بسمه تعالی

معماری اثبات‌شده برای توسعه بک‌اند پروژه "واچار"

یکی از اصلی ترین چالش های پروژه dad انتخاب معماری ای است با نیازهای پروژه و قابلیت‌های تیم همخوانی داشته باشد. برای تعیین زبان برنامه‌نویسی، فریم‌ورک و دیتابیس، جلساتی در تیم برگزار شد و معیارهایی شامل تخصص تیم، نیازمندی‌های پروژه و محدودیت زمانی را ارزیابی کردیم. در نهایت، پس از بررسی دقیق مزایا و معایب بین Django و Golang، انتخاب ما برای توسعه بک‌اند Python Django و برای دیتابیس PostgreSQL بود.

گزینه‌های فنی بررسی‌شده برای framework:

1. **Python Django**
2. **Golang**

معیارهای ارزیابی

1. **تخصص تیم و منحنی یادگیری: آشنایی تیم با زبان و فریم‌ورک.**
2. **نیازمندی‌های پروژه: همخوانی با نیازهای تجاری و محدودیت‌های فنی.**
3. **محدودیت زمانی: اهمیت سرعت تحویل پروژه.**
4. **پشتیبانی کامیونیتی : دسترسی به منابع و کامیونیتی در صورت بروز چالش.**
5. **نیازهای عملکردی: تطابق معیارهای عملکردی با نیازهای تجاری.**

تحلیل مزایا و معایب

معیار	Python Django	Golang
تخصص تیم	اکثر اعضای تیم در پایتون مهارت داشتند؛ یادگیری Django نیاز به تلاش کمی داشت.	تیم در Golang مهارت داشت، اما راه‌اندازی فریم‌ورک زمان‌بر بود.
سرعت توسعه	ویژگی‌های داخلی Django (مانند Django Admin) امکان نمونه‌سازی سریع را فراهم کرد.	Golang نیاز به پیاده‌سازی دستی حتی برای ویژگی‌های ساده داشت.
جامعه و اکوسیستم	پشتیبانی و منابع گسترده کامیونیتی برای Django.	کامیونیتی کوچک‌تر اما رو به رشد با منابع محدود برای Golang.

عملکرد	نسبت به Golang کندتر، اما برای کاربرد ما که متمرکز بر داده است کافی است. (data intensive)	سریع‌تر و مقیاس‌پذیرتر برای برنامه‌های پیچیده.
مقیاس‌پذیری آینده	مقیاس‌پذیری متوسط	مقیاس‌پذیری بالا برای کاربردهای آتی.

تصمیم نهایی

با توجه به تحلیل انجام‌شده، Python Django به دلایل زیر انتخاب شد:

- **سهولت استفاده: ویژگی‌های داخلی Django توسعه را تسریع کرده و برای محدودیت زمانی ما ایده‌آل بود.**
- **پشتیبانی از پنل‌های ادمین: Django Admin امکان پیاده‌سازی سریع پنل پشتیبانی برای تیم پروداکت و پشتیبانی را فراهم کرد.**
- **پشتیبانی کامیونیتی: منابع گسترده کامیونیتی به رفع چالش‌ها کمک می‌کند.**
- **کفایت عملکردی: اگرچه کمی کندتر از Golang بود، اما برای کاربرد ما کافی بود.**

استفاده از AI برای توسعه محصول

از آن جهت که پروژه پیچیدگی بالایی ندارد و پروژه‌هایی شبیه به آن در گیت‌هاب موجودند، استفاده از AI در توسعه این پروژه بسیار کمک‌دهنده خواهد بود. و حالا که تصمیم گرفته‌ایم که از Django استفاده کنیم، کدهای نمونه با این فریم‌ورک به وفور در بستر اینترنت یافت می‌شوند و ai به خوبی برای توسعه با این فریمورک جواب می‌دهد.

انتخاب دیتابیس sql

برای دیتابیس، PostgreSQL به دلایل زیر انتخاب شد:

- **قابلیت اطمینان اثبات‌شده: مورد استفاده در شرکت‌های بزرگ و نشان‌دهنده استحکام.**
- **یکپارچگی با Django: ادغام آسان با ORM Django برای توسعه کارآمد.**
- **جستجوی Full-Text: قابلیت‌های قدرتمند جستجوی Full-Text و indexing در PostgreSQL نیازهای ما در مدیریت و بازیابی داده‌های پیچیده را به خوبی برآورده می‌کند.**

ذخیره‌سازی تصاویر

برای ذخیره‌سازی تصاویر، تصمیم گرفتیم که از دیتابیس PostgreSQL استفاده کنیم. با این حال، با توجه به این که ممکن است در آینده این تکنولوژی پاسخگوی نیاز محصول نباشد و نیاز به مهاجرت به سیستم‌های ذخیره‌سازی فایل مانند Swift داشته باشیم، تصمیم بر آن شد که جداول مربوط به ذخیره‌سازی تصاویر کاملاً جدا باشند تا مهاجرت آن‌ها در آینده به راحتی انجام شود.

استراتژی تست

برای تضمین کیفیت کد و عملکرد سیستم، استراتژی تست ما شامل موارد زیر است:

1. **Unit Testing**: برای تست منطق‌های برنامه‌نویسی و اطمینان از عملکرد صحیح توابع و ماژول‌ها به صورت مستقل
2. **Integration Testing**: برای تست تعامل بین ماژول‌های مختلف سیستم و اطمینان از سازگاری و صحت روابط بین آن ها
3. **System Testing**: در جهت تست سیستم در محیط شبیه سازی شده و اطمینان از نیازهای عملکردی و غیرعملکردی سرویس.

فریم‌ورک **Django** امکانات گسترده‌ای برای انجام **Unit testing** و **Integration testing** با هزینه فنی پایین فراهم می‌کند. ابزارهای تست داخلی **Django** به ما کمک می‌کنند تا با کمترین تلاش، تست‌های مؤثری طراحی و اجرا کنیم. به این صورت که برای لایه **Unit testing** ابزارهایی برای ماک کردن وابستگی ها و تست عملکرد یک روند به صورت مستقل فراهم شده بود.

برای مسئله **Integration test** با توجه به کم بودن تعداد ماژول ها و اهمیت فراهم کردن آن در زمان کم تصمیم بر استفاده از روش **Big Bang** گرفتیم. به این صورت که به جز ماژول مربوط به ثبت نام کاربر تقریبا بقیه ماژول ها را به صورت واقعی در ارتباط با یکدیگر قرار می‌دهیم و به این صورت سناریو های مختلفی را تست می‌کنیم.

در زمینه **System test** برای اینکه بتوانیم دیدگاه سمت کاربر را وارد سناریو تست کنیم ترجیح دادیم تا سرویس را در محیط شبیه سازی بالا بیاوریم و با یک کلاینت اتومات، فرآیند ها را تست کنیم. در این زمینه تصمیم بر استفاده از زبان برنامه نویسی **Golang** برای شبیه سازی رفتار کلاینت شد. به این دلیل که فرآیندهای مربوط به کاربرهای متفاوت به صورت همزمان رخ میدهند و در این لایه لازم است استقلال آن ها از یکدیگر مورد بررسی قرار بگیرد. همچنین برای اطمینان از تاب آوری سرویس در برابر حجم درخواست ها لازم بود سناریوهایی را در جهت بررسی آن اعمال کنیم. زبان **Golang** به واسطه قابلیت هایی که در زمینه ایجاد **Concurrency** در پردازه ها دارد برای این نیازمندی ها مناسب بود.

مدیریت کاربران و کالاهای مسدود شده

برای هندل کردن کاربران و کالاهای مسدود شده، استراتژی زیر را اتخاذ کردیم:

- کاربران مسدود شده: به ازای هر کاربر این مورد که مسدود شده است یا خیر را ذخیره می کنیم و در **api** هایی که نیاز است **permission** ای که این مورد را بررسی کند قرار می دهیم.
- کالاهای مسدود شده: در جدول مربوط به کالاها، وضعیت مسدودیت ذخیره می‌شود و در **API** های مربوطه، **Validation** های مناسب برای بررسی این وضعیت اعمال می‌شود.

مدیریت نرخ درخواست ها (نیازمندی غیر عملکردی)

- استفاده از قابلیت **throttle** در **django**

معماری UI

روش انتخاب شده: UI flow/wireframe diagram

[لینک فیگما]

معماری فرانت اند

فریمورک انتخاب شده: NextJS

دلایل انتخاب:

Developer Experience

باتوجه به زمان کوتاه برای پیاده سازی، اینکه از استکی استفاده کنیم که تجربه توسعه خوب و راحتی داشته باشد، کامیونیتی مناسبی داشته باشد (که به معنای وجود تعداد زیادی مثال و کتابخانه و ... براشون هست) و تیممان تجربهٔ کار با آن را داشته باشد، حائز اهمیت است.

Performance

این مورد شاید با توجه به کوچک بودن پروژه در اولویت اول ما نباشد ولی nextJs در عین تجربهٔ توسعه مناسب، پرفورمنس خوبی هم ارائه میدهد. React پر استفاده ترین فریمورک توسعه وب است و next ، به اصطلاح (جوری که خودش ادعا میکند) در واقع یک فریمورک برای React محسوب میشود. لذا علاوه بر مزیت های React، بهبود ها و موارد آماده ای را هم به آن اضافه کرده است.

SEO-Friendly

با توجه به ماهیت فروشگاه اینترنتی، بهینه بودن برای موتور های جست و جو یک مزیت خیلی مهم برای آن به حساب می آید. nextJs با بهره گیری از server side rendering، این خواسته را نیز برآورده میکند.

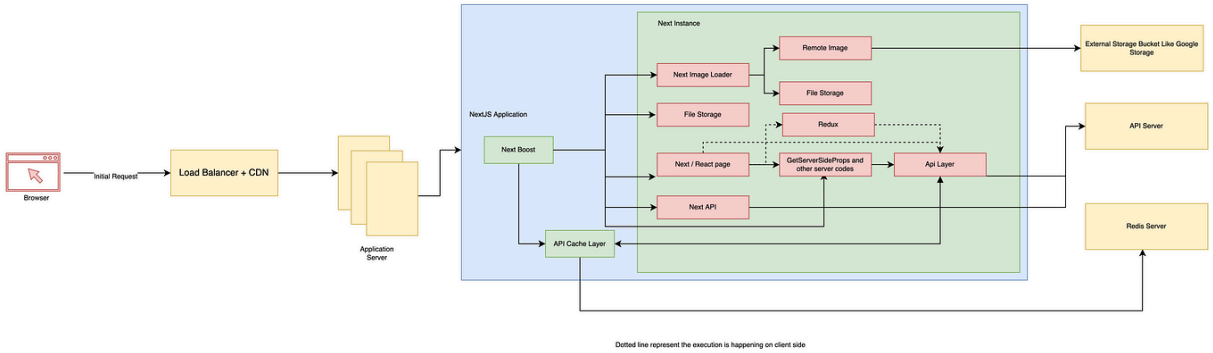
Knowledge

دانش استفاده از این فریمورک در تیم وجود دارد.

معماری کلی فرانت اند، با توجه به انتخاب NextJs به عنوان ستون اصلی آن تعیین میشود.

برای مثال ساختار کلی پروژه را خود Next معین میکند. (best practice)

همچنین از منابعی که در زمینه best practice های معماری یک اپلیکیشن با NextJS وجود دارد استفاده میکنیم. برای مثال این مقاله:



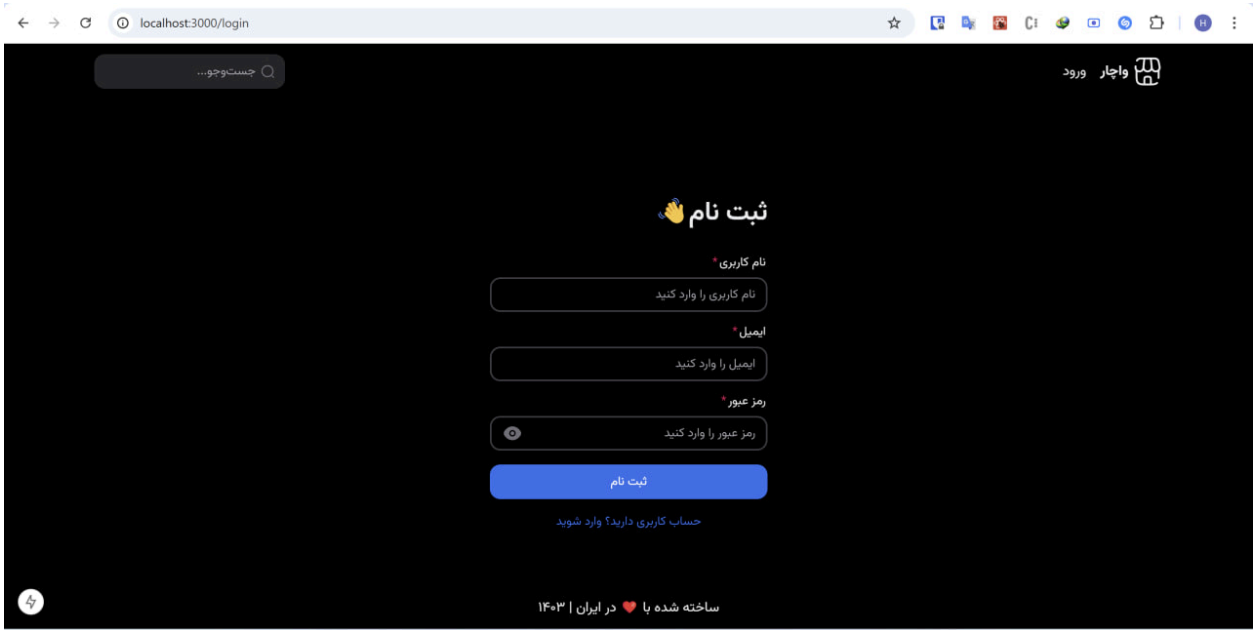
[NextJs Application Architecture for best performance]

کتابخانهٔ رابط کاربری: NextUI

دلایل انتخاب:

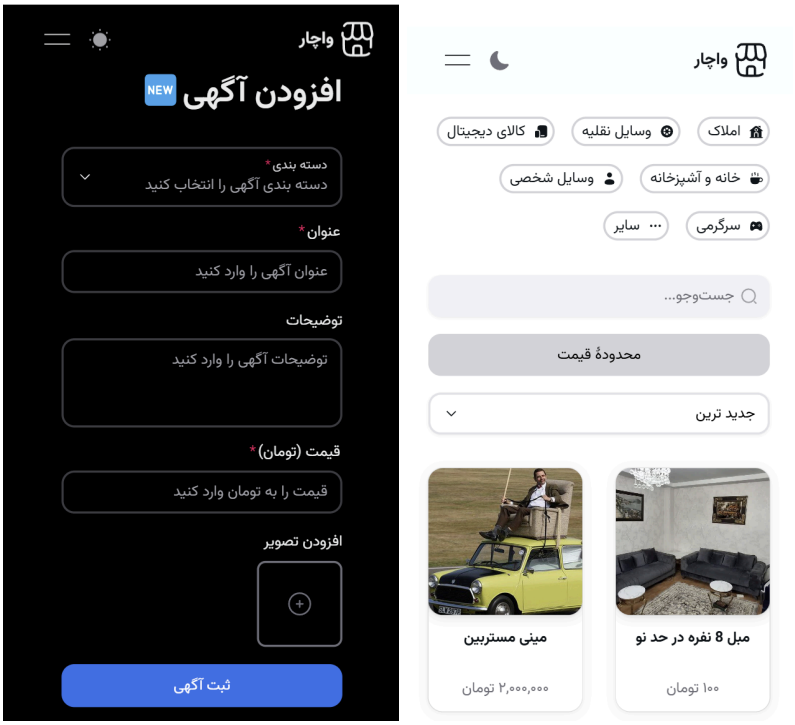
بهترین کتابخانه رابط کاربری مرتبط با NextJS بود (با توجه به میزان استفاده و star در گیتهاب)

راه حل های آماده برای کامپوننت های پرکاربرد (مثل سرچ بار) و صفحات پرتکرار (مثل لاگین) ارائه میداد.



نمایی از صفحه ثبت نام پروژه واچار

ریسپانسیو بودن رابط کاربری (نیازمندی غیر عملکردی)



دسترسی پذیری (accessibility)

به واسطه‌ی کتابخانه‌ی رابط کاربری ای که استفاده کردیم و رعایت eslint-rules های ست شده در پایپلاین تست، پروژه برای screen reader ها بهینه شده است.

Server Side Rendering

با استفاده از امکانات NextJS ، از این قابلیت استفاده کردیم تا سایت برای موتور های جست و جو هم بهینه باشد. چون اگر صرفا از react استفاده میکردیم و تمام فرایند rendering در کلاینت رخ میداد، نمیتوانستیم توسط موتور های جست و جو ایندکس شویم. در حالی که این قابلیت میتواند برای وبسایت ما که یک فروشگاه اینترنتی محسوب میشود، بسیار مفید باشد.

Infinity Scroll

سعی کردیم برای صفحه اول وبسایت که میتواند مهمترین صفحهٔ آن هم باشد، یک تجربهٔ کاربری روان و مناسب ارائه دهیم و مشابه سایت های دیگر در این حوزه، از infinity scroll برای نمایش همهٔ آگهی ها استفاده کردیم تا کاربر بدون معطلی بتواند آگهی های متعددی را ببیند.

Routing

برای سهولت بیشتر کاربر در استفاده از این وب اپ، route های مختلفی را تعریف کردیم. به اینصورت که تقریبا برای هر اکشن او، یک روت منحصر به فرد ساخته میشود و باعث میشود بتواند بین استیت های مختلفی که در آن بوده به راحتی (مثلا با دکمهٔ بک در موبایل یا دکمه previous مرورگر در دسکتاپ) جا به جا شود و یا حتی لینک یک استیت خاص (برای مثال حالتی که یکسری فیلتر انتخاب شده اند) را برای فرد دیگری ارسال کند و آن فرد هم با ورود به آن لینک، بدون نیاز به انتخاب دوبارهٔ آن فیلتر ها، همان استیت را مشاهده کند.

دیزاین دیتابیس

- کالا (item)

- آیدی
- استتوس معالمه (فعال - رزرو - فروخته شده - مسدود شده)
- ایدی کتگوری
- قیمت
- توضیحات
- ایدی کاربری که آگهی کرده (فروشنده)
- ایدی کاربری که خریداری/رزرو
- عنوان
- مسدود شده/نشده

- تصاویر (image)

- ایدی
- فایل تصویر

- بنرها (banner)

- ایدی کالا
- ترتیب
- ایدی تصویر

- کتگوری

- ایدی
- عنوان

- درخواست خرید

- ایدی کالا
- ایدی کاربر درخواست دهنده

- توضیحات خرید

- یوزر

- ایدی
- نام و نام خانوادگی
- شماره تلفن
- آدرس
- ایمیل
- مسدود شده/نشده

- گزارش تخلف کاربر

- ایدی
- نوع تخلف
- آیدی کاربر گزارش شده
- استتوس (تایید شده، رد شده، در حال بررسی)

- گزارش تخلف کالا

- ایدی
- نوع تخلف
- آیدی کالا
- استتوس (تایید شده، رد شده، در حال بررسی)