به‌نام خدا

طراحی و پیاده سازی سامانه باب‌علی برای رزرو بلیط وسایل نقلیه

محمد خندانی، محمد قادری، محمد معجونی

دانشگاه ارومیه

شهریور ماه ۱۴۰۴

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 اهداف و الزامات سامانه

سامانه باب‌علی با هدف ارائه یک بستر جامع و کارآمد برای مدیریت رزرو و خرید بلیط وسایل نقلیه مختلف طراحی شده است. در این بخش، اهداف کلان، الزامات عملکردی و غیرعملکردی، و همچنین قابلیت‌هایی که در محدوده طراحی فعلی قرار ندارند بیان می‌گردند.

۲-۱ اهداف سامانه

اهداف اصلی طراحی سامانه به شرح زیر هستند:

- ایجاد یک محیط یکپارچه برای جستجو، رزرو و خرید بلیط وسایل نقلیه (هواپیما، قطار، اتوبوس) بدون نیاز به مراجعه به سامانه‌های جداگانه.
- ارائه تجربه کاربری سریع، روان و ایمن در تمامی مراحل تعامل کاربر با سامانه.
- اطمینان از امنیت اطلاعات کاربران و جلوگیری از نشت یا سوءاستفاده از داده‌ها.
- فراهم کردن قابلیت صدور بلیط به صورت PDF و ارسال آن به کاربران پس از پرداخت موفق.
- پشتیبانی از رشد تدریجی کاربران و مقیاس‌پذیری زیرساخت به‌گونه‌ای که بتوان به افزایش ناگهانی حجم درخواست‌ها پاسخ داد.
- تسهیل نگهداری و توسعه سامانه از طریق معماری ماژولار و مستندسازی مناسب.

۲-۲ الزامات عملکردی (Functional Requirements)

الزامات عملکردی، قابلیت‌هایی هستند که کاربر مستقیماً با آن‌ها سروکار دارد:

۱. امکان جستجو بر اساس نوع وسیله نقلیه، تاریخ سفر، مبدا و مقصد.

۲. نمایش موجودی بلیط‌ها و به‌روزرسانی لحظه‌ای ظرفیت باقی‌مانده.

۳. امکان رزرو اولیه بلیط و نگه‌داشتن آن در بازه زمانی محدود تا زمان پرداخت.

۴. قابلیت خرید آنلاین و اتصال به درگاه پرداخت معتبر.

۵. صدور بلیط دیجیتال در قالب فایل PDF Ticket پس از تکمیل فرآیند خرید.

۶. مدیریت رزروها شامل مشاهده، لغو و پیگیری وضعیت.

۷. پشتیبانی از اعلان‌ها مانند ارسال پیامک یا ایمیل پس از خرید یا لغو بلیط.

۲-۳ الزامات غیرعملکردی (Non-functional Requirements)

الزامات غیرعملکردی تضمین‌کننده کیفیت سرویس و زیرساخت سامانه هستند:

- **دسترس‌پذیری:** سامانه باید حداقل در 99.9% از زمان در دسترس کاربران باشد.
- **کارایی:** زمان پاسخ‌گویی سیستم برای عملیات جستجو و رزرو کمتر از ۲۵۰ میلی‌ثانیه باشد.

- **امنیت:** اطلاعات کاربران و تراکنش‌ها باید با استفاده از پروتکل‌های امن (HTTPS، TLS) محافظت شوند.
- **قابلیت اطمینان:** هیچ اطلاعاتی از بلیط‌ها و تراکنش‌ها نباید در اثر خطای سیستمی یا بار زیاد از دست برود.
- **مقیاس‌پذیری:** معماری سیستم باید امکان افزایش سریع منابع را برای پشتیبانی از حجم بالای کاربران داشته باشد.
- **نگهداری‌پذیری:** کدها و زیرساخت باید به گونه‌ای طراحی شوند که تغییرات و به‌روزرسانی‌ها به راحتی و با کمترین ریسک اعمال شوند.

۲-۴ موارد خارج از محدوده (Out of Scope)

در نسخه اولیه سامانه، برخی قابلیت‌ها پیش‌بینی نشده‌اند و در محدوده طراحی فعلی قرار ندارند:

- پیشنهاد هوشمند وسیله نقلیه یا مسیر سفر بر اساس تاریخچه کاربر.
- افزودن بخش اجتماعی مانند ثبت نظر یا امتیازدهی پیشرفته برای خدمات.
- اتصال به APIهای خارجی مانند مقایسه قیمت با دیگر سامانه‌ها.
- پشتیبانی از روش‌های پرداخت بین‌المللی یا چند ارزی.

۲-۵ خلاصه الزامات

به طور خلاصه، سامانه باب‌علی باید:

۱. خدمات رزرو و خرید آنلاین بلیط را با امنیت و سرعت بالا ارائه دهد.

۲. از نظر مقیاس‌پذیری و دسترس‌پذیری در سطح استانداردهای حرفه‌ای باشد.

۳. قابلیت توسعه تدریجی و افزودن امکانات جدید در آینده را داشته باشد.

۳ معماری سامانه

معماری سامانه باب‌علی به صورت چندلایه‌ای طراحی شده است تا انعطاف‌پذیری، مقیاس‌پذیری و امنیت آن تضمین گردد. هر لایه مسئولیت مشخصی بر عهده دارد و تعامل بین آن‌ها با کمترین وابستگی انجام می‌شود. لایه‌های اصلی عبارتند از:

- **لایه ارائه (Presentation Layer):** شامل رابط کاربری تحت وب و موبایل است که وظیفه نمایش اطلاعات سفر، دریافت ورودی کاربران و ارسال درخواست‌ها به سرور را بر عهده دارد.
- **لایه منطق کسب‌وکار (Business Logic Layer):** این لایه شامل سرویس‌های Backend است که عملیات اصلی همچون مدیریت رزرو، پردازش پرداخت و صدور بلیط دیجیتال را انجام می‌دهند.
- **لایه داده (Data Layer):** شامل پایگاه‌داده رابطه‌ای و کش توزیع‌شده است. این لایه مسئول ذخیره‌سازی پایدار داده‌ها و تضمین صحت و یکپارچگی اطلاعات می‌باشد.
- **لایه یکپارچه‌سازی (Integration Layer):** این لایه ارتباط با سرویس‌های خارجی مانند درگاه‌های پرداخت، سامانه ارسال اعلان (SMS و Email) و سایر APIهای جانبی را فراهم می‌کند.

این ساختار لایه‌ای موجب می‌شود تغییرات یا به‌روزرسانی در یک بخش، کمترین تأثیر را بر سایر بخش‌ها داشته باشد و توسعه و نگهداری سامانه ساده‌تر گردد.

۴ برآورد ظرفیت و محدودیت‌ها

برای تخمین منابع مورد نیاز سامانه باب‌علی، فرض می‌کنیم این سامانه در مجموع دارای حدود ۲۰ میلیون کاربر ثبت‌شده باشد که از میان آن‌ها تقریباً ۲ میلیون کاربر به‌طور روزانه فعال هستند.

۴-۱ حجم تراکنش‌ها

با توجه به الگوی استفاده، پیش‌بینی می‌شود که به‌طور متوسط روزانه نزدیک به ۲ میلیون رزرو و خرید بلیط در سامانه ثبت گردد. این مقدار معادل تقریباً ۲۳ تراکنش در هر ثانیه است که بار قابل توجهی بر زیرساخت وارد می‌کند.

۴-۲ حجم داده‌ها

میانگین حجم هر رزرو (شامل اطلاعات بلیط و جزئیات سفر) حدود ۲۰۰ کیلوبایت برآورد می‌شود. بنابراین:

- فضای مورد نیاز برای ذخیره‌سازی یک روز رزروها و بلیط‌ها:

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

- فضای مورد نیاز برای یک هفته فعالیت سامانه:

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

۳-۴ تحلیل و محدودیت‌ها

این محاسبات نشان می‌دهد که حتی در بازه‌ی کوتاه‌مدت (مثلاً یک هفته)، حجم داده‌های تولیدی بسیار بالا خواهد بود. بنابراین:

۱. نیاز به راهکارهای ذخیره‌سازی مقیاس‌پذیر و توزیع‌شده اجتناب‌ناپذیر است.

۲. استفاده از سیاست‌های پاک‌سازی خودکار برای رزروهای منقضی‌شده (پس از پایان سفر) ضروری است.

۳. بهره‌گیری از مکانیزم‌های فشرده‌سازی و کش می‌تواند فشار بر پایگاه‌داده اصلی را کاهش دهد.

۵ ملاحظات طراحی

در طراحی سامانه باب‌علی، علاوه بر الزامات عملکردی و غیرعملکردی، مجموعه‌ای از ملاحظات معماری و فنی نیز در نظر گرفته شده است تا سامانه بتواند پاسخگوی نیازهای عملیاتی و مقیاس‌پذیری باشد. مهم‌ترین این ملاحظات عبارت‌اند از:

۱. **ماهیت پرخوانش داده‌ها:** از آنجا که اکثر درخواست‌های کاربران شامل جستجو و مشاهده بلیط‌ها است، ساختار پایگاه داده و کش (Cache) باید به گونه‌ای طراحی شود که دسترسی سریع به داده‌ها امکان‌پذیر باشد.

۲. **مدیریت همزمانی:** کاربران ممکن است به‌طور همزمان رزروهای مشابهی را انجام دهند. در نتیجه، مدیریت تراکنش‌ها و قفل‌گذاری در سطح پایگاه‌داده باید به‌گونه‌ای باشد که از بروز شرایط رقابتی و تضاد داده‌ها جلوگیری شود.

۳. **تفکیک زیرسیستم‌ها:** هر سرویس (بلیط اتوبوس، قطار و هواپیما) به صورت جداگانه پیاده‌سازی و مدیریت می‌شود تا اختلال در یک بخش باعث توقف کل سامانه نشود. این تفکیک، امکان توسعه و نگهداری مستقل هر سرویس را نیز فراهم می‌کند.

۴. **پایداری داده‌ها:** داده‌های مرتبط با خرید و رزرو باید کاملاً قابل اطمینان باشند. به همین دلیل از مکانیزم‌های Replication و پشتیبان‌گیری دوره‌ای استفاده خواهد شد تا هیچ اطلاعاتی از دست نرود.

۵. **کارایی و مقیاس‌پذیری:** برای اطمینان از سرعت پاسخ‌دهی بالا، استفاده از ایندکس‌های بهینه در پایگاه‌داده، به‌کارگیری Load Balancer و مقیاس‌گذاری پویا با Kubernetes در نظر گرفته شده است.

۶. **امنیت و محرمانگی:** کلیه داده‌های کاربران به‌ویژه اطلاعات پرداخت باید با استانداردهای امنیتی رمزنگاری شوند و سیاست‌های امنیتی کنترل شوند.

۶ طراحی سیستم

سامانه رزرو بلیط باب‌علی به‌گونه‌ای طراحی شده است که هر اپلیکیشن مرتبط با یک نوع وسیله نقلیه به‌صورت مستقل مدیریت شود تا اختلال در یک سرویس، عملکرد سایر بخش‌ها را تحت تأثیر قرار ندهد. به علاوه، پایگاه داده پایدار طراحی شده و عملیات رزرو، جستجو و پرداخت با کمترین تأخیر و تجربه کاربری روان انجام می‌شود.

در پیاده‌سازی کد پروژه، از چند الگوی طراحی (Design Pattern) استفاده شده است تا ساختار کد قابل نگهداری، منعطف و قابل توسعه باشد. استفاده از این الگوها موجب شد بخش‌های مختلف سیستم بتوانند مستقل کار کنند و در عین حال تعاملات بین آن‌ها به‌صورت منظم و کنترل‌شده باشد.

۶-۱ الگوهای طراحی استفاده‌شده

- Singleton: این الگو برای مدیریت یکپارچه اتصال به پایگاه داده (Database Connection) استفاده شد. با به‌کارگیری Singleton، از ایجاد نمونه‌های متعدد جلوگیری شد و همه ماژول‌ها به یک اتصال مشترک دسترسی دارند.
- Factory: برای تولید اشیاء سفر (Travel) با توجه به نوع وسیله نقلیه از الگوی Factory استفاده شد. این کار باعث شد افزودن انواع جدید سفر بدون تغییر کد اصلی امکان‌پذیر باشد و وابستگی‌ها کاهش یابند.

- Observer: هنگام تغییر وضعیت رزرو یا پردازش پرداخت، لازم بود کلاس‌های مرتبط مانند Ticket و User از این تغییر مطلع شوند. الگوی Observer برای اطلاع‌رسانی خودکار و مدیریت همزمان وضعیت‌ها به کار گرفته شد.
- Strategy: سامانه چندین روش پرداخت (Payment Gateway) دارد. با الگوی Strategy، انتخاب روش پرداخت به‌صورت پویا و مطابق با نیاز کاربر انجام می‌شود، بدون اینکه نیاز به تغییر در منطق اصلی رزرو باشد.
- Template Method: پردازش رزرو و خرید بلیط شامل مراحل مشخصی است: انتخاب صندلی، تایید موجودی، پردازش پرداخت و تولید بلیط. الگوی Template Method اجازه می‌دهد این مراحل عمومی تعریف شوند و در هر نوع وسیله نقلیه، مراحل جزئی به‌صورت سفارشی اجرا شوند.

۲-۶ مزایا و تأثیر در پروژه

استفاده از این الگوهای طراحی مزایای زیر را در پروژه فراهم کرد:

- کاهش وابستگی بین کلاس‌ها و ماژول‌ها و افزایش قابلیت نگهداری کد.
- امکان افزودن ویژگی‌ها و امکانات جدید بدون تغییرات گسترده در ساختار اصلی.
- بهبود هماهنگی بین کلاس‌های مرتبط با رزرو و پرداخت بلیط، به‌ویژه کلاس‌های Travel و Ticket.
- ساده‌سازی مدیریت اتصال به پایگاه داده و اطمینان از یکپارچگی داده‌ها.
- امکان تست جداگانه هر ماژول با حداقل نیاز به شبیه‌سازی دیگر بخش‌ها.

با این طراحی، سامانه به‌گونه‌ای ساختاربندی شده که همزمان چندین کاربر می‌توانند عملیات رزرو، پرداخت و مشاهده بلیط را انجام دهند و در عین حال قابلیت مقیاس‌پذیری و افزودن امکانات جدید در آینده به‌سادگی فراهم باشد.

۷ 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.

۸ 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.

۸-۱ 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.

۸-۲ مزایا و توضیحات مدل‌ها

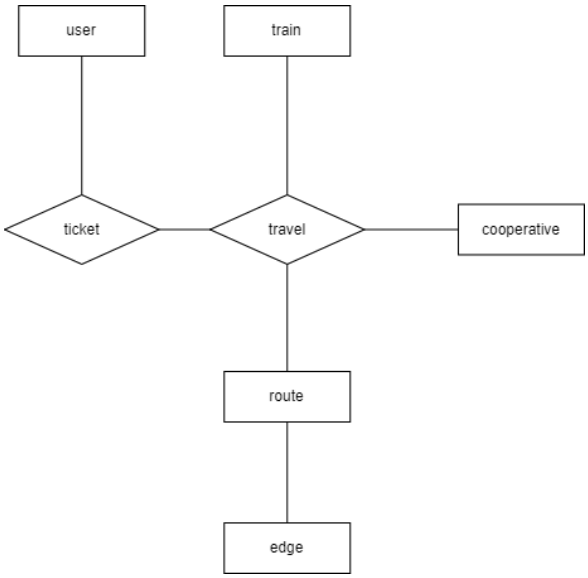
استفاده از RDBMS و مدل‌های Django امکان انجام JOIN بین جداول، گزارش‌گیری انعطاف‌پذیر و نگهداری داده‌ها به‌صورت متمرکز و پایدار را فراهم می‌کند. مدل‌ها به‌صورت کلاس‌های Python تعریف می‌شوند و روابط بین موجودیت‌ها با استفاده از ForeignKey، OneToMany و ManyToMany نمایش داده می‌شوند. به عنوان مثال:

- 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.

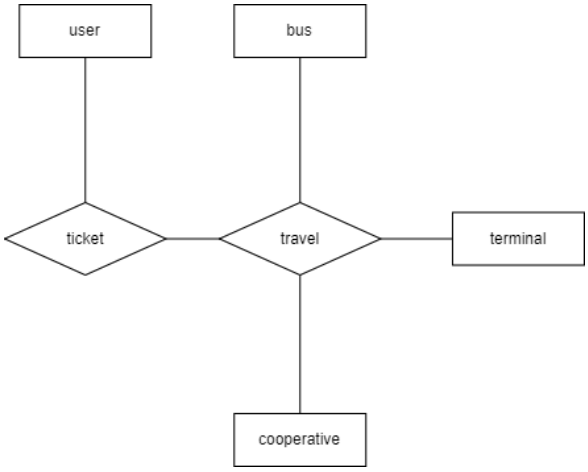
این طراحی امکان افزودن امکانات جدید مانند تخفیف‌ها، پیشنهادات ویژه و مدیریت صندلی‌ها بدون نیاز به تغییر ساختار اصلی پایگاه داده را فراهم می‌کند.

۸-۳ دیاگرام‌های کلاس Travel در اپلیکیشن اتوبوس

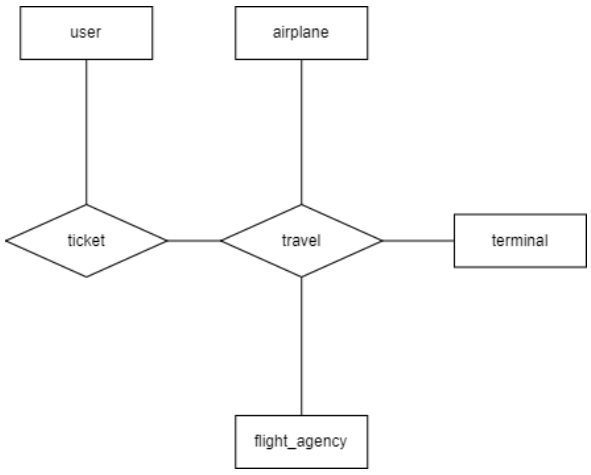
کلاس Travel هسته اصلی مدیریت سفر در اپلیکیشن اتوبوس است. این کلاس اطلاعاتی از قبیل مبدا، مقصد، تاریخ و زمان حرکت، ظرفیت صندلی‌ها، قیمت و وضعیت رزرو را در بر دارد. برای تحلیل و درک بهتر این کلاس و نقش آن در سیستم، مجموعه‌ای از دیاگرام‌های مختلف ارائه شده است.



Train Application



Bus Application



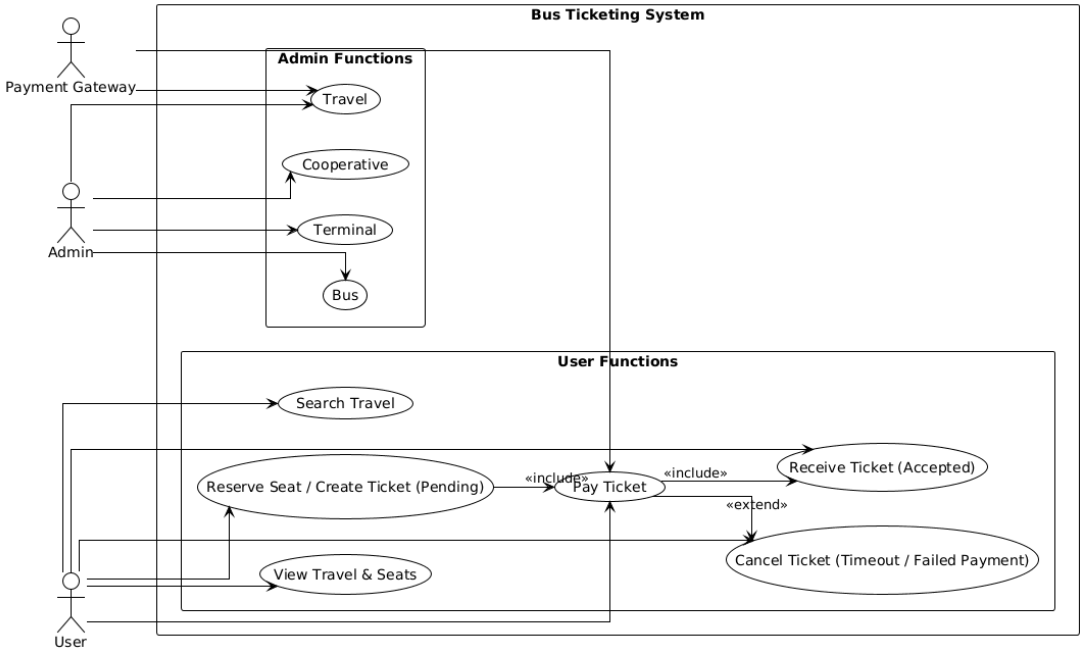
Flight Application

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

۸-۳-۱ دیاگرام موارد کاربرد (Use Case Diagram)

این دیاگرام به تعاملات کاربر و سیستم هنگام مشاهده سفرها و رزرو صندلی اشاره دارد. به طور کلی، کاربر سفر را انتخاب کرده و صندلی مورد نظر خود را مشخص می‌کند. سپس درخواست به درگاه پرداخت ارسال می‌شود. در صورتی که پرداخت موفقیت‌آمیز باشد، رزرو صندلی نهایی شده و بلیط صادر می‌گردد. در غیر این صورت، رزرو لغو خواهد شد.

برای مشاهده نمودار به شکل ۲ مراجعه کنید.



شکل ۲: دیاگرام موارد کاربرد کلاس Travel در اپلیکیشن اتوبوس

۸-۳-۲ دیاگرام کلاس (Class Diagram)

در این بخش نگاهی عمیق‌تر به مدل‌های موجود در اپلیکیشن اتوبوس داریم تا تمام فیلدهای هر کلاس مشخص شود. به‌عنوان نمونه، وضعیت صندلی‌ها به صورت JSON ذخیره می‌شود تا مدیریت و دسترسی به آن‌ها ساده‌تر باشد.

تمرکز اصلی ما بر روی کلاس Travel است. این کلاس دو نمونه از مهم‌ترین عملیات خود را در تعامل با کلاس Ticket نشان می‌دهد:

- آزادسازی صندلی‌هایی که بلیط آن‌ها لغو شده است.
- تولید و ارسال نسخه PDF بلیط برای مشتریانی که بلیط معتبر خریداری کرده‌اند.

کلاس Travel برای انجام درست وظایف خود به همکاری با سایر مدل‌های موجود در اپلیکیشن اتوبوس نیاز دارد که مهم‌ترین آن‌ها کلاس Ticket است. این همکاری شامل مواردی همچون تأیید موفق بودن خرید بلیط، ارسال نتیجه به کلاس Travel، یا لغو بلیط و اطلاع‌رسانی به کلاس Ticket می‌باشد.

برای مشاهده نمودار به شکل ۳ مراجعه کنید.

۸-۳-۳ کارت‌های CRC

کارت‌های Class–Responsibility–Collaboration (CRC) مسئولیت‌ها و همکاری‌های کلاس Travel را با سایر بخش‌های سیستم نشان می‌دهند. همان‌طور که پیش‌تر ذکر شد، کلاس Travel هسته اصلی اپلیکیشن

اتوبوس است و تمامی عملیات مربوط به سفرها و رزرو صندلی‌ها را مدیریت می‌کند.

این کارت‌ها مشخص می‌کنند که کلاس Travel مسئول چه وظایفی است و چگونه با دیگر بخش‌ها مانند صفحه پرداخت و کلاس Ticket همکاری می‌کند.

برای مشاهده کارت‌های CRC مربوط به کلاس Travel به شکل ۴ مراجعه کنید.

۸-۳-۴ دیاگرام جریان کار (Swimlane Diagram)

این دیاگرام نحوه جریان کامل فرآیند خرید بلیط را از دید کلاس Travel در اپلیکیشن اتوبوس نمایش می‌دهد.

جریان کار شامل مراحل زیر است:

- انتخاب سفر و صندلی توسط کاربر.
- ارسال درخواست رزرو به سرور و پردازش آن توسط کلاس Travel.
- ذخیره‌سازی موقت اطلاعات رزرو در پایگاه داده.
- ارسال اطلاعات پرداخت به درگاه پرداخت.
- بررسی نتیجه تراکنش و دو حالت ممکن:
  - تأیید خرید: رزرو نهایی شده، بلیط تولید و اطلاعات آن به کلاس Ticket ارسال می‌شود.
  - لغو خرید: رزرو لغو شده و صندلی دوباره در دسترس قرار می‌گیرد.

برای مشاهده دیاگرام Swimlane مربوط به کلاس Travel به شکل ۵ مراجعه کنید.

۹ 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.

۹-۱ 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.

۹-۲ 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.

۹-۳ 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.

۹-۴ 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.

۹-۵ 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.

۱۰ گزارش جلسات اسپرینت

در طول توسعه سامانه رزرو بلیط باب‌علی، تیم سه نفره ما جلسات اسپرینت کوتاه برگزار کرد تا هماهنگی، پیگیری پیشرفت و حل مشکلات پروژه به شکل منظم انجام شود. در این جلسات، تمرکز اصلی بر روی پیاده‌سازی جداگانه اپلیکیشن‌ها بود تا هر اپ بتواند عملکرد مستقلی داشته باشد و کلاس اصلی خود، یعنی Travel، بهترین خروجی و کارایی را ارائه دهد. تمام فعالیت‌ها به منظور اطمینان از عملکرد صحیح بخش رزرو صندلی، پردازش Payment و تولید PDF Ticket انجام شد. در ادامه، نمونه‌های گزارش جلسات با جزئیات بیشتر ارائه شده است.

جلسه ۱ – ۱۷ فروردین ۱۴۰۴

- **موضوعات مورد بحث:**

 تعیین اهداف اسپرینت اول، شامل پیاده‌سازی بخش جستجو و رزرو صندلی در اپلیکیشن Bus

- بررسی اولیه دیاگرام‌های Class و Use Case برای درک جریان کار و تعاملات کاربر با سیستم
- تقسیم وظایف بین اعضای تیم برای پیاده‌سازی بخش جستجو و ذخیره‌سازی وضعیت صندلی‌ها
- بررسی ابزارها و فرمت‌های مورد استفاده مانند JSON و پایگاه داده برای ذخیره‌سازی اطلاعات سفرها

- **تصمیمات:** شروع کدنویسی ماژول جستجو، ایجاد مدل‌های پایه در پایگاه داده و طراحی اولیه ساختار کلاس Travel برای مدیریت سفرها و رزرو صندلی‌ها

جلسه ۲ – ۲۰ فروردین ۱۴۰۴

- **موضوعات مورد بحث:**

- بررسی پیشرفت ماژول جستجو و بررسی صحت عملکرد ذخیره‌سازی صندلی‌ها با JSON
- هماهنگی برای طراحی کارت‌های CRC و بررسی مسئولیت‌ها و همکاری‌های کلاس Travel با سایر بخش‌ها
- شناسایی مشکلات اولیه در ارتباط کلاس Travel با کلاس Ticket و برنامه‌ریزی برای رفع آن
- بررسی تجربه کاربری اولیه در رزرو صندلی و نمایش نتایج جستجو به کاربران

- **تصمیمات:** اصلاح مدل کلاس Travel برای پشتیبانی بهتر از عملیات رزرو و لغو صندلی، تعریف توابع مدیریت صندلی‌ها و اطمینان از هماهنگی با کلاس Ticket

جلسه ۳ – ۲۳ فروردین ۱۴۰۴

- **موضوعات مورد بحث:**

- تست عملکرد ماژول رزرو و Payment، بررسی نحوه ارسال داده‌ها به درگاه پرداخت و دریافت پاسخ تراکنش
- تحلیل دیاگرام‌های Swimlane برای شناسایی جریان کامل عملیات رزرو و پردازش پرداخت و اطمینان از هماهنگی بین Frontend، Backend و پایگاه داده
- بررسی تولید PDF Ticket پس از موفقیت تراکنش و ارسال رسید به کاربران
- جمع‌بندی و آماده‌سازی گزارش اسپرینت برای ارائه به تیم و ثبت پیشرفت کارها

- **تصمیمات:** تکمیل بخش Payment، تولید و ارسال PDF Ticket، اصلاح نهایی مدل کلاس Travel و آماده‌سازی محیط برای اپلیکیشن بعدی یا اسپرینت بعدی

۱۱ استقرار

سامانه باب‌علی با معماری ماژولار و مقیاس‌پذیر طراحی شده است تا هر بخش اصلی سیستم بتواند به صورت مستقل اجرا و مدیریت شود. بخش‌های اصلی شامل Frontend، Backend، Database و مدیریت فایل‌های PDF هستند. هر بخش در قالب یک Docker Image اجرا می‌شود و می‌تواند چند نمونه (Replica) داشته باشد تا توانایی پاسخ‌دهی به تعداد کاربران زیاد فراهم شود.

۱۱-۱ استقرار Frontend

رابط کاربری (Frontend) یک تصویر مستقل دارد که مسئول ارائه صفحات و پاسخ به درخواست‌های اولیه کاربران است. این بخش می‌تواند چند Replica داشته باشد تا تجربه کاربری روان حتی در شرایط ترافیک بالا حفظ شود.

۱۱-۲ استقرار Backend

برای هر اپلیکیشن (اتوبوس، قطار، هواپیما) یک تصویر مستقل برای Backend وجود دارد. هر تصویر می‌تواند چند Replica داشته باشد تا بار درخواست‌ها به صورت یکنواخت توزیع شود و از فشار بیش از حد روی یک سرور جلوگیری گردد. این طراحی باعث می‌شود هر اپلیکیشن بتواند به شکل مستقل توسعه، نگهداری و مقیاس‌پذیری شود.

۱۱-۳ استقرار Database

پایگاه داده یک تصویر مستقل دارد و از قابلیت‌های Replication برای افزایش پایداری و کارایی بهره می‌برد. این بخش مستقل از سایر بخش‌هاست تا نگهداری، بکاپ‌گیری و مدیریت داده‌ها به راحتی انجام شود.

۱۱-۴ مدیریت فایل‌های PDF Ticket

بخش مدیریت فایل‌های PDF نیز یک تصویر مستقل دارد. این بخش مسئول تولید فایل‌های بلیط پس از تایید پرداخت، ارسال آن‌ها به کاربر و پاک‌سازی فایل‌های قدیمی است. می‌توان چند Replica برای این بخش در نظر گرفت تا فشار بالای سیستم تحمل شود.

۱۱-۵ Load Balancer و توزیع بار

یک Load Balancer میان نمونه‌های مختلف Backend قرار می‌گیرد تا درخواست‌های کاربران به شکل یکنواخت توزیع شوند. این طراحی مزایای زیر را دارد:

- جلوگیری از بارگذاری بیش از حد یک سرور
- کاهش ایجاد گلوگاه (Bottleneck)
- امکان اضافه کردن نمونه‌های جدید و افزایش مقیاس‌پذیری سیستم به سادگی

با این معماری، هر بخش سامانه به صورت مستقل مقیاس‌پذیر، قابل نگهداری و مقاوم در برابر فشار و خطا طراحی شده است. همچنین اضافه کردن امکانات جدید بدون ایجاد اختلال در سایر بخش‌ها امکان‌پذیر می‌باشد.

۱۱-۶ نحوه استقرار و مدیریت با Kubernetes

استقرار سامانه باب‌علی با استفاده از Kubernetes انجام شده است تا مدیریت مقیاس‌پذیری، Replicaها و توزیع بار به صورت خودکار و کارآمد انجام شود. هر Docker Image شامل Frontend، Backend، Database و بخش مدیریت فایل‌های PDF در Kubernetes Podهای مستقل اجرا می‌شوند.

برای مدیریت این Pods و منابع مرتبط، از فایل‌های پیکربندی YAML مختلف استفاده شده است که شامل موارد زیر می‌باشند:

- Deployment: مشخص می‌کند که هر بخش چند Replica داشته باشد و رفتار بازگردانی (Restart Policy) چگونه باشد.
- Service: برای دسترسی داخلی و خارجی به Pods استفاده می‌شود و امکان بارگذاری یکنواخت درخواست‌ها را فراهم می‌کند.
- ConfigMap / Secret: برای ذخیره تنظیمات و اطلاعات حساس سامانه به صورت امن و قابل تغییر بدون نیاز به بازسازی Image استفاده می‌شوند.
- فایل‌های پیکربندی دیگر (Ingress، Persi stentVolume، PersistentVolumeClaim، Namespace) برای مدیریت دسترسی‌ها، مسیرهای ورودی، ذخیره‌سازی پایدار و تفکیک منابع بین محیط‌های مختلف مورد استفاده قرار گرفته‌اند.

با استفاده از kubectl، تیم می‌تواند وضعیت Pods، سرویس‌ها و منابع را مشاهده و مدیریت کند، به‌روزرسانی‌ها را به شکل تدریجی اعمال نماید و در صورت نیاز به سرعت مقیاس سیستم را افزایش دهد. این روش باعث می‌شود استقرار، نگهداری و مدیریت کل سامانه ساده‌تر و قابل اعتمادتر باشد.

۱۲ 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

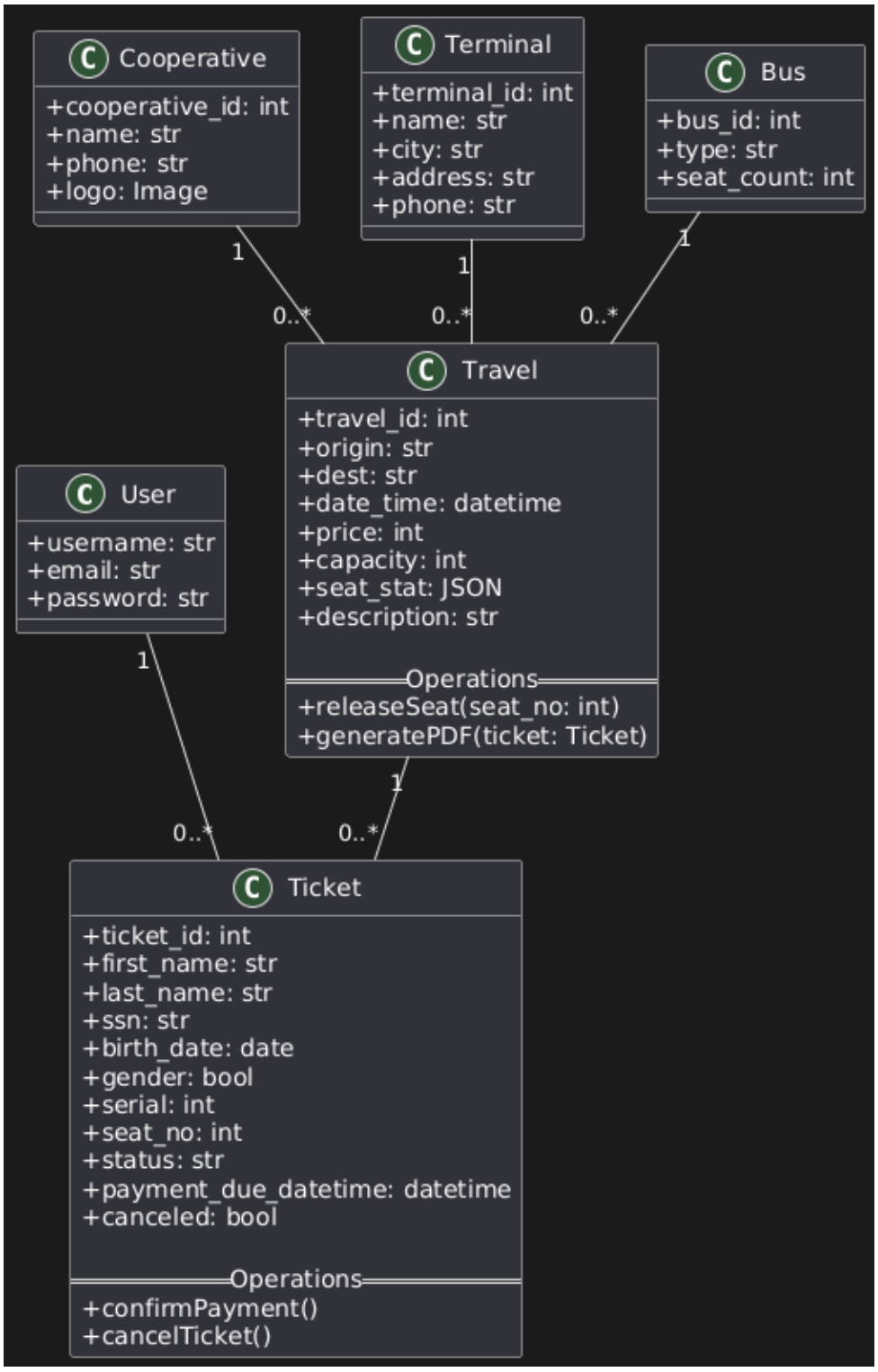
پروژه باب‌علی به‌عنوان یک سامانه جامع رزرو بلیط، با در نظر گرفتن الزامات عملکردی و غیرعملکردی، طراحی معماری ماژولار و پیاده‌سازی بر پایه الگوهای طراحی نرم‌افزار، توانسته است یک بستر پایدار، امن و مقیاس‌پذیر برای کاربران فراهم آورد.

در نسخه فعلی، سه اپلیکیشن اصلی (اتوبوس، قطار، هواپیما) پیاده‌سازی شده و امکان جستجو، رزرو، پرداخت و دریافت بلیط دیجیتال در آن‌ها فراهم است. از سوی دیگر، بهره‌گیری از روش چابک (Scrum) در مدیریت پروژه سبب شد توسعه محصول با انعطاف‌پذیری بالا و پاسخ سریع به تغییرات صورت گیرد.

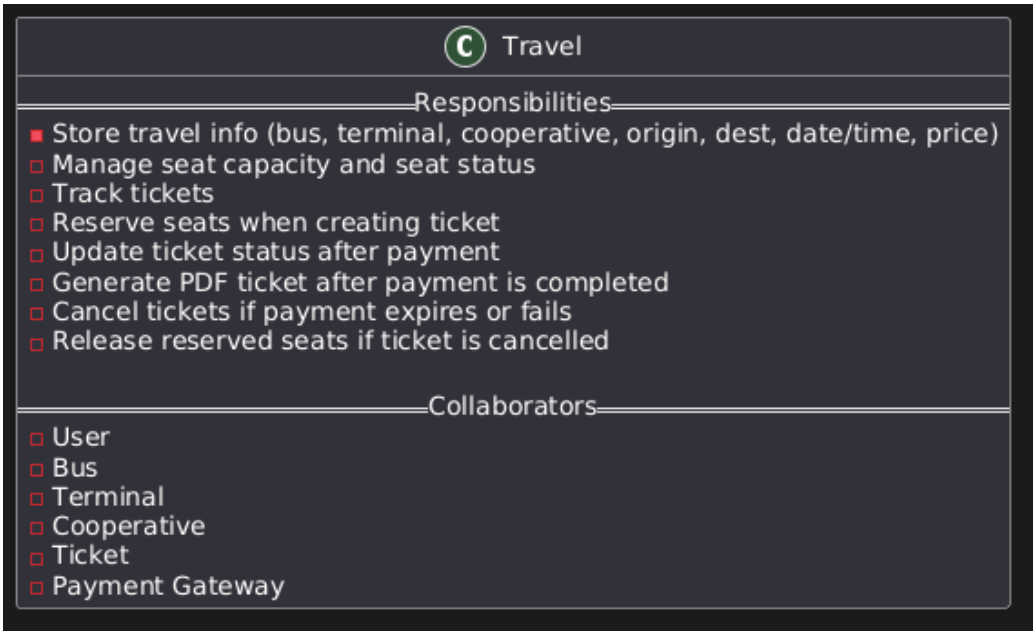
با توجه به ساختار سامانه، در آینده می‌توان قابلیت‌های پیشرفته‌ای مانند پیشنهاد هوشمند سفر، مقایسه قیمت‌ها، پشتیبانی از پرداخت‌های بین‌المللی و اتصال به سرویس‌های خارجی را نیز به آن افزود. برای دسترسی آسان‌تر به کدهای کامل پروژه و بررسی دقیق‌تر جزئیات فنی، می‌توانید به مخزن گیت‌هاب پروژه

مراجعه کنید:

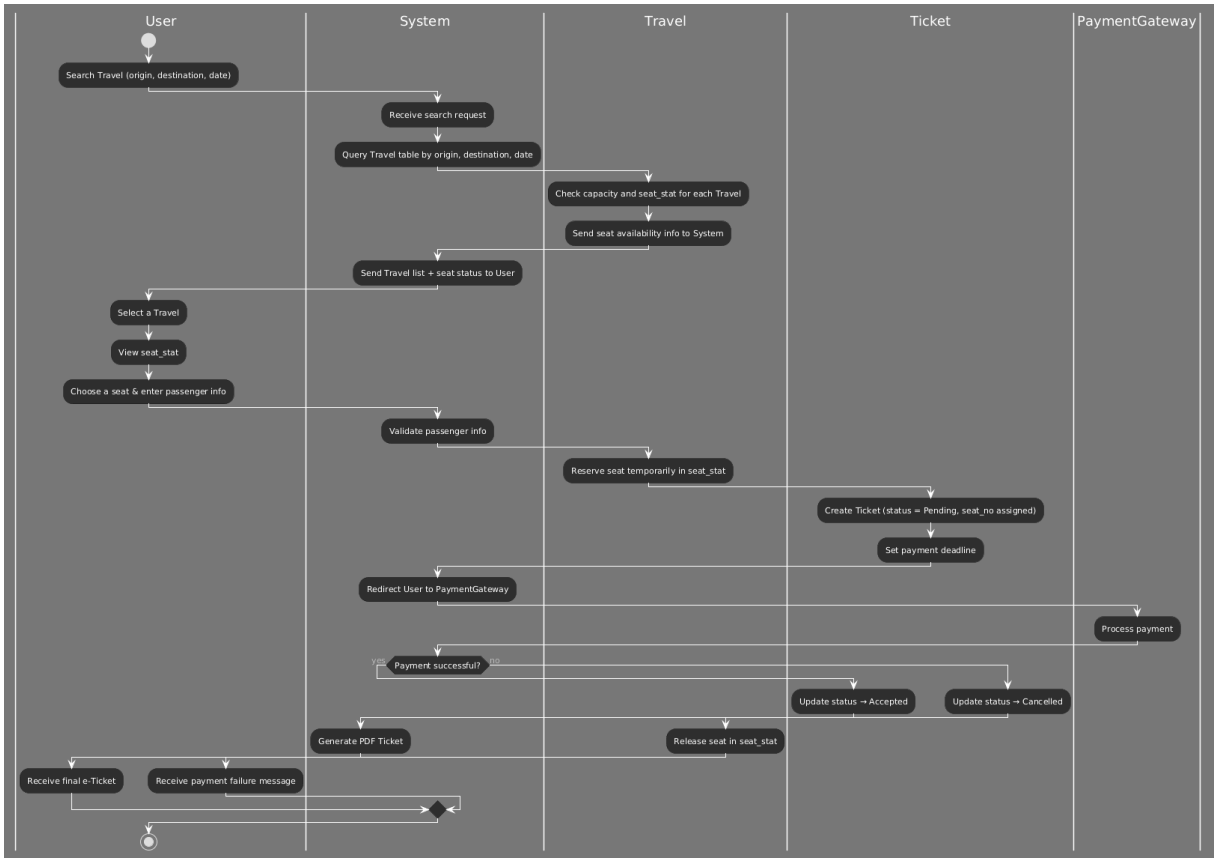
https://github.com/MMDs-team/babali



شکل ۳: دیاگرام اپلیکیشن اتوبوس



شکل ۴: کارت‌های CRC برای کلاس Travel در اپلیکیشن اتوبوس



شکل ۵: دیاگرام Swimlane مربوط به کلاس Travel در اپلیکیشن اتوبوس