سند معماری سامانه یکپارچه‌سازی آگهی‌های استخدامی (سیا)

آرمین مظفری

۴۰۲۴۴۳۱۶۷

مجتبی خانلوشندی

۴۰۲۴۴۳۰۵۵

ورودی‌های ارشد نرم‌افزار ۱۴۰۲

پاییز ۱۴۰۲

فهرست مطالب

۱- فضای مسئله

۱-۱- مقدمه

این بخش مقدمه سند معماری است که شامل اطلاعاتی مانند توصیف مختصر نرم‌افزار، scope نرم‌افزار، محدودیت‌ها و مراجع است و اطلاعات اولیه و مفیدی در مورد نرم‌افزار، به ما می‌دهد. از آنجایی که سند SRS نیز تولید شده‌است، هدف این بخش تنها این است که توضیحات مختصری در مورد نیازمندی‌ها بدهد و توضیحات تکمیلی در سندهای ذکر شده قابل رویت هستند.

۱-۱-۱- توصیف مختصر

این پروژه با هدف ایجاد یک سامانه جامع برای بهبود فرآیند جستجوی کار و ایجاد بستری جامع و یکپارچه برای پیدا کردن آگهی‌های استخدامی از طریق اپلیکیشن موبایل یا وب‌سایت، تعریف شده است. کاربران در این اپلیکیشن ثبت‌نام می‌کنند و یک سری معیار برای آگهی‌های استخدامی مورد نظر خود مشخص می‌کنند. آن‌ها سپس منابع مورد نیاز خود (سایت‌های آگهی‌های استخدامی) را انتخاب کرده و در نهایت، درصورت خرید اشتراک، راه‌های ارتباطی با خود را مشخص کرده تا در صورت ایجاد آگهی با ویژگی‌های موردنظر، به او خبر داده شود. این پروژه بهبود دسترسی به آگهی‌های استخدامی را فراهم می‌کند و به عنوان یک رویکرد جایگزین برای جستجوی آگهی‌های استخدامی و کاریابی مطرح می‌شود.

۱-۱-۲- محدوده (Scope)

این پروژه به تعریف یک سیستم جمع‌آوری و دسته‌بندی آگهی‌های استخدامی از منابع مختلف می‌پردازد. این سیستم امکان جمع‌آوری و نمایش آگهی‌های استخدامی از وبسایت‌ها، رسانه‌ها و منابع دیگر را فراهم می‌کند. کاربران قادرند با استفاده از این سیستم آگهی‌ها را بر اساس معیارهای مختلفی مانند مهارت‌ها، مکان، هزینه و سابقه کاری جستجو و فیلتر کنند. همچنین، امکان جستجوی پیشرفته و سفارشی‌سازی برای کاربران در نظر گرفته شده است. کاربران می‌توانند اشتراک ویژه خریداری کنند تا اعلان‌های استخدامی جدید را دریافت کنند. سیستم از لاگ‌های جستجو و تنظیمات شخصی‌سازی کاربران نیز پشتیبانی می‌کند. محدوده این پروژه به توسعه و بهبود این قابلیت‌ها و ارائه یک سامانه جامع برای جستجو و انتشار آگهی‌های استخدامی ادامه دارد و می‌تواند در آینده با توجه به نیازمندی‌ها و بازخوردهای مشتریان گسترش یابد.

۱-۱-۳- فهرست تعاریف، اختصارات و واژه‌نامه (Glossary)

اصطلاح	تعریف
اپلیکیشن موبایل	**برنامه کاربردی تلفن همراه که کاربران برای درخواست تاکسی، ردیابی موقعیت مکانی راننده و پرداخت هزینه سفر خود استفاده می کنند.**
آگهی‌های استخدامی	**فرصت‌های شغلی که در سامانه جمع‌آوری و نمایش می‌شوند.**
جستجوی پیشرفته	**امکان جستجو بر اساس معیارهای دقیقتر و پیچیده‌تر برای یافتن آگهی‌ها.**
فیلترینگ آگهی‌ها	**ترتیب‌دهی و نمایش آگهی‌ها بر اساس معیارهای مشخص مانند مهارت‌ها و مکان.**
سفارشی‌سازی	**امکان تعیین معیارهای خاص برای جستجو توسط کاربران.**
اعلان‌های استخدامی	**اطلاعیه‌ها درباره آگهی‌های استخدامی جدید که به کاربران ارسال می‌شوند.**
مدیریت لاگ‌ها	**فعالیت‌ها و عملکرد کاربران در سامانه را ثبت و نظارت می‌کند.**
معیارهای مختلف	**مواردی که می‌تواند برای دسته‌بندی و انتخاب آگهی‌ها استفاده شود.**

اشتراک ویژه	**خدمتی که به کاربران ارائه می‌شود تا اعلان‌های استخدامی جدید را دریافت کنند.**
محدوده	**محدوده‌ی عملیاتی سامانه جمع‌آوری و دسته‌بندی آگهی‌های استخدامی.**
Clean Architecture	**یک الگوی طراحی نرم‌افزار که اصول معماری نرم‌افزار را تعیین می‌کند.**
درگاه پرداخت	**پلتفرمی که امکان پردازش پرداختهای مسافران را فراهم می کند.**
REST API	**معماری برنامه‌نویسی که از HTTP به عنوان پروتکل ارتباطی برای انتقال داده استفاده می‌کند.**
Flutter	**یک فریم‌ورک توسعه برنامه‌های تحت پلتفرم‌های مختلف از جمله وب و اندروید است.**
Spring Boot	**یک فریم‌ورک توسعه برنامه‌های جاوا برای سرورهای وب.**
PostgreSQL	**یک سیستم مدیریت پایگاه داده رابطه‌ای (RDBMS) با قابلیت‌های پیشرفته.**
CI/CD	**Continuous Integration و Continuous Delivery، رویکردهای توسعه مداوم برنامه‌نویسی.**
Microservices	**معماری نرم‌افزاری که سیستم را به اجزای کوچکتر و مستقل تقسیم می‌کند.**
Use Case	**یک توصیف دقیق از عملکرد یک سامانه و تعاملات بین کاربر و سامانه.**
BPMN	**Business Process Model and Notation، استاندارد توصیف فرآیندهای کسب‌وکار.**
TDD	**Test-Driven Development، رویکرد توسعه که با تست‌نویسی مداوم شروع می‌شود.**
تجربه کاربری	**کیفیت تجربه کاربری هنگام استفاده از سامانه یکپارچه‌سازی آگهی‌های استخدامی (سیا)**
خدمات مشتری	**ارائه خدمات پشتیبانی به مشتریان برای پاسخگویی به سوالات، حل مشکلات و ارائه کمک.**
توسعه محصول	**بهبود و گسترش ویژگی‌های سامانه یکپارچه‌سازی آگهی‌های استخدامی (سیا)**
مدیریت پروژه	**برنامه ریزی، سازماندهی و اجرای پروژه‌های مربوط به سامانه یکپارچه‌سازی آگهی‌های استخدامی (سیا)**

۱-۱-۴- اهداف معماری

معیار طراحی	تعریف
قابلیت مقیاس‌پذیری **(Scalability)**	**توانایی سیستم برای سازگار شدن با افزایش تعداد کاربران و جستجوها بدون افت کارایی.**
کارایی **(Performance)**	**پاسخ‌گویی سریع سیستم به درخواست‌های کاربران (جستجو و ...) و فراهم کردن یک تجربه کاربری روان و بدون تأخیر.**
امنیت **(Security)**	**حفاظت از داده‌های شخصی کاربران و اطلاعات پرداخت، و مقابله با تهدیدات امنیتی.**
انعطاف‌پذیری **(Flexibility)**	**توانایی سیستم برای تغییر و تطابق با نیازمندی‌های تجاری بدون دگرگونی کلی ساختار.**
یکپارچگی **(Integration)**	**ادغام بهینه با سایر سرویس‌ها و APIها (مانند پرداخت، ایمیل و ...) برای ارتقاء تجربه کاربری و افزایش ارائه خدمات.**
قابلیت اطمینان **(Reliability)**	**سیستم باید قابل اعتماد باشد تا کاربران در هر زمانی که نیاز دارند، بتوانند به خدمات دسترسی پیدا کنند (چون Application است و به سرعت قابل تعویض است)**
دسترس‌پذیری **(Availability)**	**فراهم کردن زمانی بالا برای دسترسی به سرویس در تمام اوقات (چون Application است و به سرعت قابل تعویض است)**
یکنواختی **(Consistency)**	**حفظ یکنواختی داده‌ها و اطلاعات در سراسر سامانه به منظور جلوگیری از تضادها و ناسازگاری‌ها.**
توسعه‌پذیری **(Extensibility)**	**فراهم کردن امکانات برای آسان‌تر شدن تغییرات و اضافه کردن ویژگی‌ها در آینده. زیرا این کسب‌وکار پتانسیل بالایی دارد و می‌تواند به سرعت و به راحتی بزرگ شود.**
تعامل پذیری **(Interoperability)**	**ایجاد یک محیط که بتواند با دیگر سامانه‌ها و اپلیکیشن‌ها به خوبی ارتباط برقرار کند.**

۱-۱-۵- محدودیت‌ها (Constraints)

عنوان محدودیت	توضیحات
محدودیت‌های زمانی	سامانه باید در زمان‌های مشخص و با زمان پاسخ دادن به درخواست‌های کاربران مطابقت داشته باشد.
محدودیت‌های امنیتی	**سامانه باید به محدودیت‌ها و استانداردهای امنیتی پایبند باشد تا اطلاعات کاربران و اطلاعات مالی محافظت شوند.**

محدودیت‌های پایگاه داده	سامانه باید از پایگاه داده‌های مشخص استفاده کند و باید توانایی مقیاس‌پذیری و عملکرد بهینه را داشته باشد.
محدودیت‌های توانایی	**سامانه باید توانایی مدیریت تعداد کاربران و تراکنش‌ها را داشته باشد و از دسترسی به منابع محدود مانند پهنای باند و حافظه صحیح استفاده کند.**
محدودیت‌های مالی	سامانه باید با محدودیت‌های مالی تطابق داشته باشد و باید بودجه مشخصی برای توسعه و پشتیبانی داشته باشد.
محدودیت‌های سخت‌افزاری	**سامانه باید بر روی سخت‌افزارهای مشخصی اجرا شود و از ویژگی‌های سخت‌افزاری مشخصی استفاده کند.**
محدودیت‌های نرم‌افزاری	سامانه باید با محدودیت‌های نرم‌افزاری مشخصی سازگار باشد و از نرم‌افزارهای مشخصی برای عملکرد خود استفاده کند.
محدودیت‌های تطابق با قوانین	**سامانه باید با قوانین و مقررات مربوط به حفاظت از اطلاعات کاربران و موارد مشابه تطابق داشته باشد.**
محدودیت‌های متناسب با منابع	سامانه باید با منابع محدودی که در دسترس است بهینه عمل کند و از آنها بهره‌وری کند.
محدودیت‌های قابلیت استفاده مجدد	**سامانه باید از کدها و ماژول‌های قابل استفاده مجدد استفاده کند تا توسعه و پشتیبانی آسان‌تر شود.**
محدودیت‌های توسعه‌پذیری	سامانه باید قابلیت اضافه کردن و توسعه ویژگی‌ها و تغییرات در آینده را داشته باشد.
محدودیت‌های تعامل پذیری	**سامانه باید با سایر سامانه‌ها و اپلیکیشن‌ها به خوبی ارتباط برقرار کند و از استانداردهای مشخصی پیروی کند.**

۱-۱-۶- استانداردها

عنوان استاندارد	توضیحات
استانداردهای امنیتی	**سامانه باید از استانداردهای امنیتی مانند OWASP Top Ten و ISO/IEC 27001 استفاده کند تا از حفاظت در مقابل حملات امنیتی و نفوذکاری‌ها اطمینان حاصل شود.**
استانداردهای وب	**در صورتی که سامانه از وب برای ارتباط با کاربران استفاده می‌کند، باید استانداردهای مرتبط با وب مانند HTTP/HTTPS و RESTful API را رعایت کند.**
استانداردهای معماری	از استانداردهای معماری نرم‌افزار مانند Clean Architecture یا Microservices استفاده می‌شود تا ساختار سامانه بهینه و قابل توسعه باشد.
استانداردهای داده	**اطلاعات در پایگاه داده‌ها باید با استفاده از استانداردهای مدیریت داده مانند ACID و BASE مدیریت شوند.**

استانداردهای اطلاع‌رسانی	برای ارسال پیام‌ها و اطلاع‌رسانی به کاربران از استانداردهایی مانند SMTP برای ایمیل یا SMS Gateway برای پیامک استفاده می‌شود.
استانداردهای ایمنی و بهداشت محیط کار	**اگر در توسعه سامانه از افرادی استفاده می‌شود، استانداردهای بهداشت محیط کار و ایمنی باید رعایت شود.**
استانداردهای دسترس‌پذیری	سامانه باید به استانداردهای دسترس‌پذیری وب (مانند WCAG) پاسخ دهد تا افراد با معلولیت نیز بتوانند از سامانه استفاده کنند.
استانداردهای پرداخت	**اگر سامانه امکان پرداخت آنلاین را فراهم می‌کند، باید استانداردهای امنیتی پرداخت مانند PCI DSS را رعایت کند.**
استانداردهای متناسب با منابع	سامانه باید منابع سخت‌افزاری و نرم‌افزاری را بهینه مدیریت کند و از استانداردهای مصرف منابع بهره‌برداری کند.
استانداردهای مرتبط با زبان‌ها و فریم‌ورک‌ها	**استفاده از استانداردهای ادغام کد (مانند Git Flow و CI/CD) برای توسعه و ادغام مستمر.**
استانداردهای توسعه و ادغام	سامانه باید قابلیت اضافه کردن و توسعه ویژگی‌ها و تغییرات در آینده را داشته باشد.
استانداردهای ارتباط با سایر سیستم‌ها	**چون سامانه با سایر سرویس‌ها یا APIها ارتباط دارد، استانداردهای ارتباطشان رعایت می‌شوند.**

۱-۱-۷- مراجع

- پی‌دی‌اف دکتر علی‌اکبری درمورد روش نوشتن سند معماری
- سایت C4 Model متعلق به Simon Brown
- سایت https://structurizr.com/dsl برای کشیدن نمودارهای C4
- سایت https://www.infoq.com/articles/minimum-viable-architecture/
- ویدیوی یوتوب Simon Brown در مورد روش C4

۱-۲- نمای موارد کاربرد (مانند System Context روش C4)

۱-۲-۱- توضیحات نما

این نما مانند نمای System Context روش C4 است و ارتباط بازیگران و سیستم‌های خارجی را با سیستم نشان می‌دهد.

۱-۲-۲- کنشگرهای (actors) انسانی و سیستمی

به جای تعریف یک سیستم کلی که تمام کارها را انجام می‌دهد، زیرسیستم‌هایی داریم که هر کدام یک کنشگر (actor) هستند.

1. کاربر عادی
2. کاربر ویژه
3. سیستم یکپارچه
4. اپراتور پیامک
5. سیستم ایمیل
6. سیستم تلفن گویا
7. سیستم پرداخت
8. سیستم تحلیل‌گر

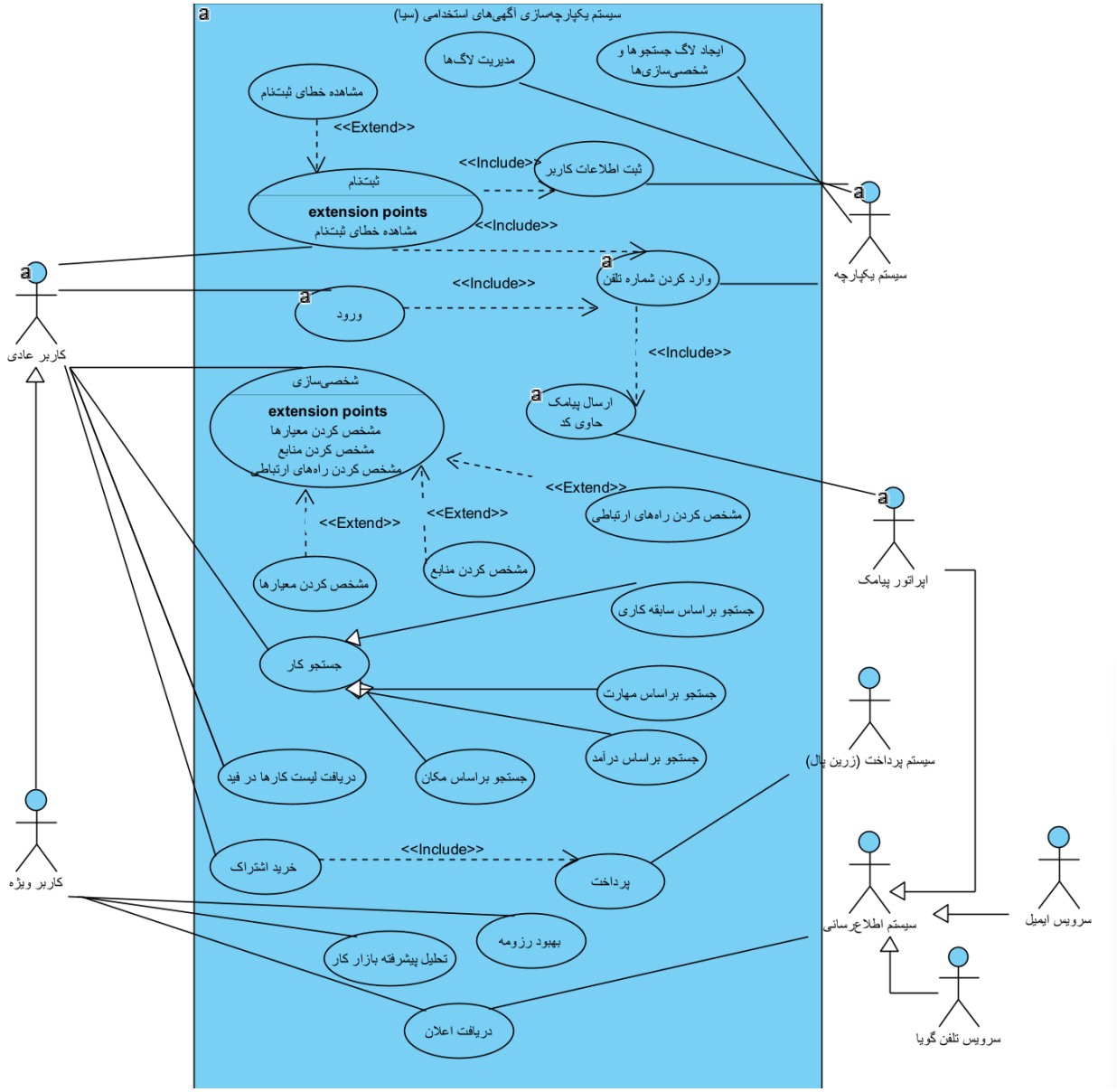
۱-۲-۴- جدول موارد کاربرد (جدول نیازمندی‌های کارکردی)

شناسه	کنشگر اصلی	مورد کاربرد کلان	توضیح

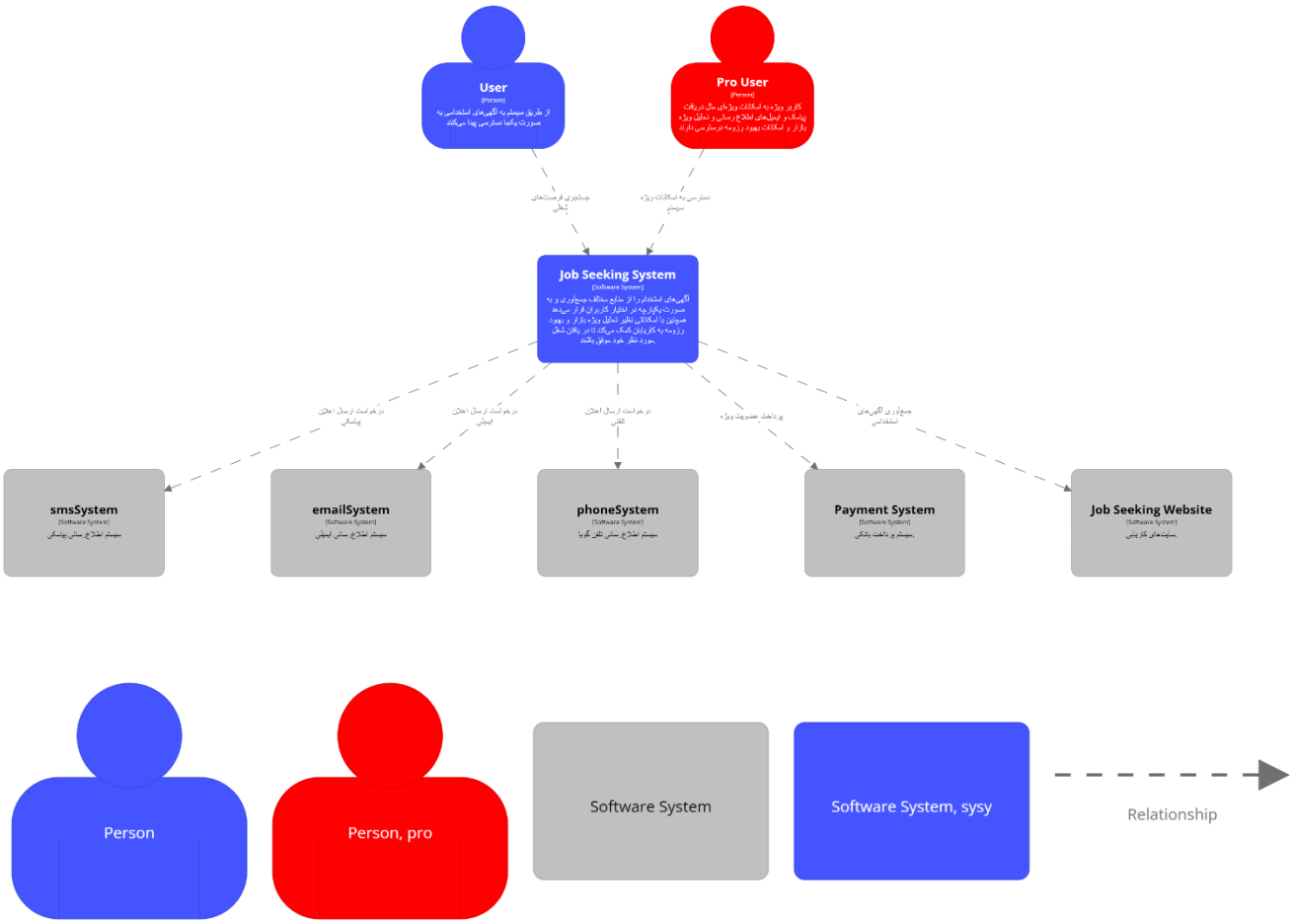
۱	کاربر عادی	ثبت‌نام	کاربر عادی باید بتواند ثبت‌نام کند.
۲	کاربر عادی	ورود	کاربر عادی باید بتواند وارد اکانت خود شود تا از امکانات نرم‌افزار استفاده کند
۳	کاربر عادی	مشخص کردن معیارها	کاربر عادی باید بتواند وارد اکانت خود شود و معیارهای خود برای دریافت آگهی‌های استخدامی را مشخص کند.
۴	کاربر عادی	مشخص کردن منابع	کاربر عادی باید بتواند وارد اکانت خود شود و منابع خود (سایت‌ها/اپلیکیشن‌ها) برای دریافت آگهی‌های استخدامی را مشخص کند.
۵	کاربر عادی	مشخص کردن راه‌های ارتباطی	کاربر عادی باید بتواند وارد اکانت خود شود و راه‌های ارتباط با خود برای را مشخص کند. این عمل با وارد کردن ایمیل، شماره تلفن یا شماره مناسب برای دریافت تماس تلفنی انجام می‌شود.
۶	کاربر عادی	جستجو کار براساس سابقه کاری	کاربر عادی باید بتواند از بین لیست انبوه کارها، براساس فیلتر سابقه‌کاری، جستجو کند و کارهای مناسب خود را مشاهده کند.
۷	کاربر عادی	جستجو کار براساس مهارت	کاربر عادی باید بتواند از بین لیست انبوه کارها، براساس فیلتر مهارت، جستجو کند و کارهای مناسب خود را مشاهده کند.
۸	کاربر عادی	جستجو کار براساس درآمد	کاربر عادی باید بتواند از بین لیست انبوه کارها، براساس فیلتر درآمد، جستجو کند و کارهای مناسب خود را مشاهده کند.
۹	کاربر عادی	جستجو کار براساس مکان	کاربر عادی باید بتواند از بین لیست انبوه کارها، براساس فیلتر مکان، جستجو کند و کارهای مناسب خود را مشاهده کند.
۱۰	کاربر عادی	دریافت لیست کارها در فید	کاربر عادی باید بتواند بعد از ورود به اکانت خود، در قسمت فید، با توجه به معیارهایی که مشخص کرده، لیست کارهای مناسبی را ببینند.
۱۱	کاربر ویژه	دریافت اعلان	کاربر ویژه بعد از ورود به اکانت، خرید اشتراک و مشخص کردن معیارها، منابع و راه‌های ارتباطی، می‌تواند با توجه به اشتراک خریداری

			شده، از طریق راه‌های ارتباطی دلخواه، اعلان مربوط به کارهای موردنظر خود را دریافت کند.
۱۲	کاربر عادی	خرید اشتراک	کاربر عادی با خرید اشتراک می‌تواند به کاربر ویژه تبدیل شده و از امکانات ویژه‌ای که سامانه به این کاربران می‌دهد، بهره‌مند شود.
۱۳	سیستم پرداخت	پرداخت	پرداخت‌ها از طریق درگاه‌های پرداخت و API گرفته شده از سیستم‌های پرداخت مانند زرین پال باید انجام شود.
۱۴	سیستم تحلیل‌گر	بهبود رزومه	سیستم تحلیل‌گر با استفاده از اطلاعات کارها و موقعیت‌های شغلی و رزومه‌هایی که موفق به دریافت این موقعیت‌ها شدند، سعی در بهبود رزومه کاربران ویژه می‌کند و مواردی که باید به رزومه فرد اضافه شوند یا از رزومه ایشان حذف شوند را بیان می‌کند.
۱۵	سیستم تحلیل‌گر	تحلیل پیشرفته بازار کار	سیستم تحلیل‌گر (که درون سیستم یکپارچه قرار دارد)، با استفاده از اطلاعاتی که دریافت می‌کند و با استفاده از الگوریتم‌های یادگیری ماشین، در بخش تحلیل پیشرفته بازار موجود در پنل کاربر، با نمایش نمودارها و تحلیل‌ها، موقعیت‌ها و مهارت‌های شغلی trend را مشخص کرده و جهت حرکت بازار را بدست می‌آورد.
۱۶	سیستم یکپارچه	ایجاد لاگ جستجوها و شخصی‌سازی‌ها	برای بهبود پیشنهاددهی و تجربه کاربری، جستجوها و شخصی‌سازی‌ها باید ثبت شوند تا بشود با تحلیل آن‌ها، تجربه کاربری را بهتر کرد و پیشنهادهای بهتری به کاربران داد.
۱۷	سیستم یکپارچه	مدیریت لاگ‌ها	برای بهبود پیشنهاددهی، بعد از ثبت لاگ‌ها، باید این لاگ‌ها مدیریت شده و تحلیل شده و از اطلاعات بدست آمده از آن‌ها، برای بهبود تجربه کاربری و پیشنهادهای کاری در فید، استفاده شود.

۱-۲-۵- نمودار موارد کاربرد



۱-۲-۶- نمودار System Context



۱-۲-۷- اسکریپت نمودار System Context

برای تولید نمودار بالا، اسکریپت زیر ایجاد شده است که در سایت https://structurizr.com/dsl، این اسکریپت تبدیل به نمودار بالا می‌شود.

```
workspace {

  model {
        group "tp"{
```

```
            paymentService = softwareSystem " درگاه پرداخت برای خرید
اشتراک ویژه" {
                tags "tp"
            }
            jobWebsites = softwareSystem "سایت‌های کاریابی" {
                tags "tp"
            }

            smsSystem = softwareSystem  "سیستم اطلاع‌رسانی پیامکی" {
                tags "tp"
            }
            emailSystem = softwareSystem "سیستم اطلاع‌رسانی ایمیلی" {
                tags "tp"
            }
            phoneSystem = softwareSystem  "سیستم اطلاع‌رسانی تلفن گویا" {
                tags "tp"
            }
        }
        user = person "کاربر"


        system = softwareSystem "سیستم یکپارچه‌سازی آگهی‌های استخدامی" {
            tags "sysy"
            webApp = container "برنامه وب"{
                tags "Web Browser"
            }

            mobileApp = container "برنامه موبایل" {
                tags "Mobile App"
                Entity = component "Entities" "Core business logic and
models" {
                    tags = "Entity"
                }
                UseCase = component "UseCases" "Application-specific
business rules and logic" {
                    tags = "UseCase"
                }
                Repository = component "Repositories" "Abstract data
access, clean API for data operations" {
                    tags = "Repository"
                }
                DataSource = component "DataSources" "Concrete data
access methods" {
                    tags = "DataSource"
                }
                Framework = component "FrameworksAndDrivers" "External
interfaces and frameworks" {
                    tags = "Framework"
                }
                Presentation = component "Presentation" "UI layer,
responsible for displaying information and capturing user input" {
                    tags = "Presentation"
                }
```

```
                Controller = component "ControllersPresenters"
"Intermediaries between UI and Use Cases" {
                    tags = "Controller"
                }
                DependencyInjection = component "DependencyInjection"
"Handle injection of dependencies into various layers" {
                    tags = "DependencyInjection"
                }
                Model = component "Models" "Hold data for UI
presentation" {
                    tags = "Model"
                }
                ExternalInterface = component "ExternalInterfaces"
"Implement platform-specific code or external services" {
                    tags = "ExternalInterface"
                }
                Test = component "Test" "Unit tests for Use Cases,
Repositories, and Entities" {
                    tags = "Test"
                }
            }
            SPA = container "Single Page Application"
            API = container "API" {
                LoginController = component "LoginController"  {
                    tags = "loginController"
                }
                ResetPassword = component "ResetPassword" "Handles reset
password" {
                    tags = "resetPassword"
                }
                AuthenticationService = component
"AuthenticationService"  {
                    tags = "authService"
                }
                SecurityComponent = component "SecurityComponent" {
                    tags = "securityComponent"
                }
                PaymentComponent = component "PaymentComponent"  {
                    tags = "paymentComponent"
                }
            }
            notification = container "اطلاع‌رسانی"
            analysis = container "سیستم تحلیلگر"
            DB = container "پایگاه داده اصلی"{
                tags "db"
            }
            mongoDB = container "پایگاه داده آگهی‌ها" "mongoDB"{
                tags "db"
            }
            logDB = container "پایگاه داده لاگ‌ها" {
                tags "db"
            }
            redis = container "redis"{
                tags "db"
```

```
            }

            crawler = container "خزشگر"
        }
        user -> system "جستجوی فرصت‌های شغلی"

        system -> paymentService " از طریق این سیستم به کاربران
اطلاع‌رسانی می‌شود"
        system -> jobWebsites " آگهی‌های استخدامی از این سایت‌ها استخراج
می‌شود"
        system -> emailSystem "درخواست ارسال اعلان ایمیلی"
        system -> smsSystem "درخواست ارسال اعلان پیامکی"
        system -> phoneSystem "درخواست ارسال اعلان تلفنی"

        user -> webApp " از طریق مرورگر به سیستم وصل شده و از آن استفاده
می‌کند"
        user -> mobileApp " از طریق برنامه موبایل به سیستم وصل شده و از
آن استفاده می‌کند"

        webApp -> SPA "از طریق این دروازه به سیستم وصل می‌شود"
        SPA -> API "فراخوانی API"

        API -> notification "درخواست اطلاع‌رسانی"
        API -> analysis "تحلیل اطلاعات"
        API -> DB "ذخیره/بازیابی اطلاعات کاربران و معیارها"
        API -> logDB "ثبت وقایع"
        API -> redis "cache"
        API -> mongoDB "کوئری و سرچ"

        analysis -> logDB "تحلیل لاگ‌ها"

        crawler -> mongoDB "ذخیره آگهی‌های استخراج کرده"

        mobileApp -> API "از طریق این دروازه به سیستم وصل می‌شود"

        UseCase -> Entity
        UseCase -> Repository
        Repository -> DataSource
        DataSource -> Framework
        Presentation -> Controller
        Controller -> UseCase
        DependencyInjection -> UseCase
        Presentation -> Model
        ExternalInterface -> DataSource
        Test  -> UseCase

        notification -> phoneSystem
        notification -> emailSystem
        notification -> smsSystem
        API -> paymentService
        crawler -> jobWebsites

        SPA -> loginController
        mobileApp -> loginController
```

```
        loginController -> AuthenticationService
        resetPassword -> AuthenticationService
        SPA -> resetPassword
        mobileApp -> resetPassword
        resetPassword -> securityComponent
        SPA -> paymentComponent
        mobileApp -> paymentComponent



        Live = deploymentEnvironment "Live" {
                    deploymentNode "User Desktop"{
                        deploymentNode "Browser"{
                            containerInstance webApp
                        }
                    }

                    deploymentNode "API" {
                      containerInstance API
                    }
                    deploymentNode "server - 2" {
                      containerInstance SPA
                    }
                    deploymentNode "server - 1"{
                        containerInstance analysis
                        containerInstance crawler
                    }

                    deploymentNode "server - 3"{
                        containerInstance DB
                    }
                    deploymentNode "server - 4"{
                        containerInstance logDB
                    }
                    deploymentNode "redis"{
                            containerInstance redis
                    }
                    deploymentNode "user mobile"{
                        deploymentNode "Android" "Android"{
                            containerInstance mobileApp
                            description "Android"
                        }
                        deploymentNode "ios" "ios"{
                            containerInstance mobileApp
                            description "ios"
                        }

                    }


        }


    }
```

```
views {
    systemlandscape "SystemLandscape" {
        include *
        autoLayout
    }

    systemContext "system" {
        include *
        autoLayout lr
    }
    container system {
        include *
        autoLayout lr
    }

    component "mobileApp" {
        include *
        autoLayout
    }
    component "API" {
    include *
    autoLayout

    }

    deployment * Live {
        include *
        autoLayout
    }

    dynamic "system"{
        user -> webApp "enter terms for search"
        webApp -> SPA "sends request to SPA"
        SPA -> API "sends request to API"
        API -> DB "send a query to data base"
        DB -> API "responses the query"
        API -> SPA "response to query"
        SPA -> webApp "gets respond"
        webApp -> user "shows jobs"


        autoLayout

    }
    styles {
    element "tp"{
        background #a6a6a6
    }
    element "Element"{
        color #ffffff
        background #4455ff
    }
```

```
            element "Person"{
                shape person
                background #a6a6a6
            }
            element "sysy"{
                color #ffffff
                background #4455ff
            }
            element "db"{
                shape cylinder


            }
            element "Web Browser" {
                shape WebBrowser
            }
            element "Mobile App" {
                shape MobileDeviceLandscape
            }
            element "Deployment"{
                color #000000
            }

        }

    }

}
```

۱-۳- ویژگی‌های کیفی (Quality Attributes)

۱-۳-۱- توضیحات ویژگی‌های کیفی

ویژگی‌های کیفی **قابل‌اندازه‌گیری** مطرح برای این پروژه در این قسمت آمده‌اند. این ویژگی‌های کیفی در این قسمت به صورت جدولی آورده شده‌اند و در ادامه سند، این ویژگی‌ها به صورت سناریوها بیان شده‌اند.

۱-۳-۲- جدول ویژگی‌های کیفی

ویژگی کیفیت	معیار	دامنه بهینه

کارایی	زمان پاسخ‌دهی	کمتر از ۲ ثانیه برای عملیات‌های حیاتی
قابلیت مقیاس‌پذیری	تعداد کاربران و تراکنش‌های همزمان	حداقل ۱۰۰۰۰ کاربر و تراکنش همزمان
یکپارچگی	تعداد APIهای متصل به سیستم	پشتیبانی از ۵+ سیستم خارجی
قابلیت استفاده	نرخ خطا	کمتر از 1%
قابلیت استفاده	زمان یادگیری	بیش از 90٪ یادگیری بدون کمک و با استفاده از رابط‌کاربری
قابلیت استفاده	نرخ رضایت‌مندی کاربر	میانگین نرخ رضایت‌مندی کاربر بالای ۸۰٪
قابلیت اطمینان	میانگین زمان بین شکست‌ها (MTBF)	حداقل ۱۰۰۰ ساعت
دسترس‌پذیری	میانگین زمان بازیابی (MTTR)	کمتر از یک ساعت
دسترس‌پذیری	درصد زمان بالا بودن سیستم	بالا بودن سیستم حداقل 99.9٪ (“سه نه”)، ایده‌آل 99.999٪ (“پنج نه”)
یکنواختی (Integrity)	تعداد تناقض‌های داده‌ای شناسایی شده	۰ تناقض در ماه
کارایی	پهنای باند	پهنای باند سیستم برای پاسخ به هزاران درخواست در ثانیه
کارایی	استفاده از منابع	استفاده بهینه از منابع
قابلیت مقیاس‌پذیری	توانایی افزایش ظرفیت کاربران و تراکنش‌ها بدون کاهش عملکرد	سیستم به صورت افقی/عمودی بدون تغییرات معنادار، حفظ شاخص‌های عملکرد

امنیت	تعداد نقض‌های امنیتی ایجاد شده	حداکثر ۲ نقض در ماه
امنیت	گزارش‌های حوادث	صفر حادثه با شدت بالا
امنیت	نتایج آزمون نفوذ	مقاومت موفقیت‌آمیز در برابر آزمون‌های نفوذ سالانه
امنیت	بررسی‌های انطباق	انطباق کامل با GDPR و سایر مقررات ذیربط
قابلیت دسترسی	امکان استفاده آسان برای افراد با نیازهای خاص	پشتیبانی کامل از استانداردهایWCAG 2.0
توسعه‌پذیری	زمان موردنیاز برای اضافه کردن ویژگی‌ها	حداکثر ۲ هفته برای ویژگی‌های متوسط
قابلیت نگهداری	زمان لازم برای پیاده‌سازی تغییرات	پیاده‌سازی تغییرات در طی چند ساعت تا چند روز
قابلیت نگهداری	تکرار نیاز به نگهداری	نیاز به نگهداری محتوایی نیم سال (شش ماه) یا کمتر
قابلیت حمل	تعداد پلتفرم‌های پشتیبانی شده	پشتیبانی از تمام سیستم‌های عامل موبایل و مرورگرهای اصلی
قابلیت حمل	زمان لازم برای تطبیق با پلتفرم‌های جدید	تطبیق با پلتفرم‌های جدید در عرض یک چرخه توسعه
قابلیت هم‌کاری	تعداد سیستم‌های یکپارچه‌شده	ادغام بدون مشکل با درگاه‌های پرداخت وAPI های اطلاع‌رسانی
قابلیت هم‌کاری	سهولت در افزودن یکپارچه‌سازی‌های جدید	قابلیت افزودن یکپارچه‌سازی‌های جدید در طول هفته‌ها
انطباق	پایبندی به قوانین	پایبندی کامل بدون تخلفات

بومی‌سازی	تعداد زبان‌ها و ویژگی‌های محلی پشتیبانی‌شده	پشتیبانی از چندین زبان و ویژگی‌های محلی با نرخ‌های رضایتمندی بالا در ناحیه‌های بومی
بازیابی از فاجعه	هدف زمان بازیابی(RTO)	RTO کمتر از چند ساعت
بازیابی از فاجعه	هدف نقطه‌ی بازیابی(RPO)	RPO بدون از دست دادن داده‌ها یا حداقل از دست دادن قابل قبول برای کسب‌وکار
کارایی	استفاده از داده‌ها برای عملیات معمول	مصرف داده‌ها کمتر از چند مگابایت برای هر Session
بهره‌وری انرژی	مصرف باتری	استفاده از باتری نباید بیش از 5٪ در ساعت به طور متوسط باشد
قابلیت آزمایش	پوشش تست خودکار	پوشش تست بیش از 80%
قابلیت آزمایش	زمان لازم برای اجرای مجموعه تست	اجرای مجموعه تست در چند دقیقه نه ساعت

۱-۳-۳- سناریوهای ویژگی‌های کیفی

ویژگی کیفی	منشا	محرک	محیط	Artifact	پاسخ	اندازه‌گیری پاسخ
کارایی	کاربر	جستجوی آگهی‌های استخدامی	وضعیت عادی	سیستم جستجو در سرور API	سیستم به سرعت نتایج جستجو را بازگرداند.	زمان پاسخ‌گویی کمتر از 2 ثانیه.
امنیت	هکر	تلاش برای نفوذ به سیستم	وضعیت عادی	سرور API	سیستم نفوذ را تشخیص داده، دسترسی را محدود	هیچ داده‌ای درز نمی‌کند و سیستم ظرف مدت 5

					کرده و جلوی آن را می‌گیرد.	دقیقه به حالت عادی برمی‌گردد.
مقیاس‌پذیری	کاربر	افزایش ناگهانی تعداد کاربران و تراکنش‌ها	اوج ترافیک	کل سیستم	سیستم به صورت خودکار منابع را مدیریت می‌کند تا پاسخگوی ترافیک باشد	سیستم بدون افت کارایی تا 10000 کاربر و تراکنش همزمان پاسخگو است
دسترس‌پذیری	خطای سیستمی	خرابی یک سرویس مهم	وضعیت عادی	سرویس مرتبط با خرابی	سیستم به سرعت به یک حالت پشتیبان سوئیچ می‌کند.	سیستم ظرف مدت 30 دقیقه به حالت عادی بازمی‌گردد و در دسترس است
قابلیت استفاده	کاربر یا سیستم	بروز خطا در اپلیکیشن، مثلاً خطا در بارگزاری داده‌ها	وضعیت عادی	کل اپلیکیشن	اپلیکیشن خطا را به کاربر نمایش می‌دهد و راهنمایی‌هایی برای رفع خطا ارائه می‌دهد.	کاربر در کمتر از 30 ثانیه از مشکل آگاه می‌شود و راهنمایی برای حل آن دریافت می‌کند.
قابلیت استفاده	کاربر	استفاده از ویژگی‌های مختلف سیستم	وضعیت عادی	واسط کاربری سیستم	کاربران می‌توانند بدون دشواری و به صورت طبیعی از ویژگی‌ها استفاده کنند.	بیش از 90٪ کاربران می‌توانند وظایف اصلی را بدون نیاز به راهنمایی تکمیل کنند
پاسخ‌دهی	کاربر	تعامل با رابط کاربری، مثلاً لمس دکمه‌ها	وضعیت عادی	واسط کاربری اپلیکیشن	واسط کاربری به تعاملات کاربر فوراً پاسخ می‌دهد.	زمان پاسخگویی کمتر از 1 ثانیه برای هر تعامل.
زیبایی‌شناسی	کاربر	استفاده از اپلیکیشن و مشاهده رابط کاربری	وضعیت عادی	واسط کاربری اپلیکیشن	رابط کاربری زیبا و جذاب است و کاربران از نظر بصری رضایت دارند.	نمره بالای 4.5 از 5 در نظرسنجی‌های مربوط به زیبایی‌شناسی.
انطباق‌پذیری	تغییر در اندازه صفحه نمایش یا جهت دستگاه	تغییر از حالت عمودی به افقی یا بالعکس	وضعیت عادی	واسط کاربری اپلیکیشن	واسط کاربری به طور خودکار با	تطابق واسط کاربری در کمتر از 1 ثانیه پس از تغییر جهت

					تغییرات سازگار می‌شود	
قابلیت دسترسی (Accessibility)	کاربر با نیازهای خاص	استفاده از اپلیکیشن	وضعیت عادی	واسط کاربری اپلیکیشن	اپلیکیشن ویژگی‌هایی مانند پشتیبانی از خواننده‌های صفحه، تنظیمات رنگ و فونت را برای دسترسی آسان‌تر فراهم می‌کند	پشتیبانی کامل از استانداردهای WCAG 2.0 و امتیاز بالا در نظرسنجی‌های کاربری با نیازهای خاص
تست‌پذیری	تیم توسعه	نیاز به اجرای آزمایشات برای تایید عملکرد	محیط توسعه	Source Code	سیستم امکان اجرای تست‌های خودکار را فراهم می‌کند	تمام تست‌ها در کمتر از ۱۰ دقیقه اجرا می‌شوند
قابلیت استفاده	کاربر	کاربر از ویژگی‌های جدید استفاده می‌کند	وضعیت عادی	رابط کاربری	کاربر به راحتی با ویژگی‌ها کار می‌کند و راهنمایی کمی نیاز دارد	کاربران در کمتر از 5 دقیقه می‌توانند ویژگی‌ها را به کار گیرند.
قابلیت یکپارچگی	تیم توسعه	ادغام با سیستم‌های خارجی یا APIها	محیط توسعه / عملیاتی	APIها	سیستم به راحتی با سیستم‌های دیگر یکپارچه می‌شود	ادغام با هر سیستم خارجی در کمتر از 1 هفته
قابلیت استقرار	تیم عملیاتی	نیاز به استقرار یا به‌روزرسانی سیستم	محیط عملیاتی	سرورهای عملیاتی	سیستم به راحتی و بدون اختلال استقرار یا به‌روزرسانی می‌شود	زمان قطعی کمتر از 5 دقیقه در هر استقرار
بهره‌وری انرژی	سیستم	عملیات معلوم سیستم	محیط عملیاتی	سرورها و دستگاه‌های کاربر	سیستم منابع را بهینه استفاده می‌کند و انرژی کمی مصرف می‌کند	مصرف انرژی کمتر از میانگین صنعت برای هر تراکنش
قابلیت تغییر	تیم توسعه	نیاز به تغییر یا اضافه کردن ویژگی‌های جدید	محیط توسعه	Source Code	سیستم به راحتی قابل تغییر و توسعه است	زمان صرف شده برای اضافه کردن ویژگی جدید کمتر از 2 هفته

تنوع‌پذیری	نیاز بازار یا کاربران	درخواست برای تنظیمات سفارشی یا تغییرات	محیط عملیاتی	مؤلفه‌های قابل تنظیم سیستم	سیستم اجازه می‌دهد تنظیمات به راحتی و بدون تغییر در کد اصلی تغییر کنند.	تغییرات سفارشی در کمتر از 1 روز انجام می‌پذیرد
قابلیت حمل	تغییر در پلتفرم‌های سخت‌افزاری یا نرم‌افزاری	نیاز به اجرای سیستم در پلتفرم جدید	محیط عملیاتی	Source Code / Components	سیستم روی پلتفرم جدید بدون تغییرات اساسی اجرا می‌شود	اجرای سیستم در پلتفرم جدید در کمتر از 2 هفته
مقیاس‌پذیری	افزایش تعداد کاربران یا داده‌ها	تقاضا برای منابع بیشتر	محیط عملیاتی	زیرساخت سرور و پایگاه داده	سیستم به صورت خودکار منابع جدید را تخصیص می‌دهد یا به روش‌های دیگری برای مقابله با تقاضای بیشتر پاسخ می‌دهد.	سیستم بدون کاهش کارایی تا 100,000 کاربر و تراکنش همزمان پاسخگو است
قابلیت اطمینان	خطاهای داخلی یا خارجی	خطا یا شکست در سیستم	محیط عملیاتی	کل سیستم	سیستم به صورت خودکار از نقاط شکست جلوگیری کرده یا بازیابی می‌کند	سیستم بیش از 99.9٪ زمان کار می‌کند
قابلیت نگهداری	تیم توسعه یا عملیاتی	نیاز به بروزرسانی یا تعمیر سیستم	حالت توسعه یا عملیاتی	Source Code / Docs	سیستم به راحتی قابل تعمیر و بروزرسانی است	به‌روزرسانی‌ها یا تعمیرات در کمتر از 1 روز انجام می‌پذیرد

۱-۳-۴- تاکتیک‌ها و patternهای استفاده شده

ویژگی کیفی	تاکتیک	توضیحات تاکتیک
دسترس‌پذیری	Redundancy	این تاکتیک را دوست داشتیم انجام بدیم اما نشد (چون منابع محدود داشتیم)، در صورت داشتن چندین سرور، می‌شد اینکار را انجام داد
دسترس‌پذیری	RAID	این تاکتیک به جای تاکتیک بالا استفاده شد و از سرورهایی که روی آن‌ها **RAID10** ست شده بود، برای ذخیره داده استفاده شد. (در ایران سایت‌هایی وجود دارند که سرور با **RAID10** ارائه می‌دهند)
دسترس‌پذیری	Exception Handling	این تاکتیک را در فرانت **SPA**، با استفاده از **hook**های **React** انجام شده است. در بک‌اند، با استفاده از **transaction**ها و جملات **try/catch**، اطمینان حاصل شده است.
دسترس‌پذیری	ELK	این تاکتیک برای مدیریت لاگ‌ها استفاده می‌شود و ما از این تاکتیک به عنوان یکی از **Use Case**های اصلی خودمان استفاده کردیم (برای تشخیص جهت‌گیری بازار و کارجویان). این تاکتیک برای تشخیص زودتر **Fault**ها هم می‌تواند بکار رود.
دسترس‌پذیری	Zabbix	این تاکتیک برای بررسی وضعیت سرورها و تشخیص زودتر مشکلات و **Fault**ها، بسیار مناسب است اما ما نتوانستیم به خوبی از آن بهره ببریم. در برنامه بلندمدت ما آمده است که از این تاکتیک استفاده کنیم.
یکنواختی (Integrity)	Transaction	با استفاده از **Transaction**ها، از یکنواختی داده اطمینان حال شده است.

۲- فضای راه‌حل

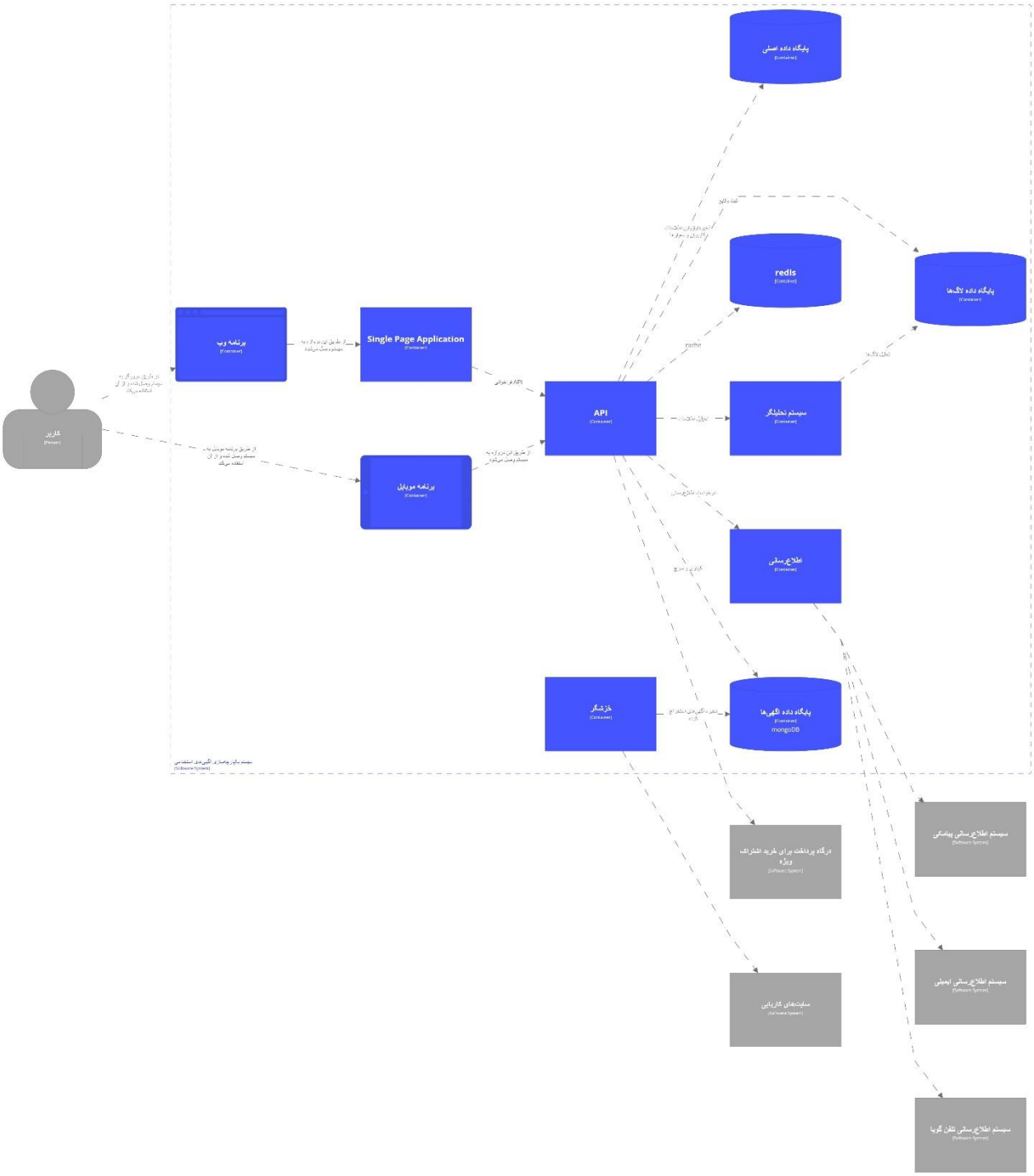
در این فضا، به جای استفاده از نمای منطقی که در روش ابداعی دکتر علی‌اکبری بود، از ترکیب دو روش C4 و روش ابداعی که در تمرین قبلی به آن اشاره کرده بودم، استفاده می‌کنم.

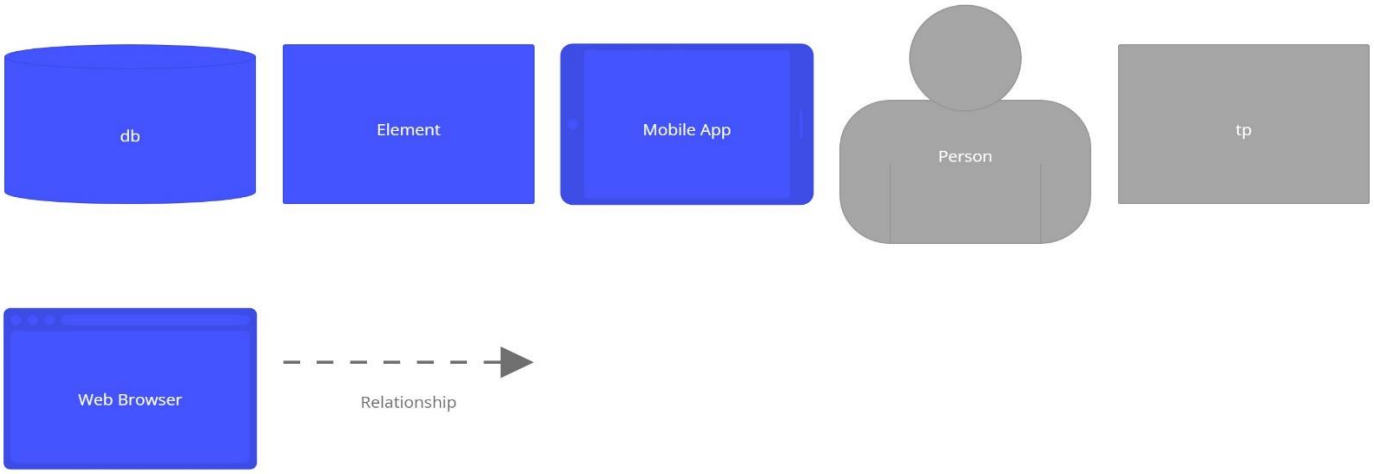
۲-۱- Container Diagram (جایگزین نمای منطقی)

۲-۱-۱- توضیحات نما

این نما، به طور کلی از Container Diagram استفاده می‌کند که به نوعی، نسخه کامل‌تر نمای منطقی است (البته نمای منطقی بخش‌هایی مانند load balancerها و ... را دارد که می‌توانند در deployment diagram قرار بگیرند)

۲-۱-۲- نمودارهای نما



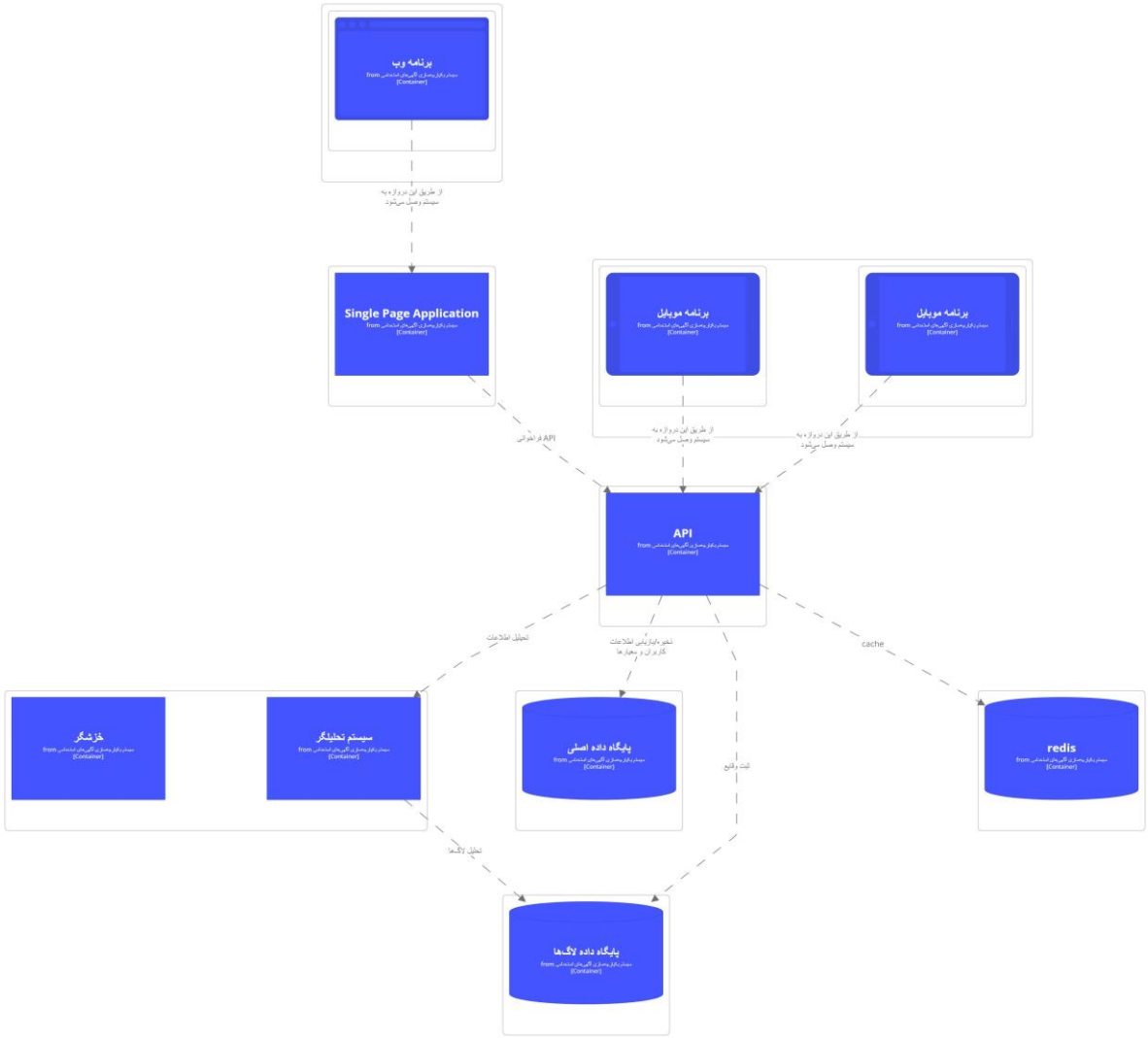


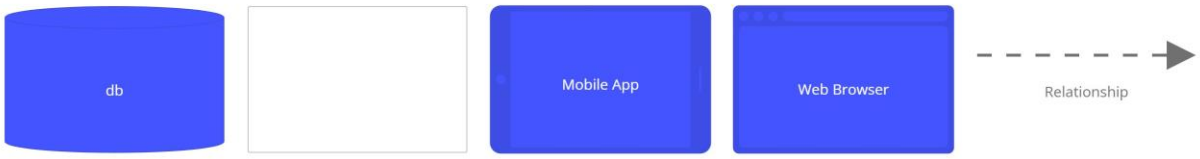
۲-۲- نمای استقرار یا Deployment Diagram

۲-۲-۱- توضیحات نما

این نما، همان Deployment Diagram درون مدل C4 است و برای ترکیب دو روش، از جدولی که در روش ابداعی بود به همراه نمودار استقرار روش C4، استفاده کردیم.

۲-۲-۲- نمودارهای نما





مشکل این نمودار را نفهمیدیم و نتونستیم زیر هر node، اسم آن را بنویسیم.

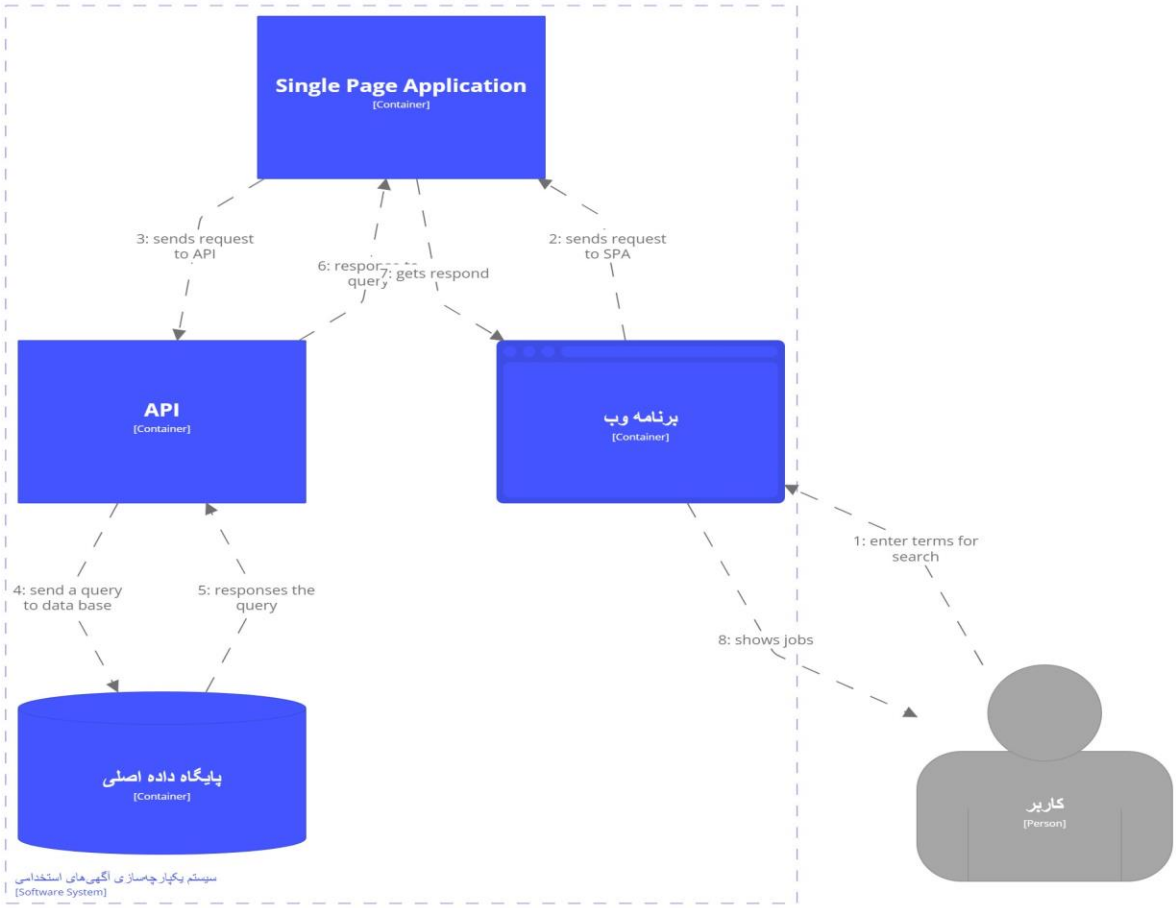
۲-۲-۳- جداول نما

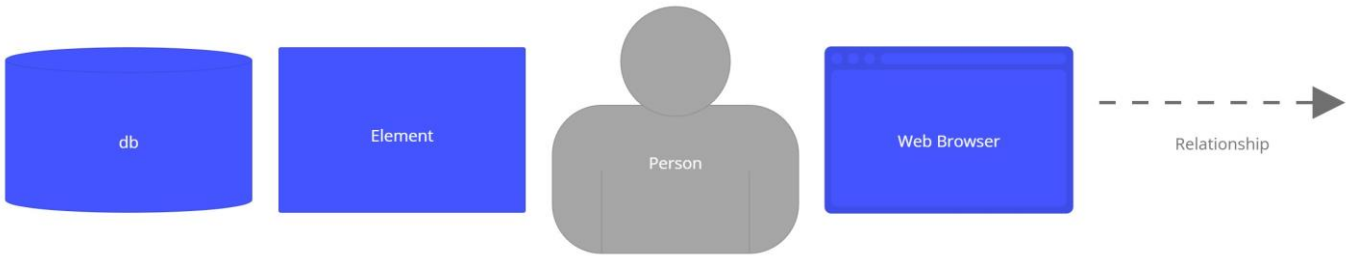
شناسه	مؤلفه‌ها	کارکرد ماشین	نوع	تعداد	تعداد پردازنده	رم	دیسک
User_Interface_Server	Single Page Application	میزبانی رابط کاربری	Virtual Machine	2	4-Core	16 GB	**500 GB SSD**
Authentication_Server	Authentication Service	مدیریت احراز هویت کاربران	Virtual Machine	2	8-Core	32 GB	**1 TB SSD**
API_Server	API Gateway	مدیریت درخواست‌های API	Virtual Machine	4	12-Core	64 GB	**2 TB SSD**
Job_Service_Server	Job Listing Service	میزبانی سرویس آگهی‌های کار	Virtual Machine	4	16-Core	64 GB	**2 TB SSD**
Search_Service_Server	Search Service	میزبانی سرویس جستجو	Virtual Machine	3	8-Core	32 GB	**1 TB SSD**

4 TB SSD	128 GB	16-Core	2	Virtual Machine	تحلیل داده‌ها و بینش‌های کسب و کار	Analytics Service	Analytics_Server
1 TB SSD	16 GB	4-Core	2	Virtual Machine	مدیریت اعلان‌ها	Notification Service	Notification_Server
20 TB SSD	256 GB	24-Core	6	High-Performance VM	میزبانی پایگاه داده	Database	Database_Server
256 GB SSD	32 GB	4-Core	3	Virtual Machine	کش کردن و افزایش کارایی	Redis	Cache_Server

۲-۳- نمای پردازه

۲-۳-۱- نمودارهای نما





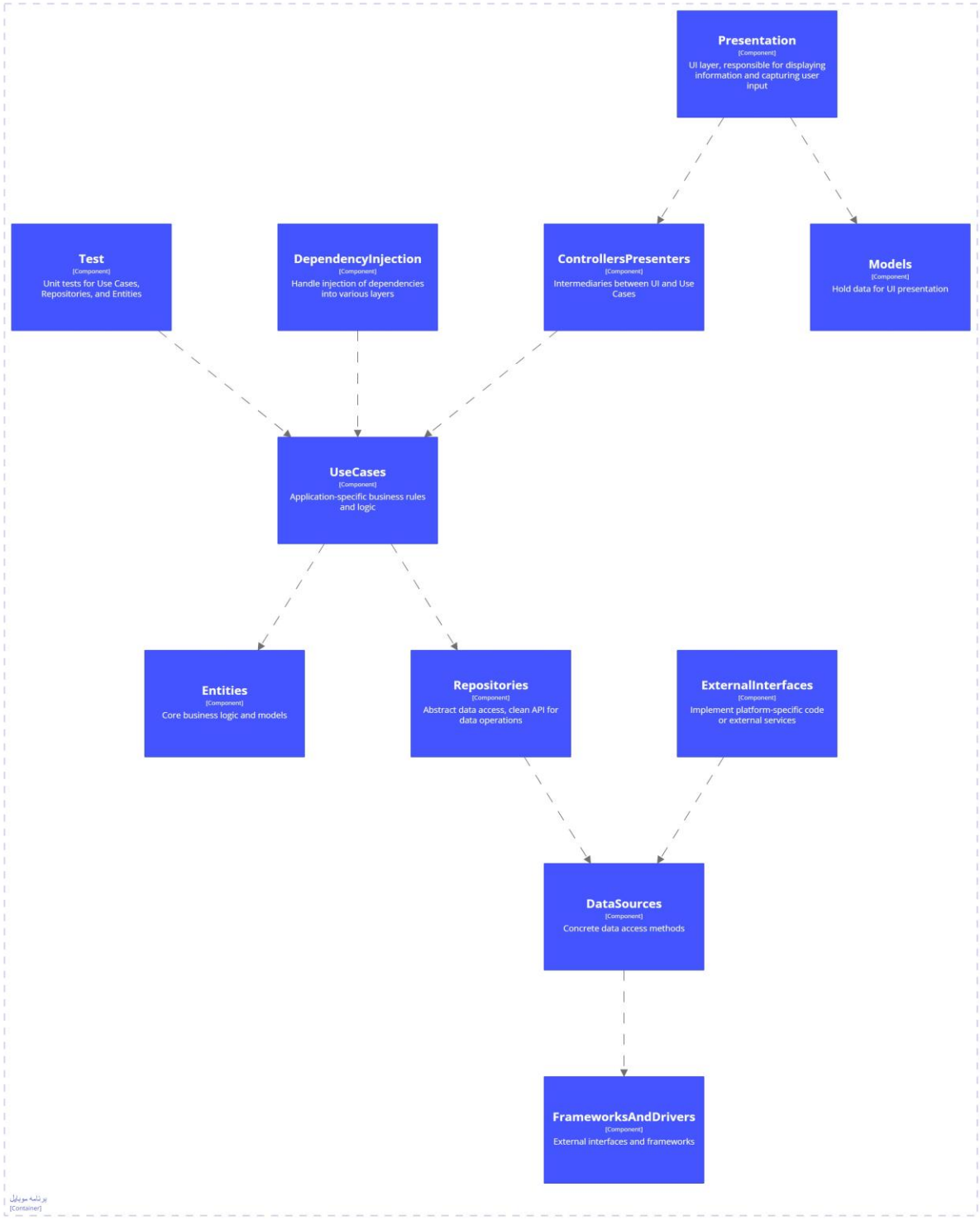
۲-۴- Component Diagram (جایگزین نمای توسعه و پیاده‌سازی)

۲-۴-۱- توضیحات نما

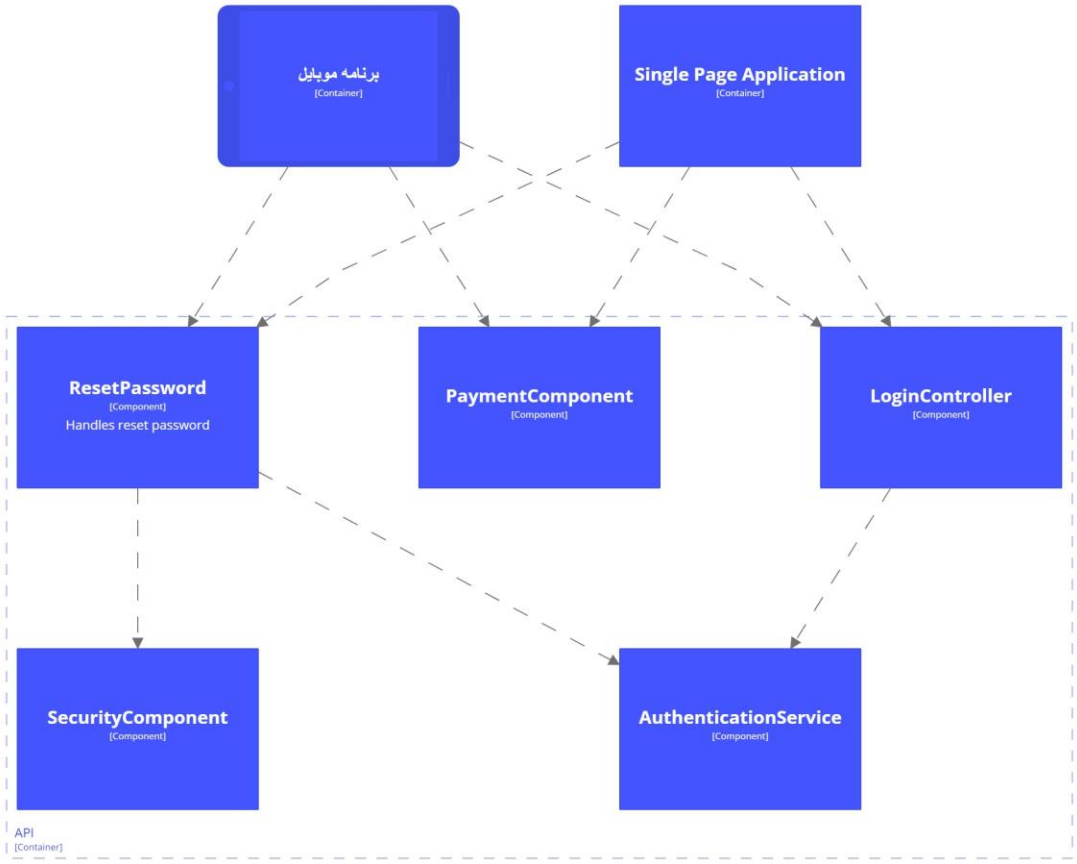
این نما یکی از نماهای اصلی مدل C4 است که تصمیم گرفتیم از آن به جای نماهای توسعه و پیاده‌سازی استفاده کنیم. لازم به ذکر است که از جداول نمای توسعه و پیاده‌سازی برای کامل‌تر شدن این بخش استفاده کردم.

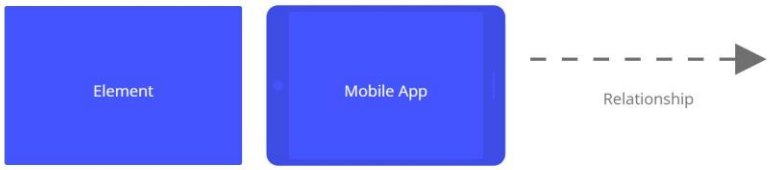
۲-۴-۲- نمودارهای نما

این نمودارها با استفاده از سایت https://structurizr.com/dsl ترسیم شده‌اند و اسکریپت‌های مخصوص آن در بالا آمده است.


Presentation
[Component]
UI layer, responsible for displaying information and capturing user input
Test
[Component]
Unit tests for Use Cases, Repositories, and Entities
DependencyInjection
[Component]
Handle injection of dependencies into various layers
ControllersPresenters
[Component]
Intermediaries between UI and Use Cases
Models
[Component]
Hold data for UI presentation
UseCases
[Component]
Application-specific business rules and logic
Entities
[Component]
Core business logic and models
Repositories
[Component]
Abstract data access, clean API for data operations
ExternalInterfaces
[Component]
Implement platform-specific code or external services
DataSources
[Component]
Concrete data access methods
FrameworksAndDrivers
[Component]
External interfaces and frameworks


Element
Relationship


برنامه موبایل
[Container]
Single Page Application
[Container]
ResetPassword
[Component]
Handles reset password
PaymentComponent
[Component]
LoginController
[Component]
SecurityComponent
[Component]
AuthenticationService
[Component]
API
[Container]



۲-۴-۳- جداول نما

مؤلفه	وابستگی‌ها	توضیحات
Authentication Service	Database, User Service	مدیریت احراز هویت و ایجاد توکن‌ها برای کاربران
User Service	Database	مدیریت اطلاعات کاربران و تعاملات مربوط به پروفایل
Job Listing Service	Database, External Job APIs	جمع‌آوری و دسته‌بندی آگهی‌های کار و ارائه آن‌ها به کاربران
Search Service	Elasticsearch, Job Listing Service	ارائه قابلیت جستجوی پیشرفته بر اساس فیلترهای مختلف
Recommendation Service	User Service, Job Listing Service	ارائه پیشنهادهای شغلی بر اساس تنظیمات و علایق کاربران
Analytics Service	Database, Job Listing Service	تحلیل داده‌ها برای ارائه بینش‌های کسب و کار و بهینه‌سازی سرویس‌ها
Notification Service	User Service	مدیریت ارسال اعلان‌ها به کاربران برای رویدادهای مهم مانند آگهی‌های جدید
API Gateway	All Services	نقطه ورودی مرکزی برای روتینگ درخواست‌ها و مدیریت نگرانی‌های امنیتی

۲-۵- نمای تست

۲-۵-۱- توضیحات نما

Unit Testing:

تست‌های واحد برای هر کامپوننت، مانند Job Listing Service, Search Service و Authentication Service.

ابزارها: JUnit برایJava ، PyTest برای Python

Integration Testing:

بررسی ارتباطات بین Job Listing Service وDatabase ، و بین API Gateway و Authentication Service.

ابزارها : Postman برای API ها، Spring Test برای ادغام Java

Functional Testing:

تست عملکردی برنامه‌های وب و موبایل از دید کاربر.

ابزارها: Selenium برای وب، Appium برای موبایل.

Performance Testing:

اطمینان از کارکرد مناسب سیستم در شرایط بار سنگین، به ویژه برای.Search Service

ابزارها: JMeter، Apache Bench

Security Testing:

تست امنیتی برای بررسی آسیب‌پذیری‌های امنیتی در Authentication Service و Payment Information.

ابزارها: OWASP ZAP، Nessus

Usability Testing:

آزمایش رابط کاربری برای اطمینان از کاربرپسندی در SPA و برنامه‌های موبایل.

ابزارها: UsabilityHub، UserTesting

End-to-End Testing:

شبیه‌سازی سناریوهای کامل کاربر از جستجوی آگهی تا پاسخ به آن.

ابزارها: Cypress، Protractor

Acceptance Testing:

آزمایش کل سیستم برای تأیید پذیرش و اطمینان از مطابقت با نیازهای کاربران و کسب‌وکار.

ابزارها: Cucumber، SpecFlow

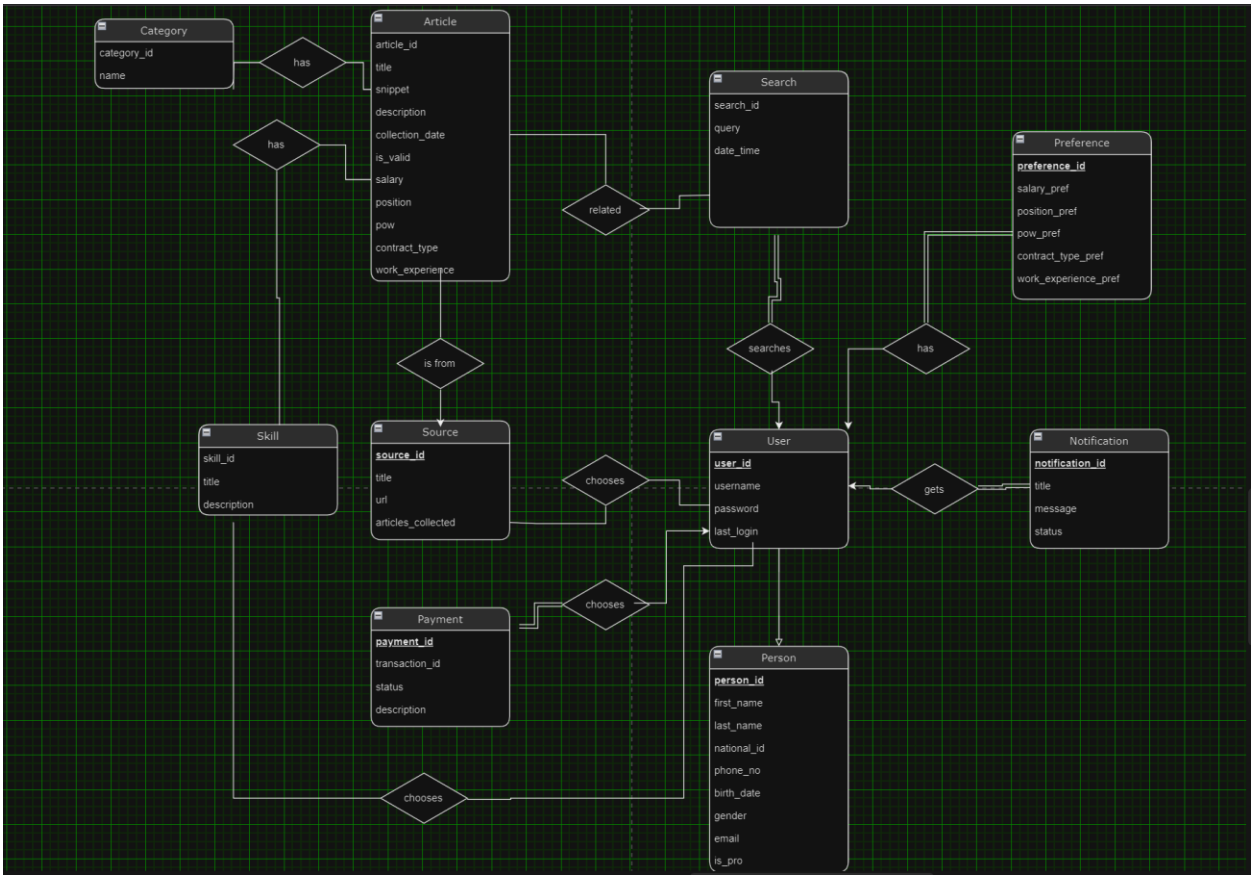
۲-۶- نمای داده (Data View)

۲-۶-۱- توضیحات نما

در این نما، اطلاعات مربوط به داده‌ها و تصمیم‌گیری‌های داده‌ای بیان شده است.

نوع پایگاه‌داده اصلی ما، به صورت رابطه‌ای است که کاربران و اطلاعات اصلی در آن ذخیره می‌شود. ابزار استفاده شده برای پایگاه داده اصلی ما، PostgreSQL است. مقالات و آگهی‌ها در پایگاه داده noSQL ما و با ابزار mongoDB ذخیره می‌شوند. لاگ‌ها و معیارهای ما در استک ELK ذخیره می‌شوند (پایگاه داده لاگ‌ها) و برای عمل Caching، از پایگاه داده in-memory (درون رم) Redis استفاده شده است. در بخش نمودارها، ERD اولیه داده‌ها رسم شده است. توجه شود با اینکه بیشتر قسمت‌های این نمودار از هم جدا و در پایگاه داده‌های متفاوتی ذخیره می‌شوند، برای درک بهتر ارتباط بین آن‌ها، آن‌ها را در یک شکل کنارهم آوردیم.
فرایند پشتیبان‌گیری ما توسط خود DBMSها انجام می‌شود و به صورت یک Scheduled Job در آخر هر روز، در سرورهای Ubuntu انجام می‌شود. این jobها در /var/spool/cron/crontabs/ ذخیره شده و در آخر شب، command مخصوص هر DBMS برای پشتیبان‌گیری اجرا می‌شود.

۲-۶-۲- نمودارهای نما



۲-۷- ابزارها و فناوری‌ها

در این بخش ابزارها و فناوری‌های استفاده شده برای ایجاد مؤلفه‌های این پروژه، معرفی می‌شوند.

۲-۷-۱- توضیحات بخش

مؤلفه	فناوری	جایگاه
برنامه موبایل کاربر	Flutter	**پیاده‌سازی کلاینت گوشی، ANDROID/IOS**
وب اپلیکیشن	Java / Spring MVC	**پیاده‌سازی کلاینت وب**
SPA	ReactJS	**پیاده‌سازی کلاینت وب به صورت PWA**
API	Java/Spring Boot	**پیاده‌سازی بک‌اند اصلی که با باقی قسمت‌ها در ارتباط است**
خزشگر	Python + Selenium	**پیاده‌سازی خزشگر سایت‌های استخدامی**
اطلاع‌رسانی	NodeJS	**فراخوانی APIهای سیستم‌های پیامک، ایمیل و تلفن گویا. به دلیل سبک وزن بودن وظایف از این فریمورک استفاده شده است. یک MQ مانند RabbitMQ یا Kafka برای این بخش درنظر داشتیم که نرسیدیم طراحیش کنیم.**
پایگاه داده آگهی‌ها	MongoDB	**استفاده به عنوان پایگاه داده NoSQL به دلیل اینکه آگهی‌ها ممکن است ساختارمند نباشند و ساختار خاص خود را داشته باشند**

پایگاه داده Caching	Redis	**برای Cache کردن requestها و کاهش زمان پاسخ‌دهی (عمل join جداول بسیار زمان‌بر است و cache کردن از انجام آن جلوگیری می‌کند)**
تحلیل‌گر	Python	**سیستم تحلیل‌گر با تکنیک‌های داده‌کاوی و یادگیری ماشین همراه است، برای همین به زبان پایتون نوشته شده است که کتابخانه‌های مناسبی برای این عمل دارد.**
پایگاه‌داده لاگ‌ها	ELK Stack	**برای ذخیره‌سازی و مدیریت لاگ‌ها، از استک ELK استفاده شده است، که elasticSearch در آن برای ایندکس کردن و جستجو، logstash برای مدیریت و پردازش داده‌ها و kibana برای نمایش گرافیکی و جستجوی گرافیکی لاگ‌ها بکار رفته‌اند. از رابط REST خود elasticSearch برای ارتباط با آن استفاده شده است.**
پایگاه‌داده اصلی	PostgreSQL	**برای ذخیره‌سازی اطلاعات اساسی مانند کاربران، از پایگاه داده رابطه‌ای استفاده شده است.**

۲-۸- شاخه‌های گیت (Git Flow)

در این بخش ابزارها و فناوری‌های استفاده شده برای ایجاد مؤلفه‌های این پروژه، معرفی می‌شوند.

۲-۸-۱- توضیحات بخش

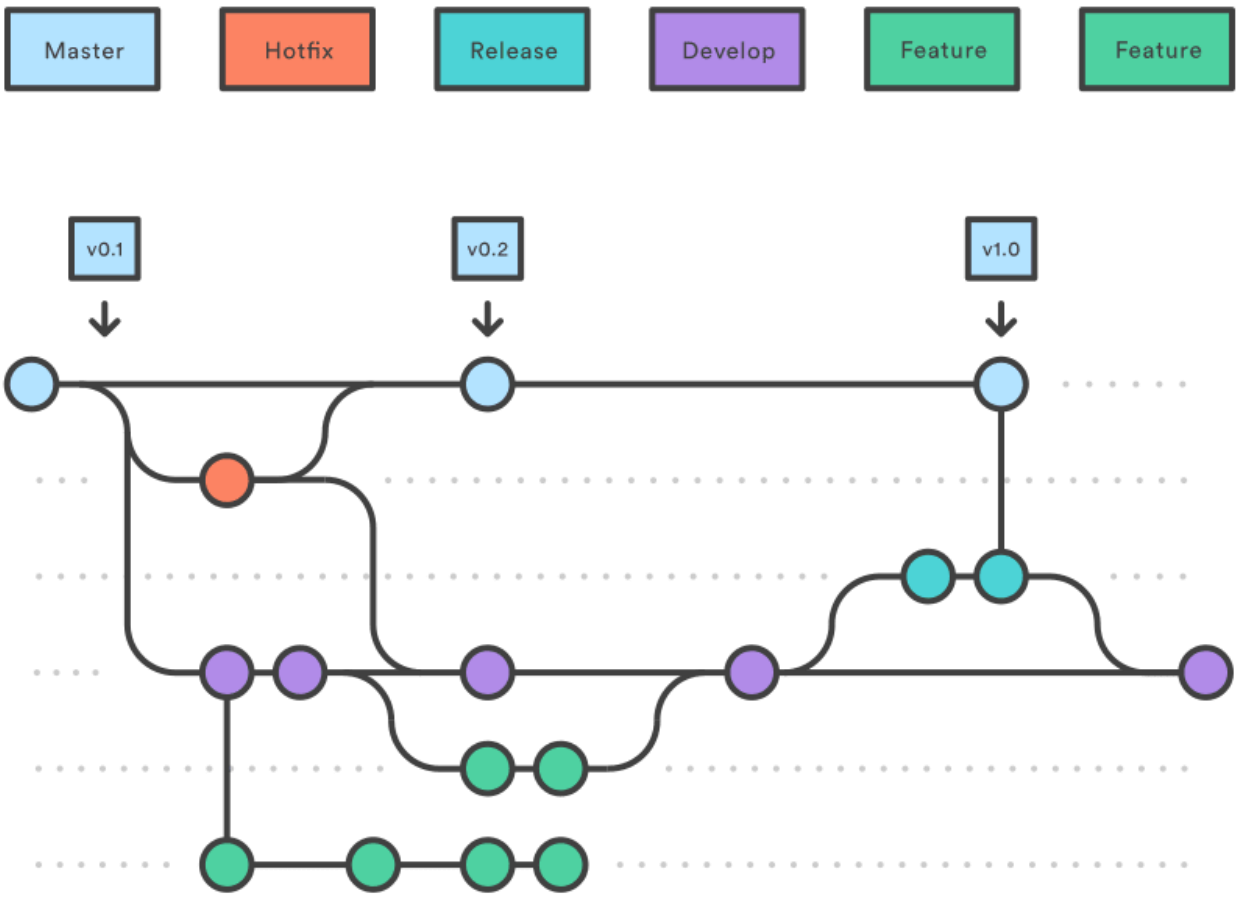
Gitflow شامل شاخه های زیر است:

- شاخه develop: این شاخه از شاخه main ایجاد می شود و به عنوان شاخه ادغام برای ویژگی ها عمل می کند. این شاخه تاریخچه کامل پروژه را شامل می شود.

- شاخه release: این شاخه از شاخه develop ایجاد می شود و برای آماده سازی انتشارات استفاده می شود. در این شاخه فقط اشکال زدایی، تولید مستندات و سایر وظایف مربوط به انتشار انجام می شوند. وقتی انتشار آماده ارسال شود، این شاخه با شاخه main ادغام می شود و با یک شماره نسخه برچسب گذاری می شود. همچنین باید با شاخه develop ادغام شود، که ممکن است از زمان شروع انتشار پیشرفت کرده باشد.

- شاخه feature: این شاخه ها از شاخه develop ایجاد می شوند و برای توسعه ویژگی های جدید استفاده می شوند. وقتی ویژگی کامل شود، آن را با شاخه develop ادغام می کنند. ویژگی ها هرگز مستقیما با شاخه main ارتباط ندارند.

- شاخه hotfix: این شاخه ها برای رفع سریع اشکالات در انتشارات تولید استفاده می شوند. این شاخه ها شبیه به شاخه های انتشار و ویژگی هستند، با این تفاوت که بر اساس شاخه main به جای develop هستند. این تنها شاخه ای است که باید مستقیما از شاخه main شاخه بندی شود. به محض اینکه رفع اشکال کامل شود، باید با هر دو شاخه main و develop (یا شاخه انتشار فعلی) ادغام شود و شاخه main با یک شماره نسخه به روز شود.



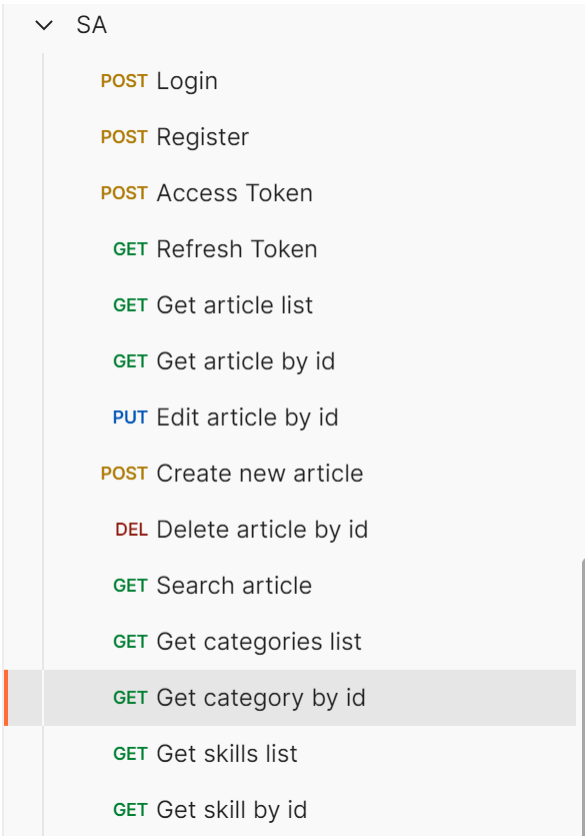
۲-۹- ENDPOINTS

در این بخش اندپوینت (Endpoint)های بک‌اند (API) مشخص می‌شوند.

۲-۹-۱- توضیحات بخش

نام	متد	آدرس
Login	POST	**{{base-url}}/api/v1/authentication/login**
Register	POST	**{{base-url}}/api/v1/authentication/register**
Access Token	POST	**{{base-url}}/api/v1/authentication/token/access**
Refresh Token	GET	**{{base-url}}/api/v1/authentication/token/refresh**
Get Article List	GET	**{{base-url}}/api/v1/articles**
Get Article By ID	GET	**{{base-url}}/api/v1/articles/1**

{{base-url}}/api/v1/articles/1	PUT	Edit Article By Id
{{base-url}}/api/v1/articles	POST	Create New Article
{{base-url}}/api/v1/articles/1	DELETE	Delete Article By ID
{{base-url}}/api/v1/articles/search?q=sl&sal=&pos=&pow=&con=&exp=	GET	Search Article
{{base-url}}/api/v1/category/	GET	Get Categories List
{{base-url}}/api/v1/category/1	GET	Get Category By ID
{{base-url}}/api/v1/skill	GET	Get Skills List
{{base-url}}/api/v1/skill/1	GET	Get Skill By ID



۲-۱۰- مهمترین تصمیمات معماری

۲-۱۰-۱- توضیحات بخش

استفاده از معماری Microservices :

تقسیم بندی منطقی سرویس‌ها برای افزایش قابلیت نگهداری و مقیاس‌پذیری. از معماری modular monolithic به معماری microservice کوچ کردیم.

Single Page Application: برای بهبود تجربه کاربری با بارگذاری سریع‌تر و کمتر.

استفاده از Redis: برای Caching برای کاهش بار بر سرورهای پایگاه داده و بهبود زمان پاسخگویی.

استفاده ازAPI Gateway : مدیریت و ایجاد نقطه مرکزی برای routing درخواست‌ها و مسائل مرتبط با امنیت.

انتخاب استک فناوری:

انتخاب زبان‌های برنامه نویسی مثلاً جاوا اسپرینگ بوت برای API، Node.jsبرای سرویس اعلان.

انتخاب تکنولوژی وب برای کاربران ادمین و مشتریان برای برنامه وب (ReactJS)

انتخاب Flutter برای کلاینت موبایل.

استفاده از درگاهAPI :

استفاده از درگاه API برای ارتباط کلاینت‌ها با بک‌اند و دیتابیس‌ها.

انتخاب دیتابیس و مدل‌سازی داده:

تصمیم‌گیری برای استفاده از PostgreSQL برای داده‌های رابطی و MongoDB برای ذخیره سازی اسناد، پیروی از ACID

یکپارچه‌سازی با سیستم پرداخت:

تصمیم‌گیری برای ادغام درگاه‌های پرداخت شخص ثالث(ZarinPal’s APIs) .

سیستم‌های اعلان و ارتباط:

انتخاب سرویس اطلاع‌رسانی (Node.js) که بتواند چندین کانال ارتباطی (پیام کوتاه، ایمیل، push notification و تلفن گویا) را پشتیبانی کند.

نظارت و ثبت وقایع:

ادغام یک استک ثبت و نظارت مانند Elasticsearch برای ذخیره‌سازی وقایع، Logstash برای پردازش وقایع و Kibana برای گرافیکی کردن عمل نظارت بر عملکرد سیستم.

تدابیر امنیتی:

تعیین استراتژی‌های احراز هویت و مجوز برای امن کردن دسترسی بهAPI ها و استفاده از توکن JWT برای احرازهویت در کلاینت‌ها.

ذخیره توکن در Cookie و ایجاد SSL برای امنیت بیشتر

استفاده از Access Token با ماندگاری ۵ دقیقه و Refresh Token با ماندگاری ۱ روز برای اوج امنیت.

بازیابی پس از فاجعه و پشتیبان‌گیری:

برنامه‌ریزی برای پشتیبان‌گیری‌های منظم، استفاده از RAID10 برای از دست نرفتن Integrity.

استراتژی استقرار:

استفاده از Docker و Kubernetes برای مدیریت استقرار Containerهای مختلفمان (اصطلاح C4).

توسعه برنامه کاربردی موبایل با flutter:

انتخاب استفاده از flutter برای ساخت کلاینت موبایل و build گرفتن دو نسخه IOS و ANDROID.

۲-۱۱- ریسک‌ها و بدهی‌های فنی (Risks and Technical Debts)

در این بخش به ریسک‌ها و بدهی‌های فنی این پروژه می‌پردازیم.

۲-۱۱-۱- توضیحات بخش

ریسک‌ها:

تغییرات مداوم در منابع آگهی:

ممکن است از منابع خارجی که آگهی‌ها را تامین می‌کنند، داده‌های غیرقابل پیش‌بینی دریافت شود که می‌تواند روی دقت جستجو و فیلترها تاثیر بگذارد.

پیچیدگی مدیریت میکروسرویس‌ها:

با افزایش تعداد میکروسرویس‌ها، مدیریت و هماهنگی بین آن‌ها می‌تواند پیچیده شود.

مسائل مربوط به امنیت داده‌ها:

اطلاعات حساس کاربران و آگهی‌ها نیاز به حفاظت دقیق دارند و هر گونه نقص در امنیت می‌تواند به مشکلات جدی منجر شود.

بدهی‌های فنی:

نیاز به توسعه‌ی مداوم واسط‌های:API

در صورتی کهAPI ها با تغییرات مکرر در منابع داده به‌روز نشوند، ممکن است دچار کندی یا از کار افتادن شوند.

بستگی به زیرساخت‌های خارجی:

استفاده از سرویس‌های کلود و بسترهای مدیریت شده ممکن است سیستم را به این سرویس‌ها وابسته کند.

مدیریت حالت‌های خطا و بازیابی:

برخورد با حالت‌های خطا و اطمینان از بازیابی سریع سیستم در زمان بروز مشکلات می‌تواند به تدریج به بدهی فنی تبدیل شود.