open" slides up, "Close" slides down

  onMessage: ((m: { type: string; data?: unknown }) => void) | null = null;
  onClosed: (() => void) | null = null;

  private origin = "";
  private closeTimer: ReturnType<typeof setTimeout> | null = null;
  private listener = (e: MessageEvent) => this.onWindowMessage(e);

  open(url: string) {
    this.origin = new URL(url).origin;
    this.webView.node.active = false;
    this.closeButton.node.active = false;
    // Our own close button appears after a few seconds: a way out even if the page never
    loads.
    this.closeTimer = setTimeout(() => {
      if (this.closeButton?.isValid) this.closeButton.node.active = true;
    }, 3000);
    this.closeButton.node.on(Button.EventType.CLICK, () => this.close(), this);
    this.webView.node.on(WebView.EventType.LOADED, () => {
      this.loadingOverlay.active = false;
      this.webView.node.active = true;
    });
  }
}

```

```

    }, this);
    this.webView.node.on(WebView.EventType.ERROR, () => this.close(), this);
    window.addEventListener("message", this.listener); // on web builds the WebView is an
iframe
    this.webView.url = url;
    this.animation.play("Open");
}

close(immediately = false) {
    window.removeEventListener("message", this.listener);
    if (this.closeTimer) clearTimeout(this.closeTimer);
    if (!immediately) this.animation.play("Close");
    setTimeout(() => {
        this.onClosed?.();
        if (this.node?.isValid) this.node.destroy();
    }, immediately ? 0 : 500);
}

private onWindowMessage(e: MessageEvent) {
    if (e.origin !== this.origin) return; // only the page we opened
    try {
        const msg = typeof e.data === "string" ? JSON.parse(e.data) : e.data;
        if (msg && typeof msg.type === "string") this.onMessage?.(msg);
    } catch {
        /* not a Baziness message */
    }
}
}
}

```

#### BazinessController.ts

```

// BazinessController.ts – boot-time wiring between your server, the SDK and the page
messages.
import { BazinessSDK, bazinessEvents } from "./BazinessSDK";
import { GameApi } from "./GameApi"; // your HTTP client for /api/*
import { LocalData } from "./LocalData"; // your localStorage wrapper

export class BazinessController {
    /** Returns true when it started an account switch or a reset – stop your own boot then. */
    static async initialize(): Promise<boolean> {
        // The Baziness app opens games with ?linkedToken=... . If it doesn't match the token this
device
        // already holds, a different Baziness account is arriving: start clean before going on.
        const fromUrl = new URLSearchParams(location.search).get("linkedToken");
        const saved = LocalData.get("baziness_token");
        if (fromUrl && saved && fromUrl !== saved) {
            this.resetAndReload();
            return true;
        }

        const resp = await GameApi.post("/api/platform/token", { linkedToken: fromUrl ?? saved
});
        if (!resp.enabled) return false; // platform off: no Baziness UI
        if (resp.switchToDeviceId) {
            this.switchTo(resp.switchToDeviceId);
            return true;
        }
        if (resp.token) {
            LocalData.set("baziness_token", resp.token);

```

```

        BazinessSDK.initialize(resp.token, resp.isLinkedToAccount);
    }

    bazinessEvents.on("login", async (token: string) => {
        const r = await GameApi.post("/api/platform/login", { token }); // the server validates
it
        if (r.switchToDeviceId) this.switchTo(r.switchToDeviceId);
        else if (r.ok) {
            LocalData.set("baziness_token", token);
            await this.initialize();
        }
    });
    bazinessEvents.on("logout", () => this.resetAndReload());
    return false;
}

private static switchTo(deviceId: string) {
    LocalData.clear();
    LocalData.set("device_id", deviceId);
    this.reload();
}

private static resetAndReload() {
    LocalData.clear(); // the next boot is a brand-new visitor
    this.reload();
}

private static reload() {
    const url = new URL(location.href);
    url.searchParams.delete("linkedToken");
    location.replace(url.toString()); // or reload your loading scene
}
}

```

# شیت پایین (وبویوی داخل بازی)

شیت کشویی که همه‌ی صفحه‌های بازی‌س را داخل بازی نشان می‌دهد. از پایین بالا می‌آید، با کشیدن، زدن بیرون یا دکمه‌ی خود صفحه بسته می‌شود و پیام‌های صفحه را به بازی می‌رساند.

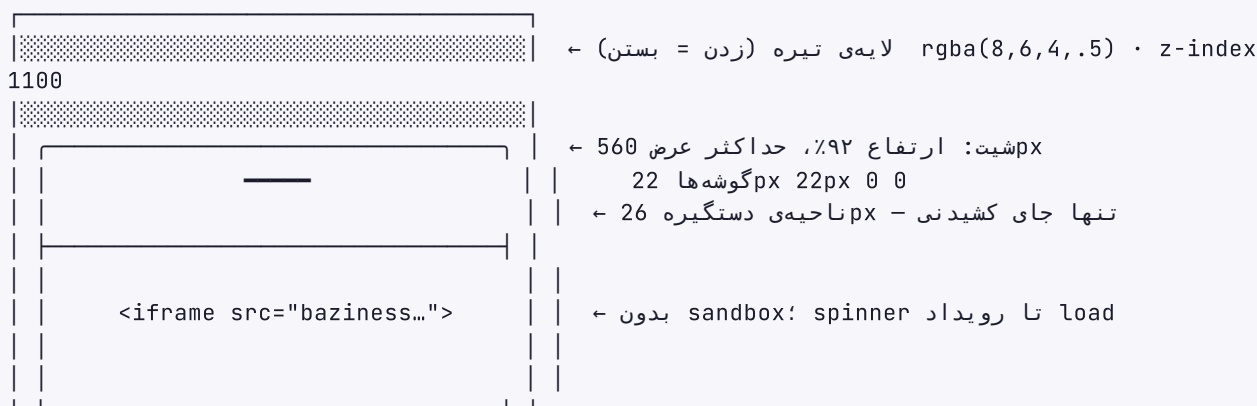
## چرا شیت

همه‌ی صفحه‌های بازی‌س (ورود، پروفایل، فروشگاه جایزه، پیشنهاد بازی) داخل بازی باز می‌شوند، نه در تب جدید. ظرف پیشنهادی یک شیت پایین است: روی بازی تیره شده از پایین بالا می‌آید، کمی پایین‌تر از بالای صفحه می‌ایستد و لبه‌ی بالایی گرد با یک دستگیره دارد. بازیکن به سه روش می‌تواند آن را ببندد و خود صفحه هم می‌تواند.

امتحان‌ش کنید:

## اجزا

Text



## رفتارهای لازم

1. باز شدن: شیت را زیر صفحه بکشید و در فریم بعد به بالا متحرک کنید (حدود ۳۲۰ میلی‌ثانیه، ease-out).

2. سه راه بستن:

- کشیدن دستگیره به پایین بیشتر از `min(150px, 28% ارتفاع شیت)`؛ کشیدن کوتاه‌تر شیت را به جایش برمی‌گرداند
- زدن روی ناحیه‌ی تیره‌ی بالای شیت
- صفحه پیام `{ "type": "close" }` بفرستد

3. فقط دستگیره کشیدن است. صفحه لمس‌های خودش را می‌بلعد، پس کشیدن که داخل صفحه شروع شود شیت را تکان نمی‌دهد.

4. روی `iframe sandbox` نگذارید. صفحه‌ی ورود به کوکی‌ها، پنجره‌های بازشو (برای کد) و `redirect` خودش نیاز دارد. `clipboard-read; clipboard-write; payment` را مجاز کنید.
5. تا بارگذاری صفحه یک `spinner` نشان دهید.
6. پیام‌ها را فقط از `origin` خود صفحه بپذیرید. هم JSON رشته‌ای و هم شیء ساده را بخوانید و `type` رشته‌ای را الزامی کنید.
7. تمیز ببندید: به پایین متحرک کنید، بعد فقط یک بار جمعش کنید: فریم را به `about:blank` ببرید، گوش دادن به پیام‌ها را قطع کنید و بعد «بسته شد» را اعلام کنید. اگر در این فاصله شیت دوباره باز شده، کاری نکنید.
8. بالای همه‌چیز باشد، اسپلش هم: صفحه‌ی ورود ممکن است از اسپلش باز شود.
9. اندازه‌اش مثل ستون بازی باشد: در گوشه‌ی تمام‌عرض، در صفحه‌های پهن حداکثر حدود ۵۶۰ پیکسل و وسط‌چین.

## وب: CSS و TypeScript، React

### TypeScript (web)

```
// bazinesswebview.ts - the sheet, framework-free.
export interface BazinessMessage { type: string; data?: unknown }

export class BazinessSheet {
  readonly el: HTMLElement;
  private sheet: HTMLElement;
  private frame: HTMLIFrameElement;
  private spinner: HTMLElement;
  private allowedOrigin: string | null = null;
  private isOpen = false;
  private dragging = false;
  private dragStartY = 0;
  private dragDelta = 0;
  private sheetHeight = 1;

  onMessage: ((m: BazinessMessage) => void) | null = null;
  onClose: (() => void) | null = null;

  constructor() {
    this.frame = Object.assign(document.createElement("iframe"), { className: "baziness-frame" });
    this.frame.setAttribute("allow", "clipboard-read; clipboard-write; payment"); // no sandbox!
    this.spinner = Object.assign(document.createElement("div"), { className: "baziness-spinner" });
    const grabberZone = document.createElement("div");
    grabberZone.className = "baziness-grabber-zone";
    grabberZone.innerHTML = '<div class="baziness-grabber"></div>';
    this.sheet = document.createElement("div");
    this.sheet.className = "baziness-sheet";
    this.sheet.append(grabberZone, this.spinner, this.frame);
    this.el = document.createElement("div");
    this.el.className = "overlay baziness-webview hidden";
    this.el.append(this.sheet);
    document.body.append(this.el);

    this.frame.addEventListener("load", () => this.spinner.classList.add("hidden"));
    grabberZone.addEventListener("pointerdown", (e) => this.dragStart(e));
    grabberZone.addEventListener("pointermove", (e) => this.dragMove(e));
```

```

grabberZone.addEventListener("pointerup", () => this.dragEnd());
grabberZone.addEventListener("pointercancel", () => this.dragEnd());
this.el.addEventListener("pointerdown", (e) => { if (e.target === this.el) this.close();
}); // tap outside
}

open(url: string, cropBottomPx = 0): void {
  if (this.isOpen) return;
  this.isOpen = true;
  this.frame.style.marginBottom = cropBottomPx > 0 ? `${-cropBottomPx}px` : "";
  this.allowedOrigin = new URL(url).origin;
  this.spinner.classList.remove("hidden");
  this.el.classList.remove("hidden");
  this.sheet.style.transition = "none";
  this.sheet.style.transform = "translateY(100%)";
  this.frame.src = url;
  window.addEventListener("message", this.onWindowMessage);
  requestAnimationFrame(() => {
    this.sheet.style.transition = "";
    this.sheet.style.transform = "translateY(0)";
  });
}

close(): void {
  if (!this.isOpen) return;
  this.isOpen = false;
  this.dragging = false;
  window.removeEventListener("message", this.onWindowMessage);
  this.sheet.style.transition = "";
  this.sheet.style.transform = "translateY(100%)";
  let torn = false;
  const teardown = () => {
    if (torn || this.isOpen) return; // already torn down, or reopened meanwhile
    torn = true;
    this.sheet.removeEventListener("transitionend", teardown);
    this.el.classList.add("hidden");
    this.frame.src = "about:blank";
    this.frame.style.marginBottom = "";
    this.sheet.style.transform = "";
    this.allowedOrigin = null;
    this.onClosed?.();
  };
  this.sheet.addEventListener("transitionend", teardown);
  window.setTimeout(teardown, 400);
}

private dragStart(e: PointerEvent) {
  this.dragging = true;
  this.dragStartY = e.clientY;
  this.dragDelta = 0;
  this.sheetHeight = this.sheet.getBoundingClientRect().height || 1;
  this.sheet.style.transition = "none";
  (e.currentTarget as HTMLElement).setPointerCapture(e.pointerId);
}

private dragMove(e: PointerEvent) {
  if (!this.dragging) return;
  this.dragDelta = Math.max(0, e.clientY - this.dragStartY); // downward only
  this.sheet.style.transform = `translateY(${this.dragDelta}px)`;
}

private dragEnd() {

```

```

    if (!this.dragging) return;
    this.dragging = false;
    this.sheet.style.transition = "";
    if (this.dragDelta > Math.min(150, this.sheetHeight * 0.28)) this.close();
    else this.sheet.style.transform = "translateY(0)"; // spring back
}

private onWindowMessage = (event: MessageEvent) => {
    if (this.allowedOrigin && event.origin !== this.allowedOrigin) return;
    let value: unknown = event.data;
    if (typeof value === "string") { try { value = JSON.parse(value); } catch { return; } }
    if (value && typeof value === "object" && typeof (value as BazinessMessage).type ===
"string") {
        this.onMessage?.(value as BazinessMessage);
    }
};
}

```

## React

```

// BazinessSheet.tsx – the React version. Mount once at the app root; baziness.host drives
it.
import { useEffect, useRef, useState } from "react";
import { createPortal } from "react-dom";
import { baziness } from "../platform/baziness";

export function BazinessSheet() {
    const [url, setUrl] = useState<string | null>(null);
    const [shown, setShown] = useState(false);
    const [loading, setLoading] = useState(true);
    const [drag, setDrag] = useState(0);
    const dragStart = useRef<number | null>(null);

    useEffect(() => {
        baziness.host = {
            open: (next) => {
                setLoading(true);
                setUrl(next);
                requestAnimationFrame(() => requestAnimationFrame(() => setShown(true))); // slide up
            },
            close: () => setShown(false),
        };
        return () => { baziness.host = null; };
    }, []);

    useEffect(() => { // messages, from the page's origin
only
        if (!url) return;
        const origin = new URL(url).origin;
        const onMessage = (e: MessageEvent) => {
            if (e.origin !== origin) return;
            const msg = parse(e.data); // see "Handling page messages"
(Client code)
            if (msg) baziness.handleMessage(msg);
        };
        window.addEventListener("message", onMessage);
        return () => window.removeEventListener("message", onMessage);
    }, [url]);
}

```

```

const onTransitionEnd = () => {
  if (!shown && url) { // finished sliding down → tear down
    once
    setUrl(null);
    setDrag(0);
    baziness.handleClosed();
    measureViewport(); // the phone may have moved its nav
  }
};

if (!url) return null;
return createPortal(
  <div className={`overlay baziness-overlay ${shown ? "is-open" : ""}` `
    onPointerDown={(e) => e.target === e.currentTarget && setShown(false)}>
    <div className="baziness-sheet"
      style={{ transform: shown ? `translateY(${drag}px)` : "translateY(100%)",
        transition: dragStart.current !== null ? "none" : undefined }}
      onTransitionEnd={onTransitionEnd}>
      <div className="baziness-grabber-zone"
        onPointerDown={(e) => { dragStart.current = e.clientY;
e.currentTarget.setPointerCapture(e.pointerId); }}
        onPointerMove={(e) => dragStart.current !== null && setDrag(Math.max(0,
e.clientY - dragStart.current))}
        onPointerUp={() => { dragStart.current = null; if (drag > 140) setShown(false);
else setDrag(0); }}>
        <span className="baziness-grabber" />
      </div>
      {loading && <div className="baziness-loading"><Spinner /></div>}
      <iframe className="baziness-frame" src={url} title="بازینس"
        allow="clipboard-read; clipboard-write; payment" onLoad={() =>
setLoading(false)} />
      </div>
    </div>,
    document.getElementById("app-frame") ?? document.body,
  );
}

```

## CSS

```

/* Above the splash (z 1000) so it shows during the first-time login choice. */
.overlay.baziness-webview {
  position: fixed;
  inset: 0;
  z-index: 1100;
  display: flex;
  justify-content: center;
  align-items: flex-end; /* dock the sheet to the bottom */
  background: rgba(8, 6, 4, 0.5); /* the dimmed, tap-to-close area */
}

.overlay.baziness-webview.hidden { display: none; }

.baziness-sheet {
  position: relative;
  width: min(560px, 100%); /* full width on phones, capped on wide screens */
  height: 92%; /* stops short of the top */
  display: flex;
  flex-direction: column;
  background: #fff;
}

```

```

border-radius: 22px 22px 0 0;
overflow: hidden; /* also clips the cropped bottom bar */
box-shadow: 0 -10px 34px rgba(0, 0, 0, 0.42);
transform: translateY(100%);
transition: transform 320ms cubic-bezier(0.22, 0.61, 0.36, 1);
will-change: transform;
}

/* The only draggable area – the iframe captures its own pointer events. */
.baziness-grabber-zone {
  flex: 0 0 auto;
  height: 26px;
  display: flex;
  align-items: center;
  justify-content: center;
  cursor: grab;
  touch-action: none;
  background: #fff;
}
.baziness-grabber {
  width: 44px;
  height: 5px;
  border-radius: 999px;
  background: rgba(0, 0, 0, 0.22);
}

.baziness-frame {
  flex: 1 1 auto;
  width: 100%;
  min-height: 0;
  border: 0;
  background: #fff;
}

.baziness-spinner {
  position: absolute;
  top: 50%;
  left: 50%;
  width: 42px;
  height: 42px;
  margin: -21px 0 0 -21px;
  border-radius: 50%;
  border: 4px solid rgba(120, 90, 46, 0.25);
  border-top-color: #b4824a;
  animation: spin 0.9s linear infinite;
}
.baziness-spinner.hidden { display: none; }
@keyframes spin { to { transform: rotate(360deg); } }

```

## Cocos Creator

کامپوننت `BazinessWebView` در کد کلاینت ← Cocos Creator همین کار را با یک پری فب WebView انجام می‌دهد: لایه‌ی بارگذاری تا لود شدن صفحه، دکمه‌ی بستنی که بعد از چند ثانیه ظاهر می‌شود، و انیمیشن‌های لغزیدن / `Open` `Close`.

Unity وب‌ویوی داخلی ندارد و به پلاگین نیاز دارید. این طرح از [gree/unity-webview](#) استفاده می‌کند: صفحه ۸٪ پایین‌تر از بالای صفحه شروع می‌شود، دستگیره یک المان **UI** یونیتی بالای آن است و پیام‌های صفحه با یک پل کوچک جاوااسکریپتی به **C#** می‌رسند.

## C# (Unity)

```
// BazinessSheet.cs - a bottom sheet around a WebView plugin (gree/unity-webview).
using System;
using UnityEngine;
using UnityEngine.EventSystems;

public class BazinessSheet : MonoBehaviour, IDragHandler, IEndDragHandler // put this on
the grabber UI
{
    [SerializeField] RectTransform grabber; // a 26px bar drawn by Unity above the web
view
    [SerializeField] GameObject dimmer; // full-screen button behind; onClick →
Close()
    WebViewObject web;
    Action<string> onMessage;
    Action onClose;
    float topMargin, drag;

    public void Open(string url, Action<string> onMessage, Action onClose = null)
    {
        this.onMessage = onMessage;
        this.onClosed = onClose;
        topMargin = Screen.height * 0.08f; // the sheet stops 8% short of the top
        drag = 0;
        web = new GameObject("BazinessWebView").AddComponent<WebViewObject>();
        web.Init(
            cb: msg => this.onMessage?.Invoke(msg), // messages forwarded by the bridge
below
            ld: _ => web.EvaluateJS(
                "window.addEventListener('message',function(e){\" +
                \"Unity.call(typeof e.data==='string'?e.data:JSON.stringify(e.data));});\"));
        Layout();
        web.LoadURL(url);
        web.SetVisibility(true);
        dimmer.SetActive(true);
        grabber.gameObject.SetActive(true);
    }

    public void OnDrag(PointerEventData e) { drag = Mathf.Max(0, drag - e.delta.y); Layout();
}

    public void OnEndDrag(PointerEventData e)
    {
        if (drag > Mathf.Min(150, (Screen.height - topMargin) * 0.28f)) Close();
        else { drag = 0; Layout(); } // spring back
    }

    public void Close()
    {
        if (web == null) return;
        Destroy(web.gameObject);
    }
}
```

```

web = null;
dimmer.SetActive(false);
grabber.gameObject.SetActive(false);
onClosed?.Invoke();
}

void Layout()
{
    int top = (int)(topMargin + drag + grabber.rect.height);
    web.SetMargins(0, top, 0, 0);
    grabber.anchoredPosition = new Vector2(0, -(topMargin + drag));
}
}

```

## اشتباه‌های رایج

| نتیجه                                                                                      | اشتباه                                                   |
|--------------------------------------------------------------------------------------------|----------------------------------------------------------|
| ورود گیر می‌کند: پنجره‌های بازشو و کوکی‌ها مسدود می‌شوند                                   | iframe روی <code>sandbox</code>                          |
| کشیدن داخل صفحه هم صفحه را اسکرول می‌کند و هم شیت را تکان می‌دهد، یا هیچ چیز تکان نمی‌خورد | کل شیت کشیدنی است                                        |
| پیامی از جای دیگر به جای پیام صفحه گرفته می‌شود                                            | بررسی origin برای پیام‌ها نیست                           |
| ورودی که در انتظار است دو بار تمام می‌شود؛ بسته‌ی شروع دو بار باز می‌شود                   | «بسته شد» دو بار اعلام می‌شود (پایان انیمیشن + زمان سنج) |
| صفحه‌ی ورود نادیده پشت اسپلش باز می‌شود                                                    | شیت زیر اسپلش است                                        |
| صفحه‌ی قبلی در پس‌زمینه به اجرا ادامه می‌دهد                                               | فریم به <code>about:blank</code> برنمی‌گردد              |

## بازی نمونه – سنگ، کاغذ، قیچی

یک بازی کامل و کوچک که همه‌ی یکپارچه‌سازی‌های این ویکی را دارد. روی سرور توسعه بازی‌اش کنید، پروژه را دانلود کنید و در دو دقیقه خودتان اجراش کنید.

---

### بازی کنید

**باز کردن بازی نمونه** (سرورهای توسعه: platform-dev، platform-web-dev، pay-dev).

**دانلود پروژه (zip):** Node.js و جاوااسکریپت مرورگر ساده، بدون هیچ وابستگی و بدون مرحله‌ی بیلد.

### چه چیزهایی را نشان می‌دهد

یک مسابقه‌ی سنگ، کاغذ، قیچی «بهترین از ۵ دست» در برابر ربات: اولین کسی که **۳ امتیاز** بگیرد برنده است و دست مساوی امتیاز ندارد. هر مسابقه **۱ بلیت** هزینه دارد و برنده **۵۰ گیتا کوین** می‌گیرد. همه‌ی قابلیت‌های پلتفرم در آن هست:

| قابلیت                                                                                                               | کجای بازی                                                        | صفحه‌ی ویکی                | کد                                                                                                       |
|----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|----------------------------|----------------------------------------------------------------------------------------------------------|
| ورود با بازی‌نس<br>(انتخاب اسپلش)،<br>جابه‌جایی حساب،<br>خروج                                                        | اجرای اول؛ تنظیمات ← «اتصال حساب»                                | ورود                       | <pre> public/js/platform.js server/services.js     platformToken     platformLogin </pre>                |
| ارزها؛ بلیت (محلی)<br>و گیتا کوین<br>(get-coin)                                                                      | نشان‌های نوار بالا، بنر جایزه                                    | گیتا کوین                  | <pre> public/js/platform.js     getCoin public/js/app.js     renderTopbar </pre>                         |
| ۵۰ گیتا کوین برای<br>برنده (add-coin)<br>با کلید بیکتا)، برای<br>دریافت ورود لازم<br>است                             | پنجره‌ی نتیجه، نوار «جایزه منتظر توست»                           | گیتا کوین                  | <pre> server/services.js     claimMatchReward public/js/app.js     claim </pre>                          |
| صفحه‌ی «انتخاب<br>جایزه»                                                                                             | بنر بنفش                                                         | صفحه‌ی جایزه               | <pre> public/js/platform.js     renderRewardSelection </pre>                                             |
| آموزش: قانون‌ها ←<br>راهنما روی «انتخاب<br>جایزه» ← صفحه‌ی<br>game-offer ←<br>«خرید موفق»<br>بسته‌ی شروع (۵<br>بلیت) | اجرای اول                                                        | صفحه‌ی<br>پیشنهاد<br>آموزش | <pre> public/js/app.js     آموزش server/services.js     claimStarterPack </pre>                          |
| شیت کشویی                                                                                                            | همه‌ی صفحه‌های بازی‌نس                                           | شیت پایین                  | <pre> public/js/sheet.js public/styles.css </pre>                                                        |
| پرداخت بازی‌نس:<br>«بسته‌ی بلیت»، ۱۰<br>بلیت به ۴,۹۰۰<br>تومان (SKU)<br>(example_pack)                               | کارت فروشگاه                                                     | پرداخت<br>بازی‌نس          | <pre> server/pay.js server/services.js     startPurchase     verifyPurchase public/js/purchase.js </pre> |
| گرفتن UTM و ?<br>tag= به‌عنوان<br>اولین تماس                                                                         | هر لینکی، مثلاً .../example/?<br>utm_source=wiki&utm_medium=docs | لینک‌ها و<br>UTM           | <pre> public/js/api.js     FIRST_TOUCH server/services.js     ensureUser </pre>                          |

## Shell

```
unzip baziness-example-rps.zip && cd baziness-example-rps
cp .env.example .env          # fill in BAZINESS_CLIENT_ID / BAZINESS_CLIENT_SECRET
npm start                     # → http://127.0.0.1:4380/
```

پیش‌نیاز: **Node.js 20** یا بالاتر. داده‌ها در `data/db.json` نگه داشته می‌شوند، یک ذخیره‌ساز JSON کوچک به جای دیتابیس شما.

| مقدار توسعه                                                        | .env                                        |
|--------------------------------------------------------------------|---------------------------------------------|
| کلاینت توسعه‌ی نمونه (از تیم پلتفرم بگیرید، یا کلاینت بازی خودتان) | BAZINESS_CLIENT_ID / BAZINESS_CLIENT_SECRET |
| com.karizmastudio.example                                          | BAZINESS_PLATFORM_PACKAGE_NAME              |
| https://pay-dev.karizmastudios.org                                 | BAZINESS_PAY_BASE_URL                       |
| com.karizmastudio.example                                          | BAZINESS_PAY_PACKAGE_NAME                   |
| 4900 / example_pack                                                | PACK_PRICE / PACK_SKU                       |

## پیشنهاد

**کلید ندارید؟** باز هم اجرا می‌شود. با `BAZINESS_PLATFORM_ENABLED=false` رابط بازی‌نس پنهان می‌شود و جایزه‌ها محلی نگه داشته می‌شوند. با `DEV_FAKE_BAZINESS_PAY=true` و `BAZINESS_PAY_BASE_URL=http://localhost:4380/dev/baziness`، خریدها از یک درگاه آزمایشی داخلی با دکمه‌ی «پرداخت موفق» رد می‌شوند.

## نکته‌های تست

- **ورود:** روی این نسخه‌ی توسعه، «ورود با حساب بازی‌نس» هر شماره موبایلی را با کد `111111` می‌پذیرد. پیامکی فرستاده نمی‌شود.
- **دوباره دیدن آموزش:** از «بازنشانی این دستگاه» پایین لابی استفاده کنید. حافظه‌ی محلی پاک می‌شود و بار بعد بازیکن کاملاً جدید است.
- **بازیکن دوم در همان مرورگر:** `?device=someone-else` را به آدرس اضافه کنید.
- **دریافت جایزه به‌عنوان مهمان:** بدون ورود یک مسابقه را ببرید. درخواست دریافت `PLATFORM_NOT_LINKED` برمی‌گرداند و شیت ورود باز می‌شود. بعد از ورود، دریافت خودش دوباره تلاش می‌کند.
- **خریدی که بعداً تمام می‌شود:** خرید را شروع کنید، پرداخت کنید، بازی را ببندید و دوباره باز کنید. لابی پرداخت‌های در انتظار را تطبیق می‌دهد.

- لینک کمپین: `.../example/?utm_source=wiki&utm_campaign=test` را باز کنید. آدرس بعد از بارگذاری تمیز است و برچسب‌ها هنگام ساخت حساب روی آن ذخیره می‌شوند.

## ساختار پروژه

### Text

```

baziness-example-rps/
├── .env.example          every setting, documented
├── package.json          "npm start" – no dependencies
├── server/
│   ├── index.js          HTTP routes + static files (relative paths: works at / or
/example/)
│   ├── env.js            configuration (.env loader)
│   └── platform.js        platform client: auth, create-user, tokens, add-coin ← secret
stays here
├── pay.js                Baziness Pay: buy URL, get_payment, verify_payment (form), consume
├── services.js            accounts & linking, match rules, rewards, starter pack, purchases
├── store.js              JSON-file store (swap for your database)
├── fake-pay.js           local fake gateway (never in production)
└── public/
    ├── index.html, styles.css
    ├── assets/           art (hands, bot, pack, prize, Gita coin, ticket, Baziness logo)
    └── js/
        ├── api.js        first module: UTM capture, device id, fetch wrapper
        ├── platform.js    BazinessPlatform: page URLs, messages, login, get-coin
        ├── sheet.js       the swipe-down sheet
        ├── purchase.js    Pay popup: tab opened in the click, verify on focus
        ├── ui.js          DOM helpers, Persian numbers, toasts, modals
        └── app.js          splash, lobby, match, tutorial

```

## میزبانی زیر یک مسیر

کلاینت فقط از آدرس‌های نسبی (`assets/...` ، `api/...`) استفاده می‌کند، پس همین فایل‌ها هم در `/` و هم زیر یک پیشوند کار می‌کنند. روی سرور توسعه‌ی ویکی، `nginx` پیشوند `/example/` را برمی‌دارد و درخواست را به پروسه‌ی `Node` می‌فرستد:

### NGINX

```

location = /example { return 301 /example/; }
location /example/ {
    proxy_pass http://127.0.0.1:4380/;          # trailing slash: /example/api/me → /api/me
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

```

# چک لیست انتشار و رفع اشکال

هر چیزی که باید پیش از روشن کردن بازی‌نس در محیط اصلی بررسی کنید، و راه حل رایج‌ترین مشکلات یکپارچه‌سازی.

## پیش از انتشار

### تنظیمات

- شناسه و کلید محرمانه‌ی **محیط اصلی** (جدا از توسعه)، فقط روی سرور.
- `BAZINESS_PLATFORM_ENABLED=true` ، `BAZINESS_PLATFORM_USE_PRODUCTION=true` ، و نام بسته‌ی پلتفرم و نام بازی تنظیم شده.
- `BAZINESS_PAY_ENABLED=true` ، `BAZINESS_PAY_BASE_URL=https://pay.baziness.com` و نام بسته‌ی پرداخت تنظیم شده؛ درگاه آزمایشی محلی خاموش.
- بیلد کلاینت را برای کلید محرمانه جست‌وجو کنید: **نباید** آنجا باشد.
- `/api/platform/config` مقدار `enabled: true` را با آدرس‌های **محیط اصلی** برمی‌گرداند.

### ورود

روی پلتفرم توسعه همه‌ی حالت‌ها را با شماره‌های ساختگی و کد `111111` تست کنید، چون پیامکی فرستاده نمی‌شود (جزئیات).

- اسپلش «ورود با حساب بازی‌نس» / «ادامه به عنوان مهمان» را برای هر دستگاه یک بار می‌پرسد.
- بستن صفحه‌ی ورود بدون وارد شدن، دو دکمه را برمی‌گرداند.
- ورود به حسابی که روی دستگاه دیگری استفاده شده، به آن **جابه‌جا** می‌کند و پیشرفتش سر جایش است.
- تنظیمات ← اتصال حساب، صفحه‌ی پروفایل را باز می‌کند و بعدش «متصل» نشان می‌دهد.
- خروج داخل یک صفحه‌ی بازی‌نس، بازی را به حالت یک بازدیدکننده‌ی کاملاً جدید برمی‌گرداند.
- باز کردن بازی از اپ بازی‌نس ( `?linkedToken=` ) آموزش را برای بازیکن جدید اجرا می‌کند.

### سکه و صفحه‌ها

- موجودی گیتا کوین از پلتفرم می‌آید: تا وقتی معلوم نیست ... ، هیچ‌وقت یک `0` ساختگی.
- موجودی بعد از بسته شدن صفحه و بعد از فوکوس دوباره‌ی بازی تازه می‌شود.
- هر پرداخت یک‌باره `transactionId` ای دارد که از ماهیت پرداخت ساخته شده، و `400` پرداخت شده حساب می‌شود.
- «انتخاب جایزه» `/shop?game_name=...` را بدون دیوار ورود باز می‌کند.
- (اگر استفاده می‌کنید) ضربه‌ی اول آموزش `/game-offer` را باز می‌کند و بستنش به بسته‌ی شروع می‌رسد.
- نما با دستگیره، با زدن بیرون و با دکمه‌ی خود صفحه بسته می‌شود.

## لینک‌ها

- ☐ • لینک هر صفحه `packageName` و `sdkType=js` و در صورت وجود `linkedToken` را دارد.
- ☐ • لینک‌های کمپین هر چهار `utm_*` را به ASCII دارند و نوار آدرس بعد از بارگذاری تمیز است.
- ☐ • کمپین بعد از ورود بازینس و جابه‌جایی حساب حفظ می‌شود.

## پرداخت بازینس

- ☐ • همه‌ی SKUها زیر نام بسته‌ی پرداخت شما ثبت شده‌اند.
- ☐ • یک خرید واقعی در محیط اصلی یک بار تحویل و بعد consume شده.
- ☐ • خریدن دو بار پشت سر هم همان بسته کار می‌کند، که نشان می‌دهد `consume` اجرا شده.
- ☐ • پرداختی که بعد از بستن بازی تمام شده، دفعه‌ی بعد که فروشگاه باز می‌شود تحویل داده می‌شود.

| نشانه                                              | علت احتمالی                                                                                                 | راه حل                                                                                                                          |
|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| دکمه‌ی بستن صفحه کاری نمی‌کند                      | <code>sdkType=js</code> نیست، یا بررسی <code>origin</code> پیام با <code>origin</code> اشتباه مقایسه می‌کند | <code>sdkType=js</code> را به همه‌ی لینک‌های صفحه اضافه کنید و <code>origin</code> پیام را با آدرسی که باز کرده‌اید مقایسه کنید |
| فروشگاه جایزه به محض باز شدن بسته می‌شود           | دست‌دادن <code>token</code> ورود حساب شده                                                                   | <code>token</code> را نادیده بگیرید؛ فقط <code>login</code> ورود است                                                            |
| ورود اول گیر می‌کند یا سرور 400 می‌دهد             | ورود پیش از اینکه بازیکن کاربر بازیکن و توکن داشته باشد شروع شده                                            | اول <code>/api/platform/token</code> را صدا بزنید و صفحه را با همان توکن باز کنید                                               |
| وارد شده ولی بازی هنوز مهمان نشان می‌دهد           | <code>/api/platform/login</code> صدا زده نشده یا پروفایل دوباره خوانده نشده                                 | با <code>login</code> صدایش بزنید و بعد پروفایل بازیکن را دوباره بخوانید                                                        |
| صفحه‌ی ورود «ادامه به عنوان مهمان» نشان می‌دهد     | <code>allowGuest=false</code> نیست                                                                          | روی صفحه‌های ورود، پروفایل و پیشنهاد اضافه‌اش کنید (نه <code>/shop</code> )                                                     |
| صفحه‌ی جایزه از بازیکن می‌خواهد وارد شود           | ریشه‌ی سایت به جای <code>/shop</code> باز شده                                                               | <code>{web}/shop?game_name=...</code> را با <code>linkedToken</code> باز کنید                                                   |
| درخواست‌های پلتفرم بعد از مدتی شکست می‌خورند       | توکن کلاینت منقضی شده (401)                                                                                 | توکن تازه بگیرید (قدم ۱) و یک بار دیگر بفرستید                                                                                  |
| توکنی که درست به نظر می‌رسد رد می‌شود              | <code>validate-user-token</code> پاسخ <code>isValid: false</code> داده (HTTP 200)                           | فیلد <code>isValid</code> را بررسی کنید، نه فقط کد HTTP؛ کلاینت دوباره وارد شود                                                 |
| یک بازیکن همان سکه‌ها را دو بار گرفت               | <code>transactionId</code> از زمان ساخته شده                                                                | آن را از ماهیت پرداخت بسازید (قدم ۵)                                                                                            |
| دریافت جایزه بعد از یک کرش برای همیشه شکست می‌خورد | <code>add-coin</code> پاسخ 400 داده و دریافت آن را خطا حساب کرده                                            | 400 را «قبلاً پرداخت شده» حساب کنید                                                                                             |
| تب پرداخت سفید می‌ماند                             | تب بعد از منتظر ماندن برای سرور یا با <code>noopener</code> باز شده                                         | داخل همان ضربه یک تب خالی باز کنید و بعد آدرسش را تنظیم کنید                                                                    |
| بازیکن پرداخت کرد ولی چیزی نگرفت                   | هیچ وقت برگشت، پس بررسی اجرا نشد                                                                            | با فوکوس و «پرداخت کردم» بررسی کنید، و هنگام باز شدن فروشگاه پرداخت‌های در انتظار را بررسی کنید                                 |

| نشانه                                                 | علت احتمالی                                                 | راه حل                                                               |
|-------------------------------------------------------|-------------------------------------------------------------|----------------------------------------------------------------------|
| <code>verify_payment</code> همیشه شکست می خورد        | به شکل JSON فرستاده شده                                     | بدنه ی فرم بفرستید                                                   |
| پرداخت پاسخ <code>no_required_params</code> می دهد    | <code>uid</code> عددی نیست                                  | شناسه ی عددی بازیکن را بفرستید                                       |
| خرید دوم همان بسته مسدود است                          | خرید consume نشده                                           | بعد از دادن آیتم و با چند بار تلاش <code>consume</code> را صدا بزنید |
| بازیکنانی که از اپ بازیکن می آیند از آموزش رد می شوند | <code>?linkedToken=</code> نشانه ی «بازیکن قدیمی» گرفته شده | «بازیکن جدید» را از داده های خودتان تشخیص دهید                       |
| کمپین بعد از ورود گم می شود                           | برچسب ها فقط از آدرس خوانده شده اند و بازی از اول اجرا شده  | پیش از ورود آن ها را روی دستگاه ذخیره کنید                           |
| نام فارسی کمپین ها نیست                               | مقدارهای UTM فقط به صورت ASCII نگه داشته می شوند            | از نام ASCII استفاده کنید ( <code>shahrivar_1405</code> )            |
| نمای ورود روی اسپلش دیده نمی شود                      | نما زیر اسپلش است                                           | آن را بالای همه چیز بگذارید                                          |

## کمک گرفتن

- کلیدها، صفحه ی پیشنهاد بازی و SKUها: از تیم پلتفرم بازیکنس بپرسید.
- برای دیدن همه چیز در کنار هم، کد بازی نمونه را بخوانید.

# مرجع API

همه‌ی endpointها، آدرس‌ها و پیام‌ها در یک صفحه: API پلتفرم، پرداخت بازی‌نس، صفحه‌های بازی‌نس و endpointهایی که سرور خودتان باید داشته باشد.

---

## API پلتفرم

- اصلی: <https://platform.karizmastudio.com/api>
- توسعه: <https://platform-dev.karizmastudios.org/api>

توضیح قدم‌به‌قدم: API پلتفرم.

| متد و مسیر                                 | احراز هویت     | بدنه                                                 | پاسخ                                       | توضیح                                                                                                                            |
|--------------------------------------------|----------------|------------------------------------------------------|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| POST<br>/client/auth                       | —              | { clientId,<br>clientSecret<br>}                     | { clientId,<br>token }                     | توکن کلاینت                                                                                                                      |
| POST<br>/client/create-user                | توکن<br>کلاینت | { deviceId,<br>platform,<br>store,<br>linkedToken }  | { userId, token,<br>isLinkedToAccount<br>} | : platform<br>/ Web<br>/ Android<br>: Editor / Ios<br>: store<br>/ Direct<br>/ GooglePlay<br>/ AppStore<br>/ CafeBazaar<br>Myket |
| GET /client/get-user-token/{userId}        | توکن<br>کلاینت | —                                                    | { userId, token,<br>isLinkedToAccount<br>} | توکن کاربر تازه                                                                                                                  |
| GET<br>/client/validate-user-token/{token} | توکن<br>کلاینت | —                                                    | { userId,<br>isValid }                     | نامعتبر ← 200 با<br>isValid:<br>false و<br>userId: 0                                                                             |
| POST<br>/client/add-coin                   | توکن<br>کلاینت | { userId,<br>amount,<br>transactionId,<br>metaData } | { resourceId,<br>amount }<br>(موجودی جدید) | transactionId<br>400 { ← تکراری<br>success:<br>false, error:<br>"add resource<br>failed" }                                       |
| GET /user/get-coin                         | توکن کاربر     | —                                                    | { resourceId,<br>amount }                  | کلاینت صدا می‌زند؛<br>همه‌ی origin‌ها<br>مجازند                                                                                  |

هر درخواستی که توکن کلاینت نداشته باشد یا توکنش منقضی شده باشد پاسخ 401 می‌گیرد.

## صفحه‌های بازنیس

- اصلی: <https://baziiness.com>
- توسعه: <https://platform-web-dev.karizmastudios.org>

جزئیات: صفحه‌های بازی‌نس.

| صفحه                            | مسیر        | لازم                                                                            | اختیاری                                                                                    |
|---------------------------------|-------------|---------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| ورود                            | /           | <code>packageName</code><br><code>sdkType=js</code>                             | <code>linkedToken</code><br><code>allowGuest=false</code><br><code>buildStore</code>       |
| پروفایل                         | /profile    | <code>packageName</code><br><code>sdkType=js</code>                             | <code>linkedToken</code><br><code>redirect=profile</code><br><code>allowGuest=false</code> |
| فروشگاه جایزه<br>«انتخاب جایزه» | /shop       | <code>packageName</code><br><code>sdkType=js</code><br><code>linkedToken</code> | <code>redirect=shop</code> ، <code>game_name</code>                                        |
| پیشنهاد بازی                    | /game-offer | <code>game_name</code><br><code>packageName</code><br><code>sdkType=js</code>   | <code>linkedToken</code><br><code>allowGuest=false</code>                                  |

## پیام‌های صفحه‌ها

به شکل متن JSON یا شیء. `{ "type": string, "data"? : string | { "token": string } }`

| کار                                                              | type                         |
|------------------------------------------------------------------|------------------------------|
| به سرور خودتان بفرستید ← بررسی ← وصل یا جابه‌جایی ← بستن         | <code>login</code> با توکن   |
| در همان جا وصل شد ← تازه کردن                                    | <code>login</code> بدون داده |
| نادیده بگیرید (دست‌دادن بی‌صدا)                                  | <code>token</code>           |
| بستن، پاک کردن داده‌های محلی، شروع دوباره مثل بازدیدکننده‌ی جدید | <code>logout</code>          |
| بستن نما                                                         | <code>close</code>           |

## پرداخت بازی‌نس

- اصلی: `https://pay.baziness.com`
- توسعه: `https://pay-dev.karizmastudios.org`

`uid` باید عددی باشد. جزئیات: پرداخت بازی‌نس.

| متد و مسیر           | قالب بدنه | بدنه                       | پاسخ                                                                                       |
|----------------------|-----------|----------------------------|--------------------------------------------------------------------------------------------|
| GET /buy             | query     | { uid, sku, pname }        | صفحه‌ی پرداخت                                                                              |
| POST /get_payment    | JSON      | { uid, sku, pname }        | یا { uid, sku, token, time, success }<br>{ error: "payment_is_not_valid", success: false } |
| POST /verify_payment | فرم       | { uid, token, sku, pname } | { success, error }                                                                         |
| POST /consume        | JSON      | { uid, sku, pname }        | { success }                                                                                |

## endpoint های سرور شما (پیشنهادی)

| Endpoint                       | بدنه                                   | پاسخ                                                                                    |
|--------------------------------|----------------------------------------|-----------------------------------------------------------------------------------------|
| GET /api/platform/config       | —                                      | { enabled, webBase, apiBase, packageName, gameName, gameOfferUrl }                      |
| POST /api/platform/token       | { deviceId, linkedToken?, tag?, utm? } | { enabled, تنظیمات..., token?, userId?, isLinkedToAccount?, switchToDeviceId?, error? } |
| POST /api/platform/login       | { deviceId, token }                    | { ok, switchToDeviceId?, reason? }                                                      |
| GET /api/shop                  | —                                      | محصولاتی که این بازیکن می تواند بخرد                                                    |
| POST /api/shop/purchase/start  | { productId }                          | { paymentId, url }                                                                      |
| POST /api/shop/purchase/verify | { paymentId }                          | { status, granted, balances }                                                           |
| POST /api/shop/purchase/cancel | { paymentId }                          | —                                                                                       |
| GET /api/shop/purchase/pending | —                                      | [{ paymentId, productId }]                                                              |

| تنظیم                                                    | پیش فرض                            | معنی                                                              |
|----------------------------------------------------------|------------------------------------|-------------------------------------------------------------------|
| BAZINESS_PLATFORM_ENABLED                                | false                              | کلید اصلی حساب، سکه و صفحه‌ها                                     |
| BAZINESS_PLATFORM_USE_PRODUCTION                         | false                              | آدرس‌های اصلی یا توسعه                                            |
| / BAZINESS_CLIENT_ID<br>BAZINESS_CLIENT_SECRET           | —                                  | کلیدها (محرمانه)                                                  |
| BAZINESS_PLATFORM_PACKAGE_NAME                           | —                                  | در لینک صفحه‌ها packageName                                       |
| BAZINESS_GAME_NAME                                       | —                                  | در صفحه‌ی جایزه و پیشنهاد game_name                               |
| / BAZINESS_PLATFORM_API_URL<br>BAZINESS_PLATFORM_WEB_URL | محاسبه شده                         | آدرس جایگزین                                                      |
| BAZINESS_GAME_OFFER_URL                                  | {web}/game-offer?<br>game_name=... | آدرس جایگزین صفحه‌ی پیشنهاد                                       |
| BAZINESS_PAY_ENABLED                                     | false                              | کلید اصلی خرید                                                    |
| BAZINESS_PAY_BASE_URL                                    | —                                  | https://pay-dev.karizmastudios.org یا<br>https://pay.baziness.com |
| BAZINESS_PAY_PACKAGE_NAME                                | —                                  | pname                                                             |
| BAZINESS_SKU_PREFIX                                      | خالی                               | SKU = پیشوند + شناسه‌ی محصول                                      |
| DEV_FAKE_BAZINESS_PAY                                    | false                              | درگاه آزمایشی محلی (هرگز در محیط اصلی)                            |