بیایید سیستمی طراحی کنیم مشابه علی‌بابا، جایی که کاربران بتوانند بلیط انواع وسایل نقلیه (هواپیما، قطار، اتوبوس) را خریداری کنند.

سرویس های مشابه: مستر بلیط، اسنپ‌تریپ

۱ سامانه رزرو بلیط باب‌علی چیست؟

باب‌علی یک سامانه آنلاین تحت وب است که امکان جستجو، رزرو و خرید بلیط وسایل نقلیه مختلف (هواپیما، قطار، اتوبوس) را برای کاربران فراهم می‌کند. کاربران می‌توانند بلیط‌ها را بر اساس تاریخ، مقصد، قیمت و نوع وسیله نقلیه جستجو کنند و پس از انتخاب، به‌صورت آنلاین پرداخت انجام داده و رسید دیجیتال دریافت کنند.

این سامانه دارای پایگاه داده داخلی است و مدیریت تمامی اطلاعات وسایل نقلیه و زمان‌بندی‌ها به‌صورت داخلی انجام می‌شود. هدف پروژه، پیاده‌سازی نسخه‌ای ساده از سامانه‌های مشابه است که تجربه کاربری روان و مدیریت کارآمد پایگاه داده را تضمین کند.

با توجه به این ویژگی‌ها، سامانه باب‌علی در دسته نرم‌افزارهای کاربردی/تجاری تحت وب قرار می‌گیرد، زیرا هدف اصلی آن پشتیبانی مستقیم از نیاز کاربران نهایی در فرآیند رزرو و خرید بلیط است.

۲ الزامات و اهداف سامانه

در طراحی سامانه باب‌علی، بر روی مجموعه‌ای از الزامات زیر تمرکز خواهیم کرد:

۲-۱ الزامات عملکردی (Functional Requirements)

۱. کاربران باید بتوانند بلیط وسایل نقلیه مختلف (هواپیما، قطار، اتوبوس) را جستجو، خریداری و رزرو کنند.

۲. امکان پرداخت آنلاین برای بلیط فراهم باشد و رسید دیجیتال صادر شود.

۳. سامانه باید وضعیت رزروها و موجودی بلیط‌ها را به‌صورت به‌روز نمایش دهد.

۲-۲ الزامات غیرعملکردی (Non-functional Requirements)

۱. سامانه باید از دسترس بالایی برخوردار باشد تا کاربران همیشه بتوانند خدمات را استفاده کنند.

۲. زمان پاسخ‌گویی سیستم برای جستجو و رزرو بلیط باید کمتر از ۲۵۰ میلی‌ثانیه باشد.

۳. سامانه باید از قابلیت اطمینان بالایی برخوردار باشد؛ هیچ اطلاعاتی از بلیط‌ها نباید از بین برود.

۴. مدیریت پایگاه‌داده باید کارآمد و پایدار باشد تا همزمان چندین کاربر بتوانند عملیات رزرو را بدون مشکل انجام دهند.

۳-۲ موارد خارج از محدوده (Not in scope)

- ارائه پیشنهاد خودکار وسیله نقلیه بر اساس تاریخچه کاربر
- امکان افزودن نظرات یا امتیازدهی پیشرفته
- اتصال به APIهای خارجی

۳ برخی ملاحظات طراحی

سامانه باب‌علی سیستم پرخوانشی خواهد بود؛ بنابراین تمرکز بر طراحی ساختاری است که بتواند اطلاعات بلیط‌ها و رزروها را به سرعت بازیابی کند.

۱. کاربران ممکن است همزمان چندین بلیط داشته باشند؛ بنابراین مدیریت بهینه پایگاه‌داده و ذخیره‌سازی اطلاعات باید در طراحی سیستم مورد توجه ویژه قرار گیرد.

۲. سیستم باید تأخیر کمی هنگام نمایش اطلاعات بلیط‌ها داشته باشد تا تجربه کاربری روان فراهم شود.

۳. داده‌ها باید ٪۱۰۰ قابل اطمینان باشند. اگر کاربری رزرو یا خرید بلیطی انجام دهد، سامانه باید تضمین کند که هیچ اطلاعاتی از بین نرود.

۴. هر زیرسیستم (مثلاً بلیط اتوبوس، قطار، هواپیما) باید به‌صورت جداگانه مدیریت شود تا مشکلات یا خطاهای یک زیرسیستم، عملکرد دیگر بخش‌ها را مختل نکند.

۴ برآورد ظرفیت و محدودیت‌ها

فرض کنید سامانه باب‌علی دارای ۲۰،۰۰۰،۰۰۰ کاربر کل باشد، که ۲،۰۰۰،۰۰۰ کاربر فعال روزانه دارند.

- روزانه حدود ۲،۰۰۰،۰۰۰ رزرو و خرید بلیط انجام می‌شود، تقریباً ۲۳ رزرو در هر ثانیه.
- میانگین حجم هر رزرو/بلیط و اطلاعات مرتبط با سفرها حدود ۲۰۰ کیلوبایت است.

فضای مورد نیاز برای ۱ روز رزروها و بلیط‌ها:

$$2\,000\,000 \times 200KB = 400\,GB$$

فضای مورد نیاز برای ۱ هفته فعالیت سامانه:

$$400\,GB \times 7 \approx 2.8\,TB$$

۵ طراحی سیستم

در سطح کلان، سامانه باب‌علی باید دو سناریو اصلی را پشتیبانی کند: ثبت رزرو و خرید بلیط‌ها و همچنین جستجو و مشاهده بلیط‌ها.

- هر اپلیکیشن مرتبط با یک نوع وسیله نقلیه به‌صورت جداگانه مدیریت می‌شود تا اختلال در یک سرویس، عملکرد سایر سرویس‌ها را تحت تأثیر قرار ندهد.
- اطلاعات رزروها پس از گذشت یک هفته از پایان سفر به‌صورت خودکار حذف می‌شوند تا حجم داده‌ها کنترل شده و کارایی سیستم حفظ گردد.
- پایگاه داده به‌صورت پایدار طراحی شده تا در برابر خرابی سرورها و از دست رفتن داده‌ها محافظت شود.
- سیستم باید قابلیت مقیاس‌پذیری داشته باشد تا بتواند تعداد کاربران و رزروهای روزانه را بدون افت عملکرد پشتیبانی کند.
- طراحی سامانه به گونه‌ای است که عملیات ثبت، جستجو و پرداخت رزروها با کمترین تأخیر و تجربه کاربری روان انجام شود.

۶ مسیر داده و جریان کار (Data Flow / Workflow)

در این بخش، جریان داده‌ها و تعاملات بین بخش‌های مختلف سیستم شامل کاربر، سرور، پایگاه داده و درگاه پرداخت توضیح داده می‌شود. فرآیند اصلی سیستم شامل مراحل جستجو، رزرو و پرداخت بلیط است که در Swimlane Diagram مشخص شده و مسئولیت هر بخش (Lane) به‌وضوح نمایش داده می‌شود.

- **جستجوی بلیط:**
  - کاربر اطلاعات سفر (source, destination, date) را وارد می‌کند.
  - سرور درخواست را دریافت کرده و به پایگاه داده ارسال می‌کند.
  - پایگاه داده نتایج مطابق با مشخصات سفر را جستجو می‌کند.
  - نتایج جستجو توسط سرور پردازش شده و به کاربر نمایش داده می‌شود.
- **رزرو بلیط:**
  - کاربر صندلی مورد نظر خود را انتخاب می‌کند.
  - سرور بررسی می‌کند که صندلی در دسترس است یا خیر.
  - در صورت آزاد بودن صندلی:
    - صندلی به کاربر اختصاص داده شده و وضعیت آن به حالت pending تغییر می‌کند.
    - یک تایمر ۱۰ دقیقه‌ای برای تکمیل خرید فعال می‌شود.
    - اگر خرید طی ۱۰ دقیقه انجام نشود، سرور صندلی را آزاد (unset) می‌کند و دوباره برای سایر کاربران در دسترس قرار می‌دهد.
- **پرداخت:**
  - کاربر اطلاعات پرداخت را وارد کرده و سرور آن را به درگاه پرداخت (payment gateway) ارسال می‌کند.
  - درگاه پرداخت تراکنش را پردازش کرده و نتیجه (success یا failure) را به سرور اعلام می‌کند.
  - در صورت موفقیت‌آمیز بودن تراکنش:
    - رزرو نهایی شده و رکورد پایگاه داده به‌روز می‌شود.
    - رسید پرداخت (receipt) برای کاربر صادر می‌شود.

- فایل PDF بلیط (ticket PDF) تولید شده و از طریق ایمیل یا دانلود مستقیم در اختیار کاربر قرار می‌گیرد.

- در صورت عدم موفقیت تراکنش:
  - رزرو موقت لغو شده و صندلی آزاد می‌شود.
  - کاربر می‌تواند دوباره برای رزرو اقدام کند.

- **کنترل و ثبت وضعیت:**

- سرور تمامی رویدادها شامل جستجو، رزرو موقت، پرداخت و لغو رزرو را در پایگاه داده ثبت می‌کند.
- سیستم لاگ‌ها (logs) برای مانیتورینگ و بررسی خطاها ذخیره می‌کند.
- در صورت بروز مشکل، پیام خطا به کاربر نمایش داده شده و وضعیت صندلی‌ها بازبینی می‌شود.

- **نمایش وضعیت نهایی:**

- پس از تکمیل موفق پرداخت، کاربر وضعیت نهایی رزرو را مشاهده می‌کند.
- اطلاعات شامل شماره بلیط، صندلی، تاریخ و زمان سفر، و رسید پرداخت به کاربر ارائه می‌شود.
- در صورت لغو یا عدم موفقیت در پرداخت، پیام مناسب به کاربر نشان داده می‌شود تا مجدداً اقدام کند.

۷ ساختار پایگاه داده

سامانه باب‌علی برای مدیریت بلیط‌های کاربران و وضعیت سفرها نیازمند یک پایگاه داده رابطه‌ای (RDBMS) مانند MySQL است. طراحی پایگاه داده باید امکان ذخیره‌سازی پایدار اطلاعات کاربران، بلیط‌ها و سفرها را فراهم کند و همچنین دسترسی سریع به بلیط‌های اخیر و گزارش‌گیری را ممکن سازد.

استفاده از مدل‌های Django برای پیاده‌سازی این جداول باعث می‌شود روابط بین موجودیت‌ها به صورت واضح و قابل نگهداری باشد و عملیات CRUD به شکل استاندارد انجام شود.

۷-۱ جداول اصلی و مدل‌ها

- **User:** شامل مشخصات کاربران مانند نام، نام خانوادگی و اطلاعات تماس است. کلید اصلی این جدول UserID است.

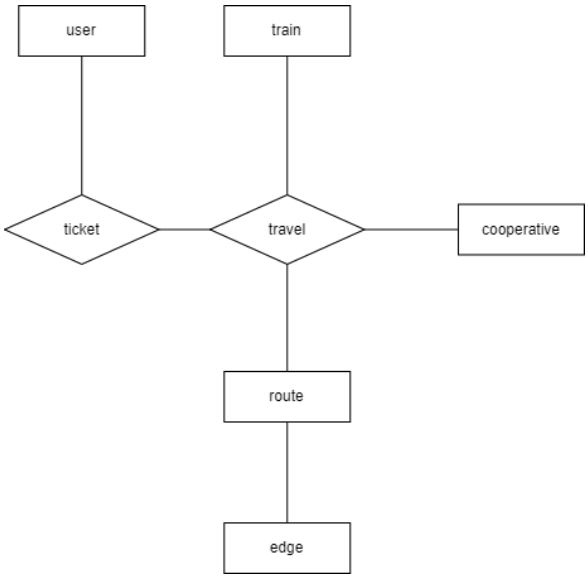
- **Cooperative / Agency:** شامل اطلاعات شرکت‌های حمل و نقل، شماره تماس و لوگوی آن‌ها است. هر شرکت می‌تواند چند سفر ارائه دهد (One-to-Many با جدول Travel).
- **Terminal / Airport / Station:** مشخصات پایانه‌ها، فرودگاه‌ها یا ایستگاه‌ها، شامل نام، شهر، آدرس و شماره تماس. این جدول به Travel متصل است تا مبدا و مقصد هر سفر مشخص شود.
- **Bus / Train / Airplane:** اطلاعات وسایل نقلیه شامل نوع، ظرفیت صندلی‌ها و مدل. هر وسیله نقلیه به سفرهای مشخصی اختصاص دارد (Foreign Key به جدول Travel).
- **Travel:** اطلاعات هر سفر شامل مبدا و مقصد، تاریخ و ساعت حرکت، ظرفیت، قیمت، توضیحات و وضعیت صندلی‌ها (در قالب JSON). ظرفیت به‌صورت خودکار بر اساس وسیله نقلیه مرتبط تنظیم می‌شود. این جدول با User، Ticket و Cooperative / Agency در ارتباط است.
- **Ticket:** شامل اطلاعات بلیط‌های خریداری شده توسط کاربران است. این جدول شماره بلیط (TicketID) را به عنوان کلید اصلی دارد و با User و Travel رابطه دارد. اطلاعات ذخیره شده شامل شماره صندلی، مشخصات مسافر (نام، نام خانوادگی، کد ملی و تاریخ تولد) و وضعیت پرداخت است.

۲-۷ مزایا و توضیحات مدل‌ها

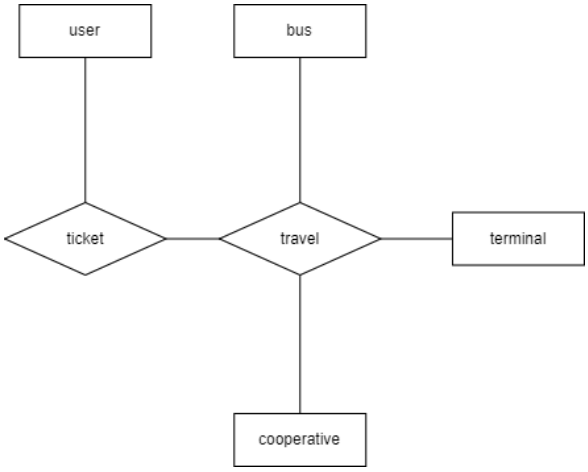
استفاده از RDBMS و مدل‌های Django امکان انجام JOIN بین جداول، گزارش‌گیری انعطاف‌پذیر و نگهداری داده‌ها به‌صورت متمرکز و پایدار را فراهم می‌کند. مدل‌ها به‌صورت کلاس‌های Python تعریف می‌شوند و روابط بین موجودیت‌ها با استفاده از ForeignKey، OneToMany و ManyToMany نمایش داده می‌شوند. به عنوان مثال:

- هر Travel متعلق به یک Cooperative / Agency است و می‌تواند چندین Ticket داشته باشد.
- هر User می‌تواند چندین Ticket خریداری کند، اما هر بلیط مربوط به یک سفر مشخص است.
- ارتباط بین Terminal / Airport / Station و Travel تضمین می‌کند که مبدا و مقصد هر سفر به درستی مشخص شوند.

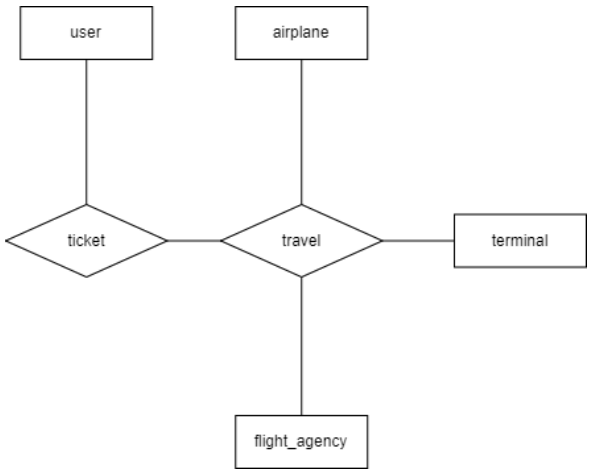
این طراحی امکان افزودن امکانات جدید مانند تخفیف‌ها، پیشنهادات ویژه و مدیریت صندلی‌ها بدون نیاز به تغییر ساختار اصلی پایگاه داده را فراهم می‌کند.



Train Application



Bus Application



Flight Application

شکل ۱: ER Diagram پایگاه داده برای سه اپلیکیشن سامانه باب‌علی

۸ انتخاب و اعمال روش چابک

برای توسعه سامانه رزرو بلیط باب‌علی، تیم سه نفره ما تلاش کرد تا به روش Scrum نزدیک شود. پروژه در چند اسپرینت کوتاه طراحی و اجرا شد و هر اسپرینت شامل فعالیت‌های چارچوبی بود که هدف آن افزایش انعطاف‌پذیری، پاسخ سریع به تغییرات و بهبود مستمر محصول بود.

۸-۱ ارتباطات (Communication)

- جلسات کوتاه روزانه (Daily Stand-up) برای هماهنگی اعضای تیم برگزار شد تا هر عضو بتواند پیشرفت کار خود، مشکلات احتمالی و نیاز به کمک را مطرح کند.

- تصمیمات تیم بر اساس مقایسه با نمونه اصلی Alibaba و نیازهای پروژه اتخاذ شد و تمامی تغییرات به صورت جمعی بررسی و تأیید شدند.
- استفاده از ابزارهای مدیریت پروژه و پیام‌رسانی داخلی، امکان ثبت و پیگیری تمام مباحث و وظایف را فراهم کرد و شفافیت در کار تیم را افزایش داد.

۸-۲ برنامه‌ریزی (Planning)

- اهداف کوتاه‌مدت هر اسپرینت مشخص شد تا تمرکز تیم بر اولویت‌های کلیدی حفظ شود و امکان ارزیابی پیشرفت پروژه در بازه‌های زمانی کوتاه فراهم گردد.
- فعالیت‌ها اولویت‌بندی شدند تا تیم سه نفره بتواند آن‌ها را به‌طور مؤثر مدیریت کند و منابع محدود تیم به بهترین نحو مورد استفاده قرار گیرد.
- در طول برنامه‌ریزی، هر وظیفه به گام‌های کوچک‌تر تقسیم شد تا تخمین زمان انجام کار دقیق‌تر باشد و امکان تحویل تدریجی و پیوسته فراهم شود.

۸-۳ ساخت و ساز و کدنویسی (Construction – Coding)

- در هر اسپرینت، که مربوط به یک اپ از پروژه بود، هر عضو تیم مسئول بخشی از اپ می‌شد و همه اعضا با همکاری هم روی پیشرفت اپ کار می‌کردند.
- کدها با رعایت اصول Clean و قابل نگهداری نوشته شدند و تمامی کلاس‌ها و توابع دارای نام‌گذاری واضح و مستندات مرتبط بودند.
- روند Iterative داخلی تیم باعث اصلاح و بهبود مداوم کدها شد و بازخوردها در هر چرخه سریعاً اعمال می‌شد.

۸-۴ ساخت و ساز و آزمایش (Construction – Testing)

- تست هر ماژول ابتدا به‌صورت دستی انجام شد تا صحت عملکرد هر بخش و ارتباط بین ماژول‌ها بررسی شود.
- تست‌ها از طریق Postman برای بررسی درخواست‌ها و پاسخ‌های سرور انجام شد و خطاها شناسایی و رفع شدند.

- در مراحل بعد، تست‌های خودکار برای بخش‌های کلیدی سامانه ایجاد شدند تا روند کنترل کیفیت سریع‌تر و کم‌خطاتر شود.
- بازخورد تیم در طول آزمایش نسخه‌های اولیه جمع‌آوری شد و تغییرات مطابق با تجربه کاربری و عملکرد بهتر سامانه اعمال شد.

۸-۵ استقرار (Deployment)

- در پایان هر اسپرینت، با تکمیل هر اپلیکیشن، یک نسخه کامل از پروژه آماده شد تا عملکرد سامانه بررسی و نقاط ضعف احتمالی شناسایی شود.
- تغییرات و بهبودها در اسپرینت‌های بعدی اعمال شد تا سامانه با نمونه اصلی هماهنگ‌تر شده و تجربه کاربری بهبود یابد.
- پس از تکمیل نسخه نهایی، با ساخت Docker Image، محیط استقرار آماده شد تا امکان اجرای یکپارچه و انتقال آسان سامانه فراهم گردد.
- فرآیند استقرار و تحویل تدریجی به تیم امکان داد تا بازخوردها را در اسپرینت‌های بعدی اعمال کرده و کیفیت سامانه به طور مستمر بهبود یابد.=

۹ توازن بار

هر اپلیکیشن سامانه (اتوبوس، قطار، هواپیما) شامل سه بخش اصلی است: رابط کاربری (Frontend)، منطق کسب‌وکار (Backend) و پایگاه داده (Database). برای اطمینان از عملکرد بهینه در شرایط افزایش تعداد کاربران، هر بخش به صورت یک Docker Image مستقل اجرا شده است. این طراحی باعث می‌شود توسعه، نگهداری و مقیاس‌پذیری سیستم ساده‌تر باشد.

برای Frontend، یک Docker Image اختصاص داده شده است که مسئول پاسخ‌دهی به درخواست‌های اولیه کاربران و ارائه رابط کاربری است. این بخش سبک و مقیاس‌پذیر است و می‌تواند نمونه‌های متعدد داشته باشد تا تعداد کاربران زیاد را پشتیبانی کند.

برای Backend، هر اپلیکیشن (اتوبوس، قطار، هواپیما) یک Docker Image مستقل دارد. این کار اجازه می‌دهد که درخواست‌های کاربران برای هر سرویس به‌صورت جداگانه مدیریت شوند و امکان توزیع بار میان نمونه‌های مختلف هر اپلیکیشن فراهم شود. هر نمونه Replica از سرور برنامه می‌تواند بخشی از درخواست‌ها را پردازش کند، و در نتیجه هیچ سرور منفردی تحت فشار بیش از حد قرار نمی‌گیرد.

برای Database، یک Docker Image جداگانه در نظر گرفته شده است. این طراحی باعث می‌شود پایگاه داده مستقل از بخش‌های دیگر باشد و نگهداری آن ساده‌تر شود. در صورت افزایش ترافیک و نیاز به عملکرد بهتر، می‌توان از Database Clustering یا Replication استفاده کرد تا کارایی و پایداری سیستم افزایش یابد.

نقش Load Balancer در این معماری حیاتی است. این توزیع‌کننده بار میان نمونه‌های مختلف Backend قرار می‌گیرد و درخواست‌های ورودی کاربران را به شکل یکنواخت میان سرورها تقسیم می‌کند. بدین ترتیب:

- هیچ سرور منفردی بیش از حد بارگذاری نمی‌شود.
- از ایجاد گلوگاه (Bottleneck) در یک سرور خاص جلوگیری می‌شود.
- مقیاس‌پذیری سیستم افزایش می‌یابد و امکان اضافه کردن نمونه‌های جدید به سادگی وجود دارد.

۱۰ برنامه‌های توسعه آینده

در راستای گسترش امکانات سامانه و بهبود تجربه کاربران، قصد داریم در مراحل بعدی پروژه قابلیت‌های جدیدی را پیاده‌سازی کنیم. یکی از اهداف اصلی، افزودن امکان رزرو هتل (Hotel Booking) به سامانه است تا کاربران بتوانند پس از برنامه‌ریزی سفر با اتوبوس، قطار یا هواپیما، به‌طور مستقیم هتل موردنظر خود را نیز رزرو کنند. این قابلیت به کاربران اجازه می‌دهد تمامی مراحل سفر را در یک سامانه مدیریت کنند و تجربه یکپارچه‌ای داشته باشند.

علاوه بر این، برنامه‌ریزی برای استفاده از Cache در بخش‌های مختلف سامانه در نظر گرفته شده است. استفاده از Cache به کاهش زمان پاسخ‌دهی درخواست‌ها و بهبود عملکرد سامانه در شرایط ترافیک بالا کمک می‌کند. این بهینه‌سازی می‌تواند شامل ذخیره‌سازی موقت داده‌های پرتکرار کاربران یا نتایج جستجوها باشد تا نیاز به دسترسی مکرر به پایگاه داده کاهش یابد.

پیاده‌سازی این قابلیت‌ها به‌گونه‌ای انجام خواهد شد که مقیاس‌پذیری و نگهداری سامانه تحت فشار کاربران زیاد آسان باشد. همچنین، طراحی سامانه به گونه‌ای خواهد بود که افزودن ویژگی‌های جدید در آینده بدون تغییرات عمده در ساختار اصلی امکان‌پذیر باشد.

کلام پایانی

امیدوارم این گزارش توانسته باشد دیدگاهی کامل و جامع از طراحی و پیاده‌سازی سامانه رزرو بلیط و مدیریت سفر ارائه دهد. برای دسترسی آسان‌تر به کدهای کامل پروژه و بررسی دقیق‌تر جزئیات فنی، می‌توانید به مخزن گیت‌هاب پروژه مراجعه کنید:

https://github.com/MMDs-team/babali