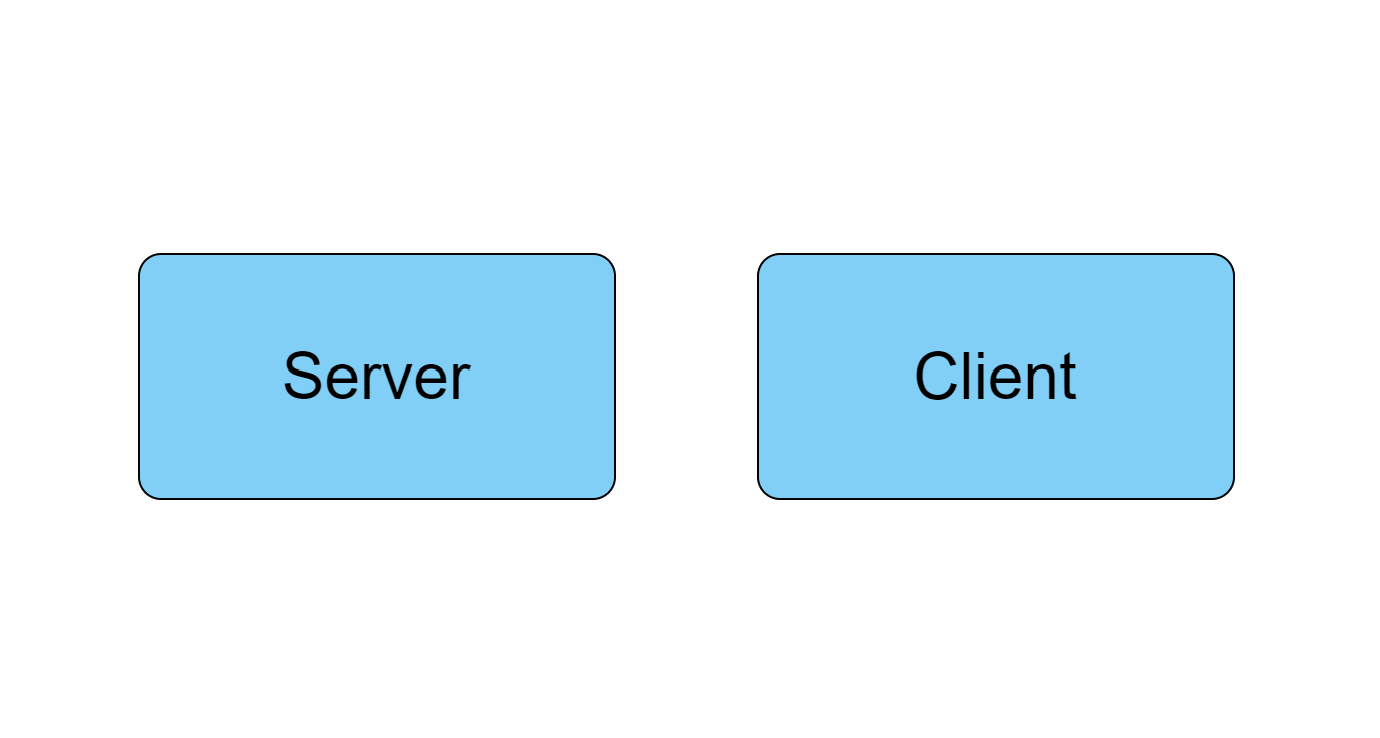
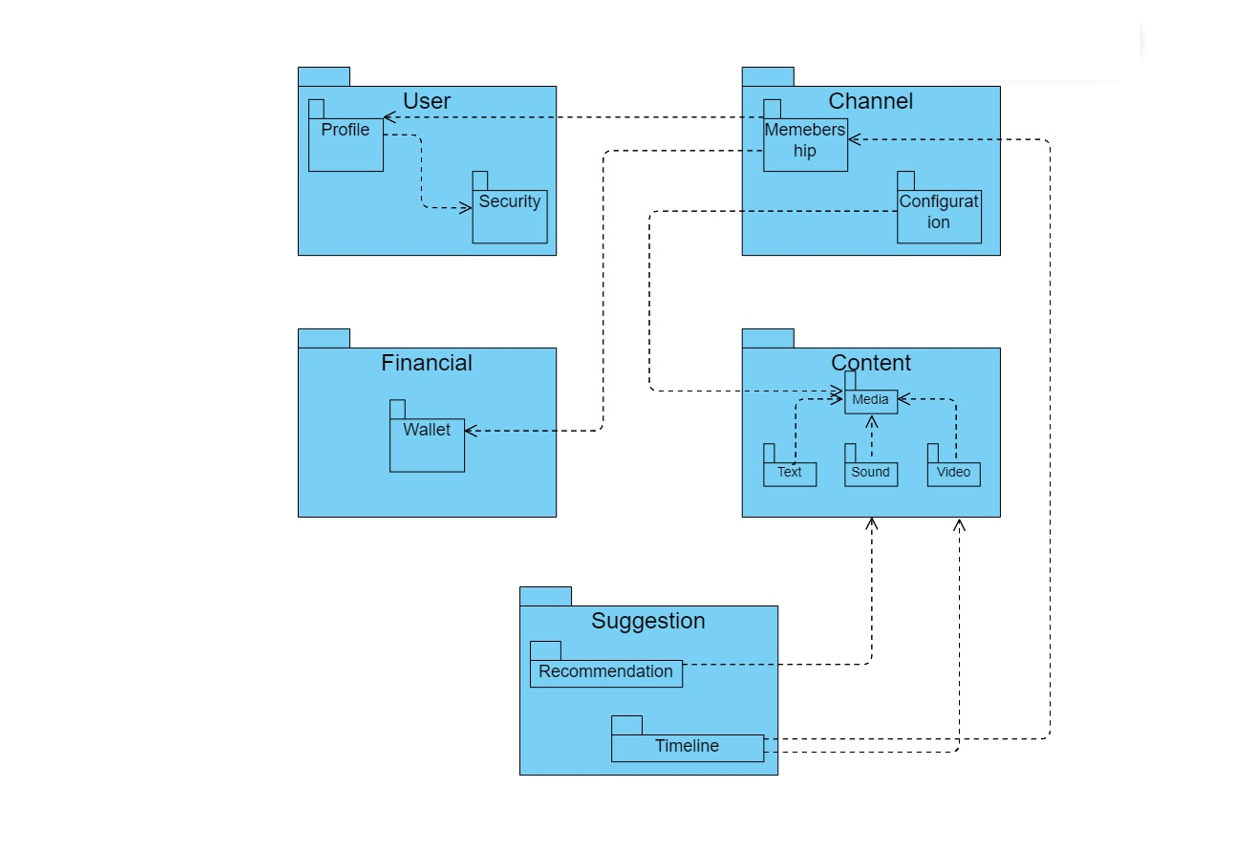
# مستند معماری اولیه

پس از بررسی موارد کاربرد و به دست آمدن بینش کلی نسبت به نیازمندی‌های سیستم، به بررسی راه‌حل‌هایی برای معماری سیستم می‌پردازیم. رویکرد ما برای معماری جداکردن دغدغه‌ها برای به‌کارگیری تصمیمات و مکانیسم‌های لازم در مواجهه با ریسک‌های سیستم است. بنابراین در طول مسیر و برای ساختن معماری، ابتدا مطابق تحلیل اولیه، زیرسیستم‌های تحلیل را به عنوان کامپوننت‌های اصلی سیستم در نظر می‌گیریم.

با جدا کردن دغدغه‌های مربوط به نمایش سیستم و منطق، می‌توان هم از پیچیدگی سیستم کاست و تقسیم کار را بین اعضا را بهتر انجام داد که بدین ترتیب به این تصمیم رسیدیم که از معماری کلاینت سرور به دو بخش اصلی سیستم استفاده کنیم.



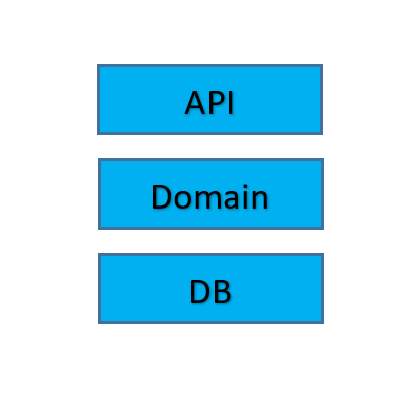
پس از آن برای ساختار داخلی سرور، برای اینکه به مدل‌های تحلیل و نیازمندی‌های مشتری نزدیک باشیم تا در صورت تغییرات بتوانیم آن را مدیریت کنیم، تلاش کردیم تا از همان زیرسیستم‌های اصلی به عنوان ساختار سطح بالا استفاده کنیم. به همین دلیل ساختار سطح بالای اولیه سیستم به شکل زیر خواهد بود:



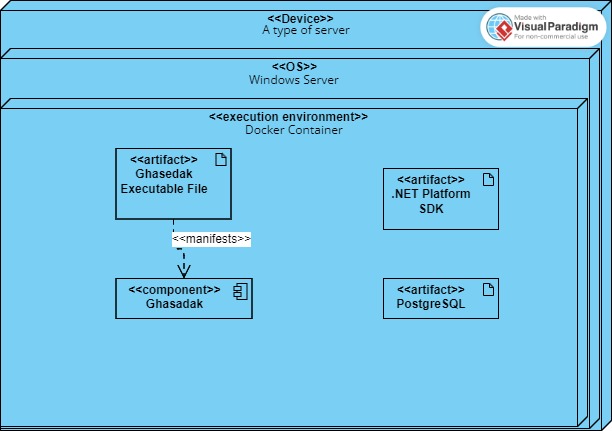
برای معماری سیستم از ساختار سطح بالای نمودار پکیج بالا و معادل آن استفاده می‌کنیم. در واقع سیستم را به تعدادی کامپوننت خواهیم شکست که هر کامپوننت مطابق معادل آن در نمودار پکیج خواهد بود که به دلیل ناکافی بودن جزئیات از آوردن آن صرف نظر شده است. در ادامه نیز هر کامپوننت به کامپوننت‌های کوچکتری خواهد شکست تا در نهایت تنها به فیچرهای‌ لازم و ریزدانه برسیم. همچنین مکانیسم‌های زیر برای مدیریت ریسک‌های موجود در پروژه اتخاذ شده است:

1. استفاده از پروسه‌های اتمیک برای مدیریت پرداختی‌های کاربران که در کامپوننت financial لحاظ خواهد شد.
2. نگهداری transactionهای سیستم به منظور بازیابی پرداخت‌ها در صورت بروز خطا که در کامپوننت financial لحاظ خواهد شد.

در هر کامپوننت یا ساب‌کامپوننت سیستم از یک ساختار سه سطحی استفاده می‌کنیم. هر کامپوننت، علاوه بر اینکه نیازمندی‌های لازم برای ارتباط با سایر بخش‌ها را از طریق نمای تعریف شده آن برآورده می‌سازد، بلکه مسئولیت ارتباطات مربوط به خود با کلاینت و پایگاه داده را نیز پیاده‌سازی می‌کند. این امر باعث یکپارچه‌سازی کامپوننت‌ها و نزدیک‌تر شدن به ساختار تحلیلی است.



برای مدیریت زیرساخت سیستم و ارتباط با سخت افزار نمودار زیر به عنوان یک دید از سطح بالا ترسیم گشته است:



فناوری های مورد استفاده:

برنامه سمت سرور(backend): با تصمیم اعضای تیم و با توجه به مهارت و تسلط اعضای تیم مقرر شد که برای برنامه نویسی سمت بکند از فریمورک asp.net core api استفاده شود. این فریمورک تصمیم خوبی در تکنولوژی های سمت سرور است و با آن میتوان نرم افزار های با قابلیت اتکای بالا و عملکرد (performance) مناسب توسعه داد. علاوه بر آن زبان مورد استفاده زبان برنامه نویسی c# است که شامل کتابخانه های بسیار زیاد و قوی ای است که کار توسعه را راحت میکند و کامیونیتی بالایی دارد.

برنامه نویسی سمت فرانت اند(frontend): با توجه به تصمیم اعضای تیم قرار شد از فریمورک react js برای برنامه نویسی سمت فرانت اند استفاده شود که یکی از فریمورک های قوی و مطرح است که شامل کتابخانه های بسیاری برای قطعات و widget های UI است و اعضای تیم از مهارت بالایی برخوردار هستند.

پایگاه داده: پایگاه داده مورد استفاده برای ذخیره اطلاعات postgresql است. انتخاب اعضای تیم sql server,mysql,postgresql بود ولی با توجه به آن که sql server رایگان نیست و نیاز به لایسنس دارد استفاده از آن منتفی شد. همچنین از میان postgresql و mysql با توجه به تحقیقات صورت گرفته توسط اعضای تیم نتیجه بر استفاده از postgresql شد زیرامکانات بهتری در اختیارمان قرار میدهد و همچنین از لحاظ عملکرد در خواندن و ریختن داده از mysql بهتر است.

DevOps : همان طور که در مستند نیازمندی های غیروظیفه ای گفته شد باید دسترس پذیری نرم افزار تامین شود. برای این کار قرار است برنامه ها dockerize شوند و با استفاده از docker compose در سمت سرور اجرا شوند بدین صورت که در سمت سرور یک proxy server مانند nginx قرار میگیرد و درخواست ها را به سمت سرور های بکند هدایت میکند. مزیت این کار علاوه بر load balancing و استفاده از گره های متعدد سرور تامین امنیت در برابر حملاتی مانند ddos است که تا حدودی از آن جلوگیری خواهد کرد.

سیستم عامل سرور: با توجه به آن که از فریمورک asp net core استفاده میکنیم پس میتوان خروجی نسخه نرم افزار را برای توزیع های لینوکس نیز آماده کرد. همچنین برای سرور ویندوزی نیز میتوان نسخه داد پس این فریمورک با سیستم عامل ویندوز و اکثر توزیع های لینوکسی مانند ابونتو،دبیان،فدورا،اپن سوزه و ... سازگار است. با توجه به آن که از لحاظ فنی دستمان باز است میتوانیم برای اکثر سیستم عامل های سرور ها نسخه نرم افزار را تولید کرده و deploy کنیم.

Cache : برای افزایش کارایی برنامه ممکن است که از کش redis استفاده شود تا ضمانت داده شده در نیازمندی های غیر وظیفه ای رفع شود.

ویژگی های سخت افزاری:

در گام اول تولید نرم افزار و deploy آن با توجه به آن که معماریمان monolith است و بودجه مان محدود است بر روی یک سرور تمامی برنامه deploy میشود. با توجه به نیازمندی های غیر وظیفه ای مطرح شده سرورمان حداقل 2 هسته cpu ، به همراه 4GB رم و حداقل 25GB حافظه برای ذخیره اطلاعات و فایل ها و محتوای 10 هزار کاربر میخواهد.