

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

صنعت حمل و نقل درون شهری

پروژه درس معماری نرم افزار

سید سجاد میرزا بابایی - امیرحسین کارگران خوزانی
(۹۹۲۰۱۱۱۹-۹۹۲۱۰۱۴۲)

استاد درس

دکتر جعفر حبیبی

تحت نظارت

مهدی لشکری

نیمسال دوم تحصیلی ۱۴۰۰-۱۳۹۹

فهرست مطالب

صفحه	عنوان
سه	فهرست مطالب
۱	چکیده
فصل اول: مقدمه	
۲	۱-۱ معرفی صنعت حمل و نقل درون شهری
۳	۲-۱ تاکسی رانی و حمل و نقل درون شهری
۴	۱-۲-۱ نیازهای عملکردی Uber
۵	۲-۲-۱ نیازهای عملکردی MyWay
۷	۳-۱ اشتراک خودرو و حمل و نقل درون شهری
۸	۱-۳-۱ نیازهای عملکردی Waze Carpool
۹	۴-۱ پارکینگ های عمومی و حمل و نقل درون شهری
۹	۱-۴-۱ نیازهای عملکردی ParkWhiz
فصل دوم: دسترسی پذیری	
۱۱	۱-۲ تعریف دسترسی پذیری
۱۲	۲-۲ سناریوی عمومی دسترسی پذیری
۱۳	۳-۲ تاکتیک ها در دسترسی پذیری
۱۳	۱-۳-۲ تشخیص خطا
۱۳	۲-۳-۲ بازبانی از خطا
۱۴	۳-۳-۲ جلوگیری از خطا
۱۴	۴-۲ طراحی فهرست بازبینی برای دسترسی پذیری
۱۴	۱-۴-۲ تخصیص مسئولیت ها
۱۵	۲-۴-۲ مدل هماهنگی
۱۵	۳-۴-۲ مدل داده
۱۶	۴-۴-۲ نگاشت در میان عناصر معماری
۱۶	۵-۴-۲ مدیریت منابع
۱۶	۶-۴-۲ زمان اتصال
۱۶	۷-۴-۲ انتخاب فناوری

۱۷	۵-۲ مطالعات موردی
۱۷	۱-۵-۲ تاکسیرانی و حمل و نقل درون شهری
۱۸	۲-۵-۲ اشتراک خودرو و حمل و نقل درون شهری
۱۹	۳-۵-۲ پارکینگ های عمومی و حمل و نقل درون شهری

فصل سوم: قابلیت همکاری

۲۱	۱-۳ تعریف قابلیت همکاری
۲۲	۲-۳ سناریوی عمومی قابلیت همکاری
۲۳	۳-۳ طراحی فهرست بازبینی برای قابلیت همکاری
۲۴	۱-۳-۳ تخصیص مسئولیت ها
۲۴	۲-۳-۳ مدل هماهنگی
۲۵	۳-۳-۳ مدل داده
۲۵	۴-۳-۳ نگاشت در میان عناصر معماری
۲۵	۵-۳-۳ مدیریت منابع
۲۵	۶-۳-۳ زمان اتصال
۲۶	۷-۳-۳ انتخاب فناوری
۲۶	۴-۳ مطالعات موردی
۲۶	۱-۴-۳ تاکسیرانی و حمل و نقل درون شهری
۲۷	۲-۴-۳ اشتراک خودرو و حمل و نقل درون شهری
۲۷	۳-۴-۳ پارکینگ های عمومی و حمل و نقل درون شهری

فصل چهارم: اصلاح پذیری

۲۹	۱-۴ تعریف اصلاح پذیری
۳۰	۲-۴ سناریوی عمومی اصلاح پذیری
۳۱	۳-۴ طراحی فهرست بازبینی برای اصلاح پذیری
۳۱	۱-۳-۴ تخصیص مسئولیت ها
۳۱	۲-۳-۴ مدل هماهنگی
۳۲	۳-۳-۴ مدل داده
۳۲	۴-۳-۴ نگاشت در میان عناصر معماری
۳۲	۵-۳-۴ مدیریت منابع
۳۳	۶-۳-۴ زمان اتصال
۳۳	۷-۳-۴ انتخاب فناوری
۳۳	۴-۴ مطالعات موردی
۳۳	۱-۴-۴ تاکسیرانی و حمل و نقل درون شهری
۳۴	۲-۴-۴ اشتراک خودرو و حمل و نقل درون شهری
۳۴	۳-۴-۴ پارکینگ های عمومی و حمل و نقل درون شهری

فصل پنجم: کارایی

۳۶	۱-۵ تعریف کارایی
۳۶	۲-۵ سناریو عمومی کارایی
۳۸	۳-۵ تاکتیک ها در کارایی
۳۸	۱-۳-۵ کنترل درخواست منابع
۴۰	۲-۳-۵ مدیریت منابع
۴۱	۴-۵ طراحی فهرست بازیابی برای کارایی
۴۱	۱-۴-۵ تخصیص مسئولیت ها
۴۲	۲-۴-۵ مدل هماهنگی
۴۲	۳-۴-۵ مدل داده
۴۳	۴-۴-۵ نقشه برداری در میان عناصر معماری
۴۳	۵-۴-۵ مدیریت منابع
۴۳	۶-۴-۵ زمان اتصال
۴۳	۷-۴-۵ انتخاب فناوری
۴۴	۵-۵ مطالعات موردی
۴۴	۱-۵-۵ تاکسیرانی و حمل و نقل درون شهری
۴۴	۲-۵-۵ اشتراک خودرو و حمل و نقل درون شهری
۴۵	۳-۵-۵ پارکینگ های عمومی و حمل و نقل درون شهری

فصل ششم: امنیت

۴۷	۱-۶ تعریف امنیت
۴۸	۲-۶ سناریوی عمومی امنیت
۵۰	۳-۶ طراحی فهرست بازیابی برای امنیت
۵۰	۱-۳-۶ تخصیص مسئولیت ها
۵۰	۲-۳-۶ مدل هماهنگی
۵۱	۳-۳-۶ مدل داده
۵۱	۴-۳-۶ نگاهت در میان عناصر معماری
۵۱	۵-۳-۶ مدیریت منابع
۵۲	۶-۳-۶ زمان اتصال
۵۲	۷-۳-۶ انتخاب فناوری
۵۲	۴-۶ مطالعات موردی

فصل هفتم: قابلیت آزمون

۵۴	۱-۷ تعریف قابلیت آزمون
۵۴	۲-۷ سناریوی عمومی قابلیت آزمون
۵۵	۳-۷ تاکتیک ها در قابلیت آزمون

۵۶	۱-۳-۷ کنترل و نظارت بر وضعیت سیستم
۵۷	۲-۳-۷ محدودسازی پیچیدگی های سیستم
۵۷	۴-۷ طراحی فهرست بازبینی برای قابلیت آزمون
۵۷	۱-۴-۷ تخصیص مسئولیت ها
۵۸	۲-۴-۷ مدل هماهنگی
۵۸	۳-۴-۷ مدل داده
۵۸	۴-۴-۷ نقشه برداری در میان عناصر معماری
۵۸	۵-۴-۷ مدیریت منابع
۵۹	۶-۴-۷ زمان اتصال
۵۹	۷-۴-۷ انتخاب فناوری
۵۹	۵-۷ مطالعات موردی

فصل هشتم: قابلیت استفاده

۶۱	۱-۸ تعریف قابلیت استفاده
۶۲	۲-۸ سناریوی عمومی قابلیت استفاده
۶۳	۳-۸ طراحی فهرست بازبینی برای قابلیت استفاده
۶۳	۱-۳-۸ تخصیص مسئولیت ها
۶۳	۲-۳-۸ مدل هماهنگی
۶۴	۳-۳-۸ مدل داده
۶۴	۴-۳-۸ نقشه برداری در میان عناصر معماری
۶۴	۵-۳-۸ مدیریت منابع
۶۴	۶-۳-۸ زمان اتصال
۶۵	۷-۳-۸ انتخاب فناوری
۶۵	۴-۸ مطالعات موردی
۶۵	۱-۴-۸ تاکسیرانی و حمل و نقل درون شهری
۶۶	۲-۴-۸ اشتراک خودرو و حمل و نقل درون شهری
۶۷	۳-۴-۸ پارکینگ های عمومی و حمل و نقل درون شهری

فصل نهم: معماری نمونه مطالعاتی تاکسی آنلاین

۶۸	۱-۹ معماری میکروسرویس دامنه‌گرا
۷۲	۲-۹ قابل مشاهده بودن در مقیاس بزرگ
۷۳	۳-۹ مدیریت منابع در Uber
۷۴	۴-۹ منابع ذخیره‌سازی داده در Uber
۷۴	۵-۹ ثبت وقایع در Uber
۷۵	۶-۹ تست در Uber
۷۵	۷-۹ قابلیت اطمینان و مشاهده‌پذیری در Uber

فصل دهم : معماری نمونه مطالعاتی برنامه اشتراک سفر

- ۱۰- ۱ زمینه برنامه اشتراک سفر ۷۷
- ۱۰- ۲ معماری پیشنهادی ۷۸
- ۱۰- ۳ نیازمندی های پوشش داده شده توسط معماری ۷۸

فصل یازدهم : معماری نمونه مطالعاتی برنامه پارکینگ

- ۱۱- ۱ زمینه برنامه پارکینگ ۸۰
- ۱۱- ۲ معماری پیشنهادی ۸۱
- ۱۱- ۳ نیازمندی های پوشش داده شده توسط معماری ۸۲
- ۸۳ مراجع

چکیده

در گام اول به شرح حوزه ی پژوهشی و مروری بر نوشتار های آن پرداخته شده است و نیازمندی های کیفی به همراه نمونه کار های بین المللی مورد بررسی قرار گرفته اند. چکیده ی مستند پس از پایان فاز سوم تکمیل خواهد شد.

کلمات کلیدی: حمل و نقل درون شهری، تاکسی های برخط، پلتفرم های اشتراک خودرو، زیرساخت پارکینگ های عمومی

فصل اول

مقدمه

در این فصل به معرفی صنعت حمل و نقل درون‌شهری پرداخته می‌شود و ۳ مطالعه موردی از آن با جزئیات بیشتری مورد بررسی قرار خواهند گرفت. در ادامه در فصل‌های بعد به بررسی نیازهای غیر عملکردی این ۳ مطالعه موردی پرداخته می‌شود. ترتیب و محتوای ارائه این نیازهای غیر عملکردی بر اساس کتاب [۴۴] انتخاب شده است.

۱-۱ معرفی صنعت حمل و نقل درون شهری

برای چندین دهه، برنامه ریزی و حمل و نقل شهری در اطراف خودرو شخصی تکامل یافته است که منجر به مشکلاتی از جمله ازدحام، صدا، آلودگی و غیره شده است. همانطور که وارد دهه جدیدی می‌شویم، تحول دیجیتال همچنان باعث رشد صنعت می‌شود. معرفی فن‌آوری‌های نوظهور و فرآیندهای تجاری، شیوه لجستیک^۱ و حمل و نقل را تغییر ساختار داده است و این روند احتمالاً در سال‌های آتی نیز ادامه خواهد داشت، خصوصاً که نوآوری در فن‌آوری منجر به رشد پایدار خواهد شد.

صنعت حمل و نقل امروزه همانند زنجیر صنایع مختلف را بهم متصل می‌کند؛ شرکت‌ها و سازمان‌های

¹Logestic

بزرگی بر پایه ی پیشرفت های اخیر صنعت حمل و نقل فرصت ظهور و خودنمایی پیدا کرده اند و سازمان های بزرگی نظیر Uber در آمریکا، cab OLA در هند و DiDi در چین فراتر از مرز های ملی ظاهر شده و در سراسر جهان شروع به ارائه ی خدمات نموده اند.

در این فصل ابتدا به تاکسی رانی آنلاین و سهم آن از حمل و نقل عمومی از کل صنعت حمل و نقل عمومی درون شهری خواهیم پرداخت؛ سپس روش های اشتراک خودرو^۱ که به تازگی در حال پیدا کردن جایگاه خود در صنعت حمل و نقل هستند را مورد بررسی قرار خواهیم داد و در پایان این فصل پیرامون یکی از محیط ترین زیرساخت های شهری یعنی پارکینگ های عمومی و ارتباط آن ها به عنوان یک زیرساخت شهری با صنعت حمل و نقل عمومی درون شهری صحبت خواهیم کرد.

۱-۲ تاکسی رانی و حمل و نقل درون شهری

تاکسی رانی برخط^۲ امروزه در بسیاری از کشور ها سهم زیادی از بازار را در اختیار خود گرفته اند؛ برای مثال Uber تا کنون فعالیت خود را به ۷۰ کشور گسترش داده است و میلیون ها راننده برای این غول بزرگ حمل و نقل درون شهری فعالیت می کنند و روزانه میلیون ها سفر درون شهری بر بستر این شرکت انجام می شود.

شاید استفاده ی راحت مهم ترین نیازی باشد که شرکت هایی نظیر Uber به آن پاسخ داده اند؛ تنها کافی است برنامه ای را بر روی تلفن همراه خود نصب داشته باشید و در کمتر از چند دقیقه نزدیک ترین راننده در حال حرکت به سمت شما برای خدمت رسانی خواهد بود. همچنین راه حل هایی که این شرکت ها در زمینه ی پرداخت هزینه ی سفر، تضمین امنیت سفر و کاهش هزینه های سفر ارائه داده اند بر استقبال هر چه بیشتر جامعه از این پلتفرم^۳ های تاکسی برخط^۴ افزوده است.

در کنار Uber، از برنامه های جهانی تاکسی رانی برخط می توان به DiDi که عمده فعالیت آن در چین است و همچنین cab OLA که در هند فعالیت می کند اشاره کرد.

یکی از موارد مطالعه در این پژوهش، پروژه MyWay است. MyWay یک پلتفرم مدیریت منابع در صنعت حمل و نقل هوشمند در شهر های اروپایی است. هدف این پروژه این است که به طور یکپارچه به ادغام کارآمد و منسجم سرویسهای حمل و نقل مکمل، در کل زنجیره سفر های درون شهری بپردازد. MyWay یک اپ عمومی در حوزه حمل و نقل است تلاش کرده که چندین ویژگی در رابطه تاکسی رانی آنلاین، اشتراک گذاری خودرو یا استفاده از اتوبوس را در کنار هم برای برنامه ریزی سفر استفاده کند که بیشتر در این گزارش ما بر روی قسمت

¹ CarPool

² Ride sharing

³ Platform

⁴ Online

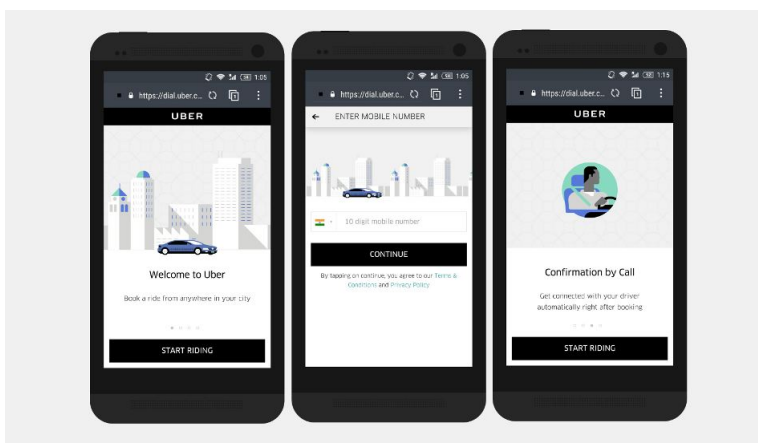
تاکسی رانی آنلاین آن تمرکز می‌کنیم.

در پایان این بخش لازم است به این نکته اشاره شود که با ورود شرکت های بزرگ تاکسی رانی آنلاین به شهر ها، کسب و کار های حمل و نقل سنتی آسیب می بینند و همچنین استفاده از وسایل حمل و نقل عمومی نیز کاهش خواهد یافت که آثار مخرب زیست محیطی و اجتماعی از جمله معایب گسترش این سیستم هاست.

۱-۲-۱ نیازهای عملکردی Uber

مهمترین نیازهای عملکردی در اپ Uber را می‌توان به شرح زیر داشت:

- کاربران باید بتوانند با نام و شماره تلفن و زبان مورد نظر ثبت نام کنند در شکل ۱-۱ روند این ثبت نام نشان داده شده است که یک SMS برای تایید به شماره تلفن فرستاده می‌شود و سپس نحوه پرداخت پیشنهادی انتخاب می‌شود.



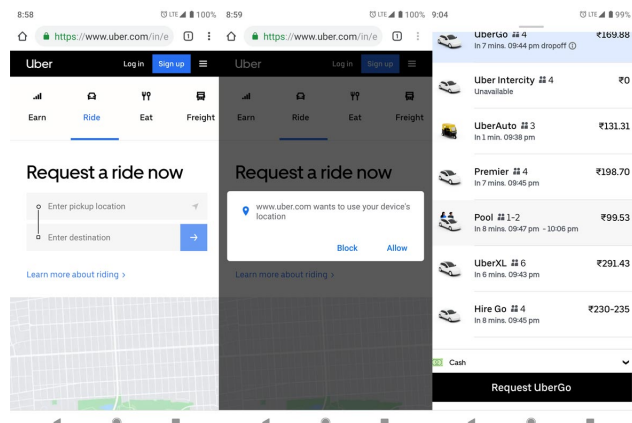
شکل ۱-۱: ثبت نام در اوبر، عکس از themobileindian.com

- قابلیت رزرو یک سرویس تاکسی: مبدا و مقصد را وارد کرده و به کاربر انواع سرویس ها و مبلغ آن ها را پیشنهاد می‌دهد، این موارد در شکل ۱-۲ نمایش داده شده است. رانندگان نزدیک تر نیز اطلاعات سفر کاربران را دریافت می‌کنند و آن را قبول یا رد می‌کنند.

- دسترسی به موقعیت جغرافیایی و پیدا کردن آن بر روی نقشه در تمامی دستگاه های پشتیبانی شده

- نشان دادن نقشه مسیر هم به کاربر و هم به راننده و راهنمایی گام به گام

- فرستادن پوش نوتیفیکیشن مواقعی که راننده ای درخواست را اکسپت می‌کند یا به نزدیکی محل می‌رسد یا به هر دلیلی سفر لغو می‌شود.



شکل ۱-۲: رزرو تاکسی در اوبر، عکس از androidinfotech.com

- محاسبه قیمت دقیق بر اساس مبدا و مقصد و ساعت حرکت و نوع ماشین
- قابلیت اشتراک سفر و مسیر خود با دیگران و دنبال کردن در زمان واقعی
- قابلیت اضافه کردن نقاط توقف در مسیر
- استفاده از سرویس‌های متنوع پرداخت نظیر paypal ، card credit و ...
- قابلیت برنامه‌ریزی سفر تا ۳۰ روز مانده
- امکان همگام‌سازی تقویم قرارهای ملاقات با برنامه

۱-۲-۲ نیازهای عملکردی MyWay

طبق مستندات [۲۲] برای MyWay می‌توان ۱۰۰ نیازمندی عملکردی بیان کرد که این نیازمندی‌ها را می‌توان در ۴ دسته کلی زیر طبقه‌بندی کرد

- برنامه‌ریزی سفر و دنبال کردن^۱
- خدمات تحرک^۲ و آگاهی از منابع
- نمایه کردن^۳ کاربران
- مدیریت سیستم

از مهمترین نیازمندی‌های این برنامه نیز می‌توان به موارد زیر اشاره کرد:

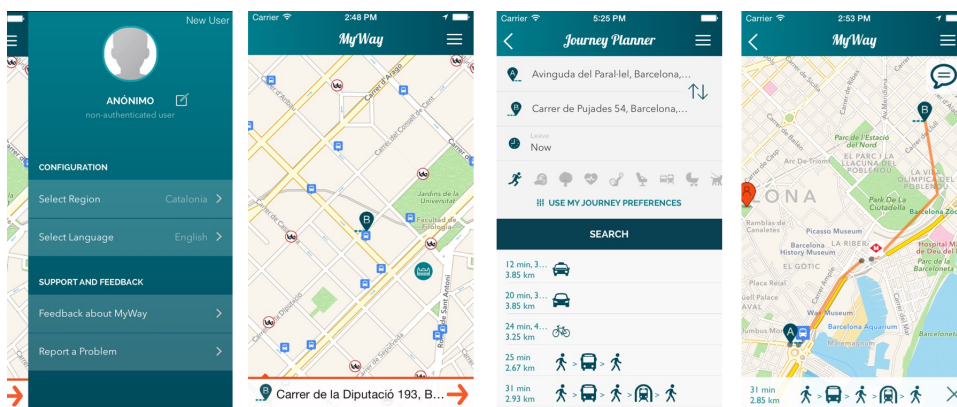
¹Tracking

²Mobility

³Profiling

- برنامه‌ریزی سفر
- رزرو یک سرویس
- اطلاعات شهر
- نمایه کردن کاربر

در شکل ۱-۳ تعدادی از صفحات نسخه موبایل نمایش داده شده است. از آنجا که نوشتن ریز جزئیات



شکل ۱-۳: نمایی از نسخه موبایل اپ Myway تصویر از appadvice.com

نیازمندی‌های عملکردی هدف این گزارش نیست بنابراین ۴ دسته کلی نیازهای عملکردی را به ۱۴ دسته تقسیم می‌کنیم و آن ۱۴ دسته را همراه با توضیحات گزارش می‌کنیم.

برنامه‌ریزی سفر و دنبال کردن:

- برنامه‌ریزی سفر: نیاز است که کاربر با استفاده از موقعیت مکانی خود، آدرس‌ها، انتخاب نقطه بر روی نقشه، نقاط مورد علاقه یک سفر را برنامه‌ریزی کند و برنامه نیز مسیری بین مبدا و مقصد پیشنهاد دهد و اطلاعات ترافیک را بازتاب دهد و آن سفر را به پروفایل کاربر اضافه کند و ...

- رزرو کردن یک سرویس: رزرو کردن تاکسی، قطار، اتوبوس، اسکوتر، جای پارک
- دنبال کردن یک سفر: دوباره فیدبک‌دهی و محاسبه به کاربر در رابطه با سفر با استفاده از دنبال کردن آن با GPS

- فیدبک‌دهی آنلاین از شرایط جاده و محدودیت‌ها
- فیدبک دادن به سفر: کاربر می‌تواند فیدبک خود را ارائه دهد.

خدمات تحرک و آگاهی از منابع:

- اطلاعات شهر: ذخیره نقاط مورد علاقه در شهر و اطلاعات دادن در مورد رستوران‌ها و ارزیابی آن‌ها و سرویس‌های زمان‌بندی شده تحرک و اطلاعات در مورد توزیع آن‌ها و ...
- فیدبک دادن به Myway در مورد سرویس‌ها
نمایه کردن کاربر:
- نمایه کاربر: اضافه کردن اطلاعات عمومی در نمایه کاربر رو تغییر آن‌ها، علایق و ترجیحات و ...
- گروه کردن کاربران: بیشتر برای اشتراک‌گذاری خودرو استفاده می‌شود و کاربر می‌تواند یک جامعه اشتراک‌گذاری خودرو درست کند و به آن اضافه و حذف شود و با افراد آن جامعه برنامه سفر خود را به اشتراک بگذارد.
- مدیریت سیستم:
- تحلیل: جمع‌آوری تاریخچه‌ها و انجام عملیات‌های آماری و تحلیل داده‌ها
- احراز هویت کاربر: تعیین نقش و احراز هویت کاربر با استفاده از سیستم احراز هویت
- مدیریت داده: مدیریت شهرها و تنظیمات و تغییر داده‌ها توسط ادمین‌های Myway
- درگاه برای اپراتورهای سرویس‌ها

۳-۱ اشتراک خودرو و حمل‌ونقل درون شهری

در سال‌های اخیر دغدغه‌های اقتصادی، زیست محیطی و اجتماعی جوامع را به سمت استفاده مشترک از منابع در دسترس سوق داده است؛ صنعت حمل‌ونقل نیز از این قاعده مستثنا نبوده و در جهت استفاده مشترک از زیرساخت‌های موجود گام‌های موثری را برداشته است.

اشتراک خودرو^۱ به معنی به اشتراک گذاشتن خودرو توسط افراد در سفرهایی با مقاصد نزدیک به هم است و در نتیجه سبب می‌شود تا افراد کمتری برای رفتن به مقاصد خود از خودروهای شخصی استفاده کنند.

با استفاده بیشتر از یک فرد از وسیله نقلیه، هزینه‌های سفر هر فرد نظیر هزینه‌های سوخت، عوارض رانندگی و استرس رانندگی کاهش خواهد یافت.

رانندگان و مسافران سفرها را از طریق یکی از چندین رسانه موجود ارائه می‌دهند و جستجو می‌کنند. آنها پس از یافتن افراد با سفرهای مشابه با یکدیگر تماس می‌گیرند تا جزئیات سفر را ترتیب دهند. هزینه‌ها، نقاط

^۱ Car Pooling

ملاقات و سایر جزئیات مانند فضای وسایل اضافه همراه افراد نیز از پیش توافق شده است. سپس افراد طبق برنامه ریزی از پیش انجام شده با یکدیگر ملاقات نموده و سفر را انجام می دهند.

هم اکنون استارت آپ های زیادی در زمینه ی اشتراک گذاری خودرو در سرتاسر دنیا شروع به فعالیت کرده اند که می توان برای مثال از Waze Car ، BlaBla Car و Ride Connect نام برد.

از آنجا که استفاده از اشتراک خودرو تعداد خودروهای مورد نیاز مسافران را کاهش می دهد ، اغلب با مزایای متعددی در جامعه همراه است از جمله:

- کاهش در مصرف انرژی و انتشار گازهای گلخانه ای
- کاهش ترافیک سطح شهری
- کاهش تقاضای زیرساخت پارکینگ

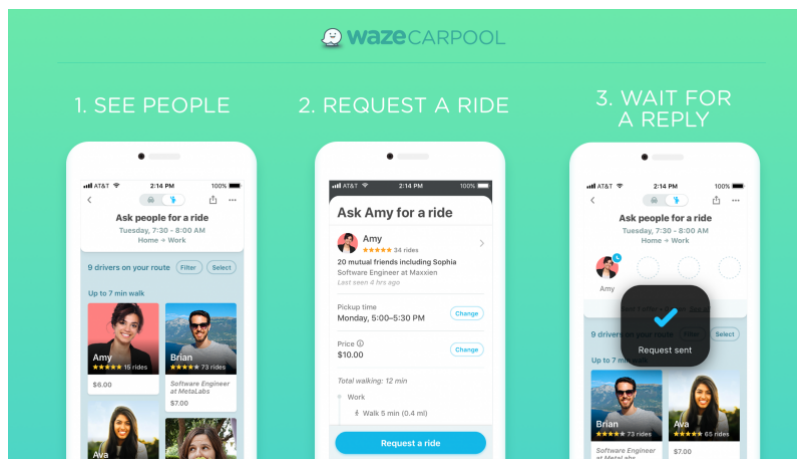
۱-۳-۱ نیازهای عملکردی Waze Carpool

اشتراک گذاری خودرو به دو صورت در شرکت waze انجام می شود. یک آن که در خود اپلیکیشن اصلی waze یک گزینه جدا برای اشتراک گذاری خودرو وجود دارد. دو، برنامه مجزا این شرکت برای اشتراک گذاری خودرو است [۴۰].

هر یک از افراد می توانند در نقش راننده یا مسافر ظاهر شوند. برای مسافر تنها کافی است زمان اشتراک گذاری خودرو را تنظیم کنند، مشاهده کنند که چه کسی در آن زمان رانندگی می کند و از او درخواست عضویت کنند و سپس با یکدیگر تایید کنند. برای رانندگی نیز کافیست که زمان، تعداد صندلی ها و مسیر خود را مشخص کنید از مسافران دعوت کنید و اشتراک گذاری خودرو را شروع کنید. نحوه درخواست از راننده برای برنامه موبایل در تصویر ۱-۴ نمایش داده شده است.

از مهمترین نیازمندی های این برنامه می توان به موارد زیر اشاره کرد:

- ثبت نام و ورود به برنامه
- وارد کردن اطلاعات در نمایه و تغییر آنها
- تنظیم زمان اشتراک گذاری خودرو و مسیر برای راننده و مسافر
- درخواست به رانندگان
- مشاهده درخواست مسافران و قبول و رد آنها



شکل ۱-۴: درخواست خودرو برای هم‌اشتراک‌گذاری، عکس از sourceWaze

- چت و گرفتن تایید بین راننده و مسافر
- پرداخت با روش‌های مختلف
- تایید رانندگان با سرویس‌های ثالث

۱-۴ پارکینگ‌های عمومی و حمل‌ونقل درون شهری

برای افرادی که در شهرهای بزرگ زندگی می‌کنند یافتن پارکینگ مقرون به صرفه می‌تواند یک چالش باشد؛ طی تحقیق که توسط IBM انجام شد محققان به این واقعیت پی بردند که نزدیک به ۳۰٪ از ترافیک شهر به رانندگانی که بدنبال پارکینگ هستند، نسبت داده می‌شود. خوشبختانه، برنامه‌های مختلفی برای کمک به افراد در پیدا کردن و مقایسه نقاط پارک مناسب وجود دارند که برخی از آن‌ها حتی امکان رزرو پارکینگ را در اختیار کاربران خود قرار می‌دهند.

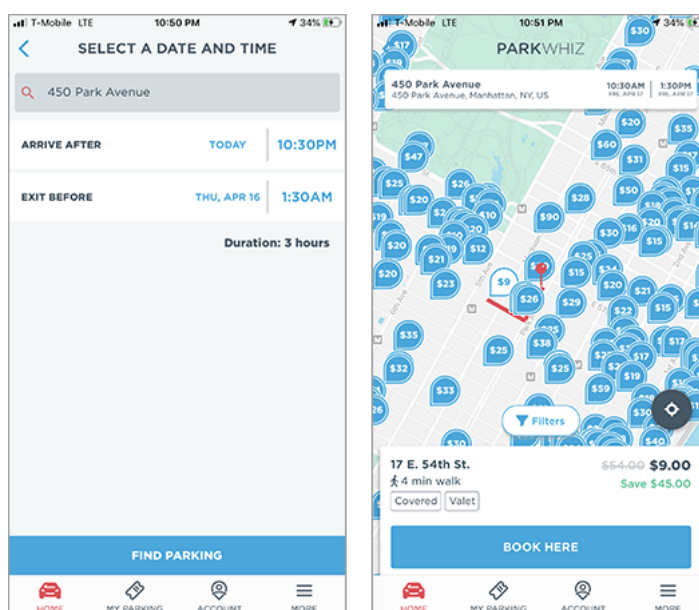
از جمله برنامه‌هایی که در زمینه‌ی پارکینگ‌های درون شهری در آمریکا فعالیت می‌کنند می‌توان به Best Parking و یا ParkWhiz اشاره کرد.

۱-۴-۱ نیازهای عملکردی ParkWhiz

از مهمترین نیازمندی‌های این برنامه می‌توان به موارد زیر اشاره کرد:

- ثبت نام و ورود به برنامه
- وارد کردن اطلاعات در نمایه و تغییر آن‌ها
- پیدا کردن پارکینگ‌ها برای الان یا برای بعد ، مقایسه قیمت آن‌ها و فاصله آن‌ها تا مقصد اصلی کاربر

- رزرو کردن پارکینگ در ساعت دلخواه
 - اتصال به سرویس‌های مسیریابی برای رسیدن به پارکینگ
 - پرداخت راحت بوسیله کارت‌های اعتباری یا دروازه‌های پرداخت دیجیتال
 - خروجی گرفتن از رزرو انجام شده و مشاهده اطلاعات رزرو
 - پنل ادمین برای مدیریت پارکینگ و ایجاد تغییرات و عملیات‌های مدیریت و گزارش‌گیری
- در شکل ۱-۵ نحوه پیدا کردن پارکینگ و مقایسه آن‌ها بر اساس قیمت و فاصله نمایش داده شده است.



شکل ۱-۵: جستجو پارکینگ و مقایسه، عکس از [۲۴]

فصل دوم

دسترسی پذیری

۲-۱ تعریف دسترسی پذیری

دسترسی پذیری^۱ به خاصیتی از نرم افزار اطلاق می شود که نرم افزار در لحظه، حاضر باشد و از آمادگی انجام وظایف خود برخوردار باشد [۴۱]. در واقع دسترسی پذیری بر پایه ی مفهوم قابلیت اطمینان^۲ بنا شده با این تفاوت که مفاهیم ترمیم^۳ نیز به آن افزوده شده است.

به طور دقیق تر، قابلیت دسترس پذیری به توانایی سیستم برای پنهان کردن یا اصلاح شکست ها اشاره دارد؛ به طوری که مجموع مدت زمان قطع خدمات از یک مقدار لازم در یک بازه زمانی مشخص بیشتر نشود. شکست از زاویه دید سیستم یا ناظر انسانی در محیط تعریف می شود و به معنای انحراف سیستم از مشخصات^۴ آن است به گونه ای که این انحراف قابل مشاهده باشد. آنچه به شکست منجر می شود، خطاست و معمار نرم افزار همواره سعی می کند تا با درک ریشه های شکست سیستمی مقاومت در برابر خطا^۵ را بنا کند.

از جمله مواردی که دغدغه ی معماری سیستم در زمینه ی دسترسی پذیری است می توان به موارد زیر اشاره

کرد:

¹Availability

²Reliability

³Recovery

⁴Specification

⁵Resilient

- چگونه خطاهای سیستم باید شناسایی شوند؟
- چقدر ممکن است خطاهای سیستم رخ دهند و عواقب رخداد یک خطا چیست؟
- یک سیستم چقدر می تواند خارج از دسترس باشد؟
- چگونه می توان از خطا و در نتیجه شکست نرم افزار جلوگیری کرد؟
- در زمان رخداد خطا چه اقداماتی لازم هستند؟

۲-۲ سناریوی عمومی دسترسی پذیری

سناریوی عمومی دسترسی پذیری به صورت زیر است:

- منبع محرک^۱ : چه درون و چه بیرون: افراد، سخت افزار، نرم افزار، محیط فیزیکی
- محرک^۲ : خطا: حذف، خرابی، زمان بندی نادرست، پاسخ نادرست
- محصول^۳: پروسسورهای سیستم، کانال های ارتباطی، ذخیره سازی مداوم، فرآیندها
- محیط^۴: عملکرد عادی، راه اندازی، خاموش کردن، حالت تعمیر، عملکرد تخریب شده، عملکرد بیش از حد
- پاسخ^۵ : جلوگیری از این که یک خطا منجر به یک شکست شود. به این منظور نیاز است که ابتدا خطا شناخته شود. در صورت وجود خطا تاریخچه آن حفظ شود و سیستم ها و آدم های مرتبط اطلاع داده شود. همچنین نیاز است که بازایی پس از انجام خطا نیز انجام شود که شامل غیرفعال کردن منبعی که باعث خطا شده است، غیرفعال کردن تا زمانی که تعمیر تکمیل شود، تعمیر خطا و شکست و عمل کردن در مد کاهش داده شده^۶ تا زمانی که تعمیر تکمیل شود.
- اندازه گیری پاسخ^۷: موارد بسیاری را می توان در نظر گرفت که به مقدار زمانی که سیستم در دسترس است اشاره دارد. از جمله آن ها می توان به درصد در دسترسی پذیری، زمانی که تا تشخیص خطا طول می کشد، زمانی که تا تعمیر طول می کشد، مقدار زمانی که سیستم در حالت کاهش یافته عمل می کند اشاره کرد.

¹Source of stimulus

²Stimulus

³Artifact

⁴Environment

⁵Response

⁶degraded mode

⁷Response Measure

۲-۳ تاکتیک‌ها در دسترسی پذیری

تاکتیک‌ها در دسترس پذیری به گونه ای طراحی شده است که سیستم را قادر می‌سازد تا خطاهای سیستم را تحمل کند تا سرویس ارائه شده توسط سیستم با مشخصات آن مطابقت داشته باشد و به شکست ختم نشود. تاکتیک‌های دسترسی پذیری را می‌توان به سه دسته‌ی تشخیص خطا^۱، بازیابی از خطا^۲ و جلوگیری از خطا^۳ تقسیم بندی کرد.

۲-۳-۱ تشخیص خطا

اولین قدم برای جلوگیری از شکست تشخیص خطاست. به منظور تشخیص خطا سیستم می‌تواند از روش‌های پینگ^۴، اکو^۵، نظارت^۶ Heartbeat، Timestamp، عقل سنجی^۷، نظارت بر شرایط^۸، رای دادن^۹، شناسایی استثنا^{۱۰}، خودآزمونی^{۱۱} استفاده کرد [۴۳].

۲-۳-۲ بازیابی از خطا

تاکتیک‌های بازیابی از خطاها در قالب روش‌های آماده‌سازی و ترمیم و تکنیک‌های بازآفرینی سیستم پیاده سازی می‌شوند. روش‌های بازآفرینی دغدغه‌ی راه اندازی مجدد یک سیستم شکست خورده را دارند. روش‌های ترمیم افزونگی فعال^{۱۲}، افزونگی منفعل^{۱۳}، پشتیبان^{۱۴}، کنترل استثنا^{۱۵}، بازگشت به عقب^{۱۶}، به روزرسانی نرم‌افزار^{۱۷}، امتحان مجدد^{۱۸}، نادیده گرفتن رفتار دارای خطا، تخریب^{۱۹}، پیکربندی^{۲۰} می‌باشند. و روش‌های بازآفرینی سیستمی که پس از شکست از اصلاح شده است، شامل سایه^{۲۱}، هماهنگ سازی مجدد^{۲۲}، راه اندازی مجدد افزایشی^{۲۳} است.

¹ fault detection

² fault recovery

³ fault prevention

⁴ Ping

⁵ Echo

⁶ Monitor

⁷ Sanity Checking

⁸ Condition Monitoring

⁹ Voting

¹⁰ Exception Detection

¹¹ self-test

¹² Active redundancy

¹³ Passive redundancy

¹⁴ Spare

¹⁵ Exception handling

¹⁶ Rollback

¹⁷ Software Upgrade

¹⁸ Try

¹⁹ The degradation

²⁰ Reconfiguration

²¹ The Shadow

²² State resynchronization

²³ Escalating restart

۲-۳-۳ جلوگیری از خطا

به جای شناسایی خطاها و سپس تلاش برای بازبازی آنها، در این تاکتیک سیستم سعی می کند تا از خطا جلوگیری کند. هر چند انجام چنین کاری در نگاه اول دشوار بنظر می رسد اما در بسیاری از موارد انجام چنین کاری ممکن است. از روش های حذف از سرویس^۱، تراکنش ها^۲، مدل پیش بینی^۳، پیشگیری از استثنا^۴، افزایش مجموعه ی صلاحیت ها^۵ به این منظور می توان استفاده کرد.

۲-۴ طراحی فهرست بازبینی برای دسترسی پذیری

فهرست بازبینی زیر برای پشتیبانی از فرایند طراحی و تجزیه و تحلیل نیاز کیفی در دسترس بودن طراحی شده است.

۲-۴-۱ تخصیص مسئولیت ها

در زمان تخصیص مسئولیت ها باید^۶ به موارد زیر توجه شود:

- مسئولیت های سیستم را که باید بسیار در دسترس باشند، باید به طور دقیق مشخص شوند.
- اطمینان حاصل شود که مسئولیت های اضافی برای تشخیص حذف، خرابی، زمان بندی نادرست یا پاسخ نادرست اختصاص داده شده است.
- اطمینان حاصل شود وظائف زیر به بخش هایی از سیستم تخصیص داده شده است:

- ثبت^۷ خطاها

- اطلاع رسانی به موجودیت های موجود در سیستم

- غیرفعال سازی منشا خطا

- خارج شدن از دسترس به صورت موقت

- اصلاح یا پوشاندن^۸ خطا

- فعالیت سیستم به حالت کاهش یافته^۹ برده شود

¹Removal from service

²Transactions

³Predictive Model

⁴Exception prevention

⁵Increase competence set

⁶Allocation of Responsibilities

⁷Log

⁸mask

⁹Degraded

۲-۴-۲ مدل هماهنگی

پس از تعیین مهم‌ترین قسمت‌های سیستم که باید بسیار در دسترس باشند، می‌بایست موارد زیر در نظر گرفته شوند:

- اطمینان حاصل شود که مکانیسم‌های هماهنگی می‌توانند حذف، خرابی، زمان‌بندی نادرست یا پاسخ نادرست را تشخیص دهند. برای مثال در نظر بگیرید که آیا هماهنگی نرم‌افزار تا کسی آنلاین در شرایط ارتباطات ضعیف کار خواهد کرد؟
- اطمینان حاصل شود که مکانیزم‌های هماهنگی امکان ثبت گزارش از خطا، اطلاع رسانی به موجودیت‌های مناسب، غیرفعال کردن منبع ایجاد کننده خطا، رفع یا پوشاندن خطا و یا عملکرد در حالت کاهش یافته را دارند.
- اطمینان حاصل شود که مدل هماهنگی از جایگزینی مصنوعات استفاده شده مثل پردازنده‌ها، کانال‌های ارتباطی، سخت‌افزارهای ذخیره‌سازی و فرایندها پشتیبانی می‌کند. به عنوان مثال، آیا جایگزینی سرور در یک سیستم پارکینگ عمومی اجازه می‌دهد سیستم به کار خود ادامه دهد؟
- تعیین کنید که آیا این هماهنگی در شرایط ارتباطات کاهش یافته، هنگام راه‌اندازی^۱ خاموش شدن^۲، در حالت تعمیر^۳ یا تحت کار بیش از حد^۴ به درستی فعالیت می‌کند. به عنوان مثال، مدل هماهنگی در نرم‌افزار اشتراک خودرو چقدر اطلاعات از دست رفته^۵ را تحمل می‌کند و با چه عواقبی؟

۲-۴-۳ مدل داده

لازم است ابتدا مشخص شود که کدام قسمت از سیستم باید همواره در دسترس باشد سپس، در این قسمت‌ها، تعیین شود که انتزاع داده‌ها، همراه با عملیات^۶ هایی که برای آن‌ها تعریف شده‌اند، می‌توانند باعث کدوم یک از خطاهای حذف، خرابی^۷، رفتار نادرست در زمان بندی یا پاسخ نادرست شوند. همچنین برای این داده‌های انتزاعی و عملکردها و خصوصیات آن‌ها، اطمینان حاصل شود که در صورت لزوم می‌توانند به طور موقت غیرفعال شوند؛ به عنوان مثال، اطمینان حاصل شود زمانی که سرور مدیریت پارکینگ عمومی موقتاً در دسترس نیست، درخواست‌های نوشتن بر روی دیسک در حافظه‌ی نهان ذخیره می‌شوند تا پس از بازگشت

¹Startup

²Shutdown

³Repair mode

⁴OverLoaded

⁵Lost information

⁶Operation

⁷crash

مجدد سرور، پردازش فرآیند آن‌ها ادامه پیدا کند.

۴-۴-۲ نگاشت در میان عناصر معماری

مشخص شود کدام بخش از سیستم نظیر پردازنده‌ها، کانال‌های ارتباطی و یا سایر بخش‌ها می‌تواند منجر به ایجاد خطا در سیستم شود. همچنین اطمینان حاصل شود نگاشت میان عناصر معماری از انعطاف کافی جهت بهبود^۱ پس از شکست برخوردار باشد.

۵-۴-۲ مدیریت منابع

در این بخش باید مشخص شود چه منابع حیاتی برای ادامه کار در صورت وجود خطا لازم است و باید از وجود منابع کافی برای ثبت^۲ خطا، اطلاع‌رسانی خطا به سایر موجودیت‌ها، غیرفعال‌سازی منشا خطا و تعمیر و بازیابی سیستم از آن، اطمینان حاصل شود.

زمان در دسترس بودن منابع حیاتی مشخص شود و تعیین شود چه منابعی باید در شرایط خاص همچون زمانی که سیستم با خطا مواجه شده و یا در حالت کاهش یافته در حال فعالیت است، باید در دسترس باشند. به عنوان مثال، اطمینان حاصل کنید که صفهای ورودی در سیستم اشتراک خودرو به اندازه کافی بزرگ هستند تا در صورت خرابی سرور پیامهای پیش‌بینی شده را بافر^۳ کند تا پیامها برای همیشه از بین نروند.

۶-۴-۲ زمان اتصال

نحوه و زمان اتصال عناصر معماری مشخص شود. اگر از اتصال دیر هنگام برای جایگزینی بین اجزایی استفاده شود که خود می‌توانند منبع خطا در سیستم باشد، اطمینان حاصل شود که استراتژی در دسترس بودن تعیین شده برای پوشش خطاهای احتمالی تولید شده توسط همه‌ی منابع کافی است. به عنوان مثال، اگر از اتصال دیر هنگام در سیستم تاکسی آنلاین برای جابجایی بین قطعاتی مانند پردازنده‌هایی که ممکن است در معرض خطا قرار گیرند، استفاده شود، آیا مکانیزم‌های تشخیص و بازیابی خطا تعیین شده برای همه اتصالات‌های احتمالی به درستی عمل می‌کنند؟

۷-۴-۲ انتخاب فناوری

فناوری‌های موجود به گونه‌ای تعیین شود که بتوانند به شناسایی خطاها، بازیابی از آن‌ها و یا بازآفرینی مجدد بخش‌هایی از سیستم که با خطا متوقف شده‌اند، کمک کند.

همچنین ویژگیهای در دسترس بودن خود فناوری‌های انتخاب شده تعیین شود: پس از برخورد با چه خطاهایی

¹Recovery

²Log

³buffer

قادر به بازیابی مجدد خواهند بود؟ چه نقص هایی ممکن است توسط این فناوری ها در سیستم وارد شود؟

۵-۲ مطالعات موردی

در اینجا مطالعات موردی در ۳ حوزه حمل و نقل درون شهری برای نیاز دسترس پذیری انجام شده است.

۱-۵-۲ تاکسیرانی و حمل و نقل درون شهری

برنامه ی تاکسی رانی برخط نیازی است تا تقریباً همواره در دسترس کاربران خود باشد؛ اگر برنامه نیاز به بروزرسانی یا تعمیرات داشته باشد، باید سعی شود در طول به عملیات خدمات حیاتی برنامه حفظ شوند و حتماً پیش از آغاز بروزرسانی کاربران را از زمان آغاز و تخمین از طول زمان عملیات باخبر سازد.

سرویس هایی نظیر Uber با کمک کافکا از تکنیک هایی نظیر افزونگی داده و پردازش برای بالابردن ضریب در دسترس پذیری خود عمل می کنند [۴].

در این رابطه می توان سناریوهای زیر را نوشت:

سناریو: یک عامل خارجی پیام هایی که توسط سیستم مورد انتظار نیستند را به سیستم ارسال می کند.

- منبع محرک: عامل خارجی

- محرک: پیامی که سیستم انتظار آن را ندارد

- محصول: فرآیند

- محیط: محیط اجرای عادی

- پاسخ: ثبت اطلاعات پیام دریافتی و عدم پذیرش پیام های بیشتر از عامل خارجی مذکور

- اندازه گیری پاسخ: درصد دسترس پذیری

سناریو: در اوبر اگر یکی از دیتاسنتر ها fail شود تمامی امکانات آن توسط یک دیتاسنتر دیگر موقتاً ارائه می شود.

- منبع محرک: heartbeat monitor

- محرک: دیتاسنتر پاسخگو نیست.

- محصول: مدل هیبریدی ابری

- محیط: نرمال، زمان اجرا

• پاسخ: اوبر از هیچ دیتاسترک آپی برای یک location استفاده نمی‌کند و در هنگام شکست با استفاده از ابزار Terraform بار آن دیتا سنتر را به سایرین انتقال می‌دهد.

• اندازه گیری پاسخ: درصد در دسترس پذیری

۲-۵-۲ اشتراک خودرو و حمل و نقل درون شهری

در برنامه‌ی اشتراک گذاری خودرو در دسترس پذیری از اهمیت بالایی برخوردار است، زیرا کاربران در صورتی که برنامه در هنگام برنامه ریزی برای سفر و یا در هنگام سفر در دسترس نباشد با مشکلات جدی ای مواجه خواهند شد. در صورتی که برنامه نیاز به بروزرسانی یا تعمیر دارد، پیش از انجام عملیات باید از وجود نقطه‌ی بازگشت^۱ مناسب اطمینان حاصل شود تا در صورت نیاز وضعیت سیستم به پیش از زمان بروزرسانی بازگشت داده شود. سرویسی نظیر waze carpool علاوه بر نیز خط مشی‌های دیگری نظیر این که یک کاربر فقط می‌تواند دو سفر در روز داشته باشد را نیز اجرا می‌کند، این خط مشی علاوه بر جلوگیری از درآمذزایی می‌تواند به دسترس پذیری بیشتر این سرویس نیز کمک کند [۳۹].

در صورت نیاز به بروزرسانی یا تعمیر باید حداقل از یک روز قبل به کاربران در مورد زمان آغاز و طول بازه‌ی عملیات آگاهی داده شود و همچنین امکان استفاده از برخی امکانات برنامه نظیر برنامه ریزی و اشتراک سفر با استفاده از سرورهای پشتیبان حفظ شود.

در این رابطه می‌توان سناریوهای زیر را نوشت:

سناریو: کاربر نهایی زمانی که سرورها در حال به روزرسانی هستند، درخواست ارسال می‌کند.

• منبع محرک: کاربر نهایی

• محرک: سرور قادر به پردازش درخواست نیست

• محصول: سرور

• محیط: در دست تعمیر

• پاسخ: سرور وضعیت تعمیر را به کاربر اعلام کند.

• اندازه گیری پاسخ: زمان لازم برای انجام به روزرسانی

سناریو: وقتی که لود بر روی سرور پردازش درخواست‌های برنامه‌ریزی زیاد است، در حالت کاهشی سرور عمل کند تا سرور از دسترس خارج نشود.

¹ Restore Point

- منبع محرک: کاربر نهایی
- محرک: درخواست برنامه‌ریزی سفر
- محصول: سرور
- محیط: overloaded
- پاسخ: به اشخاص مناسب (افراد یا سیستم‌ها) اطلاع رسانی شود. و در حالت degraded عمل کند تا تمهیدات برای بازگشت به حالت نرمال انجام شود.
- اندازه گیری پاسخ: زمان تا بازگشت به حالت نرمال، درصد دسترس پذیری

۲-۵-۳ پارکینگ های عمومی و حمل و نقل درون شهری

اختلال در برنامه ی مدیریت پارکینگ زمانی که فرهنگ استفاده از برنامه در شهری جا افتاده است، می تواند عواقب بدی را به دنبال داشته باشد. بهتر است به روزرسانی سرور تا جایی که ممکن است منطقه به منطقه و در زمانی که استفاده از خیابان ها و پارکینگ ها به حداقل می رسد صورت پذیرد. برنامه‌هایی مثل Parkwhiz که به صورت گسترده در جهان استفاده می‌شوند نیاز به دسترسی پذیری تمام مدت دارند مخصوصا در شهرهای شلوغی مثل نیویورک که پارکینگ‌ها ۲۴ ساعته هستند [۱]. همچنین اگر پس از کامپایل مولفه ای به برنامه اضافه می شود، همانند اضافه شدن پرداخت با تلفن همراه، معماری باید از اینکه اضافه شدن مولفه ی جدید خللی در نیاز در دسترس پذیری ایجاد نمی کند، اطمینان حاصل کند. برای مثال، فرض کنید اضافه کردن یک درگاه پرداخت جدید سبب بروز خطا در سیستم شده و استفاده از آن را دچار مشکل سازد و در نتیجه سیستم از دسترس کاربران خارج شود.

در این رابطه می‌توان سناریوهای زیر را نوشت:

سناریو: در اوبر اگر یکی از دیتاسنتر ها fail شود تمامی امکانات آن توسط یک دیتاسنتر دیگر موقتا ارائه می‌شود.

- منبع محرک: heartbeat monitor
- محرک: سرویس پاسخگو نیست.
- محصول: سرور
- محیط: نرمال، زمان اجرا

- پاسخ: مطلع کردن اپراتور
- اندازه گیری پاسخ: درصد در دسترس پذیری
- سناریو: کاربر نهایی زمانی که سرور ها در حال به روزرسانی هستند، درخواست ارسال می کند.
- منبع محرک: کاربر نهایی
- محرک: سرور قادر به پردازش درخواست نیست
- محصول: سرور
- محیط: در دست تعمیر
- پاسخ: سرور وضعیت تعمیر را به کاربر اعلام کند.
- اندازه گیری پاسخ: زمان لازم برای انجام به روزرسانی

فصل سوم

قابلیت همکاری

۳-۱ تعریف قابلیت همکاری

قابلیت همکاری^۱ نیازمندی‌ای است که به میزان توانایی دو یا بیشتر سیستم، در انتقال معنادار اطلاعات از طریق رابط^۲ هایشان در یک زمینه‌ی خاص می‌پردازد. این تعریف علاوه بر توانایی مبادله‌ی صحیح اطلاعات که به قابلیت همکاری نحوی^۳ منجر می‌شود به بعد فهم صحیح اطلاعات مبادله شده توسط طرفین (قابلیت همکاری معنایی^۴) نیز تاکید دارد [۴۹]. زمانی که از قابلیت همکاری صحبت به میان می‌آید بررسی یک سیستم در انزوا معنا نمی‌دهد فلذا باید در مورد سیستم‌های همکار، نحوه‌ی همکاری و شرایط همکاری صحبت کرد. از جمله دلایلی که قابلیت همکاری در میان چندین سیستم را دارای اهمیت می‌سازد، دو مورد زیر مطرح هستند:

۱. سیستم طراحی شده قرار است خدماتی را در اختیار سایر سیستم‌های از پیش شناخته شده یا نشده قرار دهد. برای مثال Google Maps سیستمی است که خدمات زیادی را به سایر سیستم‌ها در حوزه حمل و نقل درون شهری از طریق رابط‌های خود ارائه می‌دهد که در هر ۳ سیستم تاکسیرانی آنلاین، اشتراک خودرو

¹ Interoperability

² Interface

³ Syntactic Interoperability

⁴ semantic Interoperability

و پارکینگ عمومی کاربرد دارد [۳۰].

۲. سیستم متشکل از زیر سیستم‌هایی است که هر یک وظایف مشخصی را بر عهده دارند و همچنین این زیر سیستم‌ها به یک دیگر وابسته بوده و خدماتی را به یکدیگر ارائه می‌دهند. برای مثال تک تک سیستم‌های حوزه حمل و نقل درون شهری که دارای که شامل چندین زیر سیستم است.

فرآیند همکاری دو مهم کشف^۱ و بکارگیری پاسخ‌ها^۲ را در بر دارد. مشتریان یک سیستم پیش از و یا در زمان همکاری با یک سیستم نیاز دارند تا از مکان^۳، اطلاعات هویتی^۴ و همچنین رابط‌های در دسترس سیستمی که قصد همکاری با آن را دارند اطلاع داشته باشند. در زمان ایجاد همکاری و دریافت پیام از سایر سیستم‌ها، سیستم می‌تواند یکی از سه رویکرد زیر را در پیش گیرد:

۱. حالت اول این است که سیستم پاسخ را به سیستمی که درخواست را ارسال نموده است بازگرداند. همانند بسیار از سیستم‌های Server-Client : که می‌توان به درخواست تاکسی آنلاین اشاره کرد که درخواست کاربر به سرور ارسال می‌شود.

۲. گاهی ممکن است پاسخ به درخواست کننده‌ی سرویس بازگشت داده نشود و در عوض پاسخ به سیستمی دیگر ارسال گردد.

۳. گاهی سیستم، پاسخ را برای تمامی سیستم‌هایی که ممکن است علاقه‌مند باشد ارسال می‌کند.

۲-۳ سناریوی عمومی قابلیت همکاری

سناریوی عمومی قابلیت همکاری به صورت زیر است:

- منبع محرک^۵ : سیستمی که شروع کننده‌ی همکاری است.
- محرک^۶ : درخواست تبادل اطلاعات میان سیستم‌ها
- محصول^۷ : سیستمی که قصد تبادل اطلاعات با آن وجود دارد: برای مثال هر سرویس ثالثی که سرویسی را در حوزه حمل و نقل ارائه می‌دهد، مانند سرویس پیدا کردن مسیر بین دو نقطه
- محیط^۸ : مجموعه سیستم‌هایی که قصد تبادل اطلاعاتی میان آن‌ها وجود دارد و در هنگام اجرا و یا پیش

¹Discovery

²Handling of the response

³Location

⁴Identity

⁵Source of stimulus

⁶Stimulus

⁷Artifact

⁸Environment

از آن شناخته می‌شوند.

• پاسخ^۱: درخواست برای همکاری باعث تبادل اطلاعات می‌شود. اطلاعات توسط طرف گیرنده از نظر نحوی و معنایی قابل درک است. همچنین، این درخواست رد می‌شود و نهادهای مربوطه از آن مطلع می‌شوند. در هر حال ممکن است درخواست‌های انجام شده توسط سیستم‌ها ثبت^۲ شود.

• اندازه‌گیری پاسخ^۳: درصدی از اطلاعات که به درستی مبادله شده‌اند و یا درصدی از اطلاعات مبادله شده که به درستی پذیرفته نشده‌اند، می‌توانند روش‌های اندازه‌گیری پاسخ باشند.

برای دست‌یابی به هر نیازمندی مجموعه‌ای از تکنیک‌ها را به کار می‌بندیم که در مورد قابلیت همکاری این تکنیک‌ها شامل مکان‌یابی^۴ و مدیریت ارتباط‌ها^۵ است.

در دسته‌بندی مکان‌یابی روش کشف سرویس^۶ در زمان اجرا اقدام به کشف سیستمی که قصد تبادل پیام با آن را داریم می‌کند. یک سرویس می‌تواند با استفاده از نوع سرویس، نام، مکان و سایر خصیصه‌ها مکان‌یابی شود.

در مقابل در دسته‌بندی مدیریت ارتباط‌ها دو روش معمول هماهنگ‌سازی^۷ و درخورسازی رابط^۸ از جمله روش‌های معمول به شمار می‌روند. روش هماهنگ‌سازی با استفاده از مکانیزم کنترلی می‌تواند امکان هماهنگی و مدیریت توالی فراخوانی سرویس‌های خاص را فراهم سازد و در کنار آن روش درخورسازی ارتباط به افزودن یا کاهش امکانات به یک رابط توجه دارد.

در زمینه‌ی قابلیت همکاری به دلیل وجود چالش‌های مشترک در معماری‌های متفاوت استانداردهای از پیش تعریف شده‌ی زیادی وجود دارند اما به دلیل سیر تکاملی که استاندارد‌ها معمولاً طی می‌کنند، نمی‌توان معماری سیستم را بر پایه‌ی استاندارد‌ها طراحی کرد. پیشنهاد می‌شود که همواره ابتدا با توجه به سایر فاکتورهای تاثیرگذار معماری انتخاب شود و سپس از میان استاندارد‌های منطبق با معماری، به انتخاب استاندارد‌های مناسب با معماری پرداخته شود.

۳-۳ طراحی فهرست بازبینی برای قابلیت همکاری

در ادامه فهرستی را برای پشتیبانی از روند طراحی و تجزیه و تحلیل برای قابلیت همکاری ارائه می‌شود.

¹Response

²logged

³Response Measure

⁴Locate

⁵Manage Interface

⁶Discover System

⁷Orchestrate

⁸Tailor interface

۱-۳-۳ تخصیص مسئولیت ها

در زمان تخصیص مسئولیت ها باید ^۱ به موارد زیر توجه شود:

- معمار سیستم باید مشخص کند که انجام کدام یک از وظائف سیستم نیازمند همکاری با سایر سیستم هاست.
- اطمینان حاصل شود که مسئولیت هایی به جهت تشخیص درخواست های همکاری با سیستم های خارجی شناخته شده و یا ناشناخته اختصاص یافته است.
- معماری باید مسئولیت پاسخ به وظائف زیر را در سیستم لحاظ کرده باشد:

- پذیرش درخواست همکاری

- تبادل اطلاعات

- عدم پذیرش درخواست همکاری

- اعلان و آگاهی سازی هویت های مرتبط با سیستم از همکاری با سایر سیستم ها

- ثبت درخواست

۲-۳-۳ مدل هماهنگی

مدل هماهنگی ^۲ باید با در نظر گرفتن دغدغه و نیاز های معماری به آن ها پاسخ دهد. مواردی که برای پاسخ دهی به نیاز کارایی ^۳ باید توسط مدل هماهنگی مورد توجه قرار گیرند شامل موارد زیر است:

- حجم ترافیکی که به صورت مستقیم توسط سیستم های تحت کنترل و حتی سیستم هایی که کنترلی بر روی ترافیک آن ها ندارد ایجاد می شود.
- ارسال به موقع پیام ها از سمت سیستم
- هم زمانی ^۴ ارسال پیام از سمت سیستم
- و شاید مهم تر از همه این است که اطمینان حاصل شود سیستم های تحت کنترل فرضیات قابل انطباقی در همکاری با سایر سیستم هایی که کنترلی بر روی آن ها وجود ندارد، در نظر دارد.

¹ Allocation of Responsibilities

² Coordination Model

³ Performance

⁴ Currency

۳-۳-۳ مدل داده

مدل داده^۱ از مهم ترین بخش هایی است که اگر به درستی بر آن نظارت صورت نپذیرد می تواند عواقب سنگینی را به همراه داشته باشد. در مدل داده ای باید از برداشت یکسان طرفین همکاری از اطلاعات اطمینان حاصل کنید. به این منظور لازم است تا انتزاعات اصلی داده های مبادله شده از نظر نحو و معناشناسی مورد بررسی دقیق قرار بگیرند.

۳-۳-۴ نگاشت در میان عناصر معماری

جدا از ملاحظات^۲ که در نگاشت در میان عناصر معماری^۲ درباره امنیت، دسترس پذیری و کارایی وجود دارد و در سایر فصل های این مستند به آن ها پرداخته خواهد شد؛ در رابطه با قابلیت همکاری مساله ی مهم نگاشت درست از اعضای سازنده ی پردازنده ها است.

۳-۳-۵ مدیریت منابع

لازم است معمار نرم افزار اطمینان حاصل کند منابعی که سیستم به جهت همکاری با سایر سیستم ها نیاز دارد هیچ گاه سیستم را تحت فشار غیر قابل تحمل قرار نخواهد داد و بار منابع تحمیل شده توسط الزامات همکاری همواره قابل قبول است.

در ضمن نیاز یک سیستم ناظر با هدف تخصیص منابع به صورت منصفانه و بر پایه ی سیاست های تبیین شده در معماری باید حتما دیده شود.

۳-۳-۶ زمان اتصال

لازم است سیستم هایی که ممکن است با یکدیگر همکاری داشته باشند شناسایی شده و از رعایت موارد زیر در مورد آن ها اطمینان حاصل شود.

- از وجود یک سیاست مشخص به منظور همکاری با سیستم های شناخته شده و ناشناخته خارجی اطمینان حاصل کنید.
- از وجود قوانینی برای عدم پذیرش درخواست ها و ثبت درخواست هایی که پذیرفته نشده اند اطمینان حاصل کنید.
- در صورت تاخیر در اتصال، اطمینان حاصل شود که مکانیزم هایی از کشف خدمات یا پروتکل های جدید مرتبط یا ارسال اطلاعات با استفاده از پروتکل های انتخاب شده پشتیبانی می کنند.

¹Data Model

²Mapping among Architectural Elements

۲-۳-۳ انتخاب فناوری

در انتخاب فناوری باید روی این مساله تمرکز کرد که انتخاب هر فناوری چه تاثیری بر روی رابطها و همکاری سیستم با دیگر سیستمها خواهد داشت. بررسی شود آیا فناوری انتخاب شده توانایی پاسخ گویی به نیاز هایی که توسط قابلیت همکاری مطرح می شوند را دارا هستند و یا خیر؟

۴-۳ مطالعات موردی

در اینجا مطالعات موردی در ۳ حوزه حمل و نقل درون شهری برای نیاز قابلیت همکاری انجام شده است.

۱-۴-۳ تاکسیرانی و حمل و نقل درون شهری

انجام پرداختها زمانی که کاربر یک سرویس تاکسی را استفاده می کند معمولاً با استفاده از یک نهاد سوم انجام می شود، چرا که امروزه بیشتر تبادلات دیگر به صورت وجه کاغذی نقد انجام نمی شود و با پیشرفت تکنولوژی نیز گوشی ها توانسته اند به ابزار خوبی برای این کار تبدیل شوند. در تمامی اپ های مورد بررسی در حوزه تاکسیرانی همگی از یک نهاد ثالث برای پرداخت استفاده می کنند [۱۳] و بنابراین نیاز است که درک مشترکی از نظر نحوی و معنایی در مورد اطلاعات مبادله شده وجود داشته باشد. همچنین از آنجا که نیاز است کاربر موقعیت جغرافیایی خود را به سیستم اعلام کند، در این اپ ها معمولاً از یک ادغام سازی با آنچه که گوگل سرویس می دهد نیز انجام می شود تا تاکسی بتواند درست به کاربر برسد و مسیریابی مناسبی تا مقصد او نیز انجام دهد. جزئیات این موضوع کاملاً مشخص نیست و هر کدام از آن ها در یک سطح خاصی از این سرویس استفاده می کنند. در مستندات پروژه MyWay اعلام شده است که معماری اجازه دارد از یک سرویس ثالث جستجو برای انتخاب مسیر بین دو نقطه استفاده کند. مازول رابط به نحوی انعطاف پذیر است که می توان داده جدید را به ازای هر دسته از داده ها اضافه کرد. هر سیستم ثالثی اجازه دارد فرانت اند^۱ خود را با استفاده از بک اند^۲ برنامه پیاده سازی کند. مدل داده به راحتی قابل توسعه دادن است تا اجازه دهد که پیشرفت های آتی در پروژه قابل اجرا کردن باشد. برنامه از تاریخچه داده های واسط برای مازول ارزیابی استفاده می کند. فرانت های برنامه شامل مولفه های برنامه ی iOS، اندروید و وب است؛ که این مولفه به واسطه رابط های تعیین شده از سمت بک اند اقتدام به مبادله ی داده های خود می کنند. برنامه برای تبلت های با سایز مختلف نیز سازگار باشد. همچنین برنامه وب نیز باید وجود داشته باشد که توسط مرورگرهای فایرفاکس^۳ و کروم^۴ و سافاری^۵ به خوبی

^۱Front-End

^۲Back-End

^۳Firefox

^۴Chrome

^۵Safari

پشتیبانی شود. نوتیفیکشن‌ها نیز به صورت پوش^۱ ارسال شود. قابلیت استفاده و ادغام دادگان نقشه گوگل و نقشه‌های محلی با فرمت‌های به خصوصی نیز توسط اپلیکیشن فراهم باشد [۲۲].

۳-۴-۲ اشتراک خودرو و حمل‌ونقل درون شهری

در برنامه‌های اشتراک خودرو، اکثر معماری‌ها از جنس میکروسرویس‌ها^۲ توسعه داده شده‌اند؛ پس ارتباط میان این میکروسرویس‌ها در عملکرد صحیح برنامه نقش حیاتی‌ای را بازی می‌کند. معماری در این برنامه‌ها باید از درخورسازی مناسب رابط‌های میان میکروسرویس‌های متفاوت اطمینان حاصل کند. معماری باید مطمئن شود میکروسرویس‌ها درک متقابل صحیحی از معنای پیام‌های مبادله شده دارند و همچنین از نظر نحوه‌ی بیان برطبق توافق مشترک از پیش تعیین‌شده‌ای اقدام به مبادله‌ی اطلاعات می‌کنند. همچنین سیستم‌های اشتراک خودرو نظیر بسیاری دیگر از برنامه‌های فعال در حوزه‌ی حمل‌ونقل از سرویس‌دهنده‌های بزرگ GIS اطلاعات نقشه‌های خود را دریافت می‌کنند و اطمینان از وجود قابلیت همکاری با یکی از این GIS‌ها شاید مرکز عملکرد صحیح این دسته از برنامه‌ها به حساب آید. شرکت waze با همکاری شرکت ArcGIS که یک شرکت سرویس‌دهنده موقعیت جغرافیایی است به این هدف جامعه عمل می‌پوشاند [۱۲]. از آنجا در یک برنامه‌ی پیشنهاد سفر درخواست یک هم سفر زیر سیستم‌های مختلفی از سیستم را درگیر می‌کند و آن عمل پیچیده به همکاری و فراخوانی به ترتیب خاصی از زیر سیستم‌هایی نظیر مکان‌یابی و پیشنهاددهنده‌های متفاوت نیاز دارد، که حتماً به تنظیمات دقیق از پیش تعیین‌شده نیاز خواهد داشت.

۳-۴-۳ پارکینگ‌های عمومی و حمل‌ونقل درون شهری

در مورد مطالعاتی پارکینگ‌های عمومی، وجود زیر سیستم‌های سخت افزاری نظیر سنسورهای تشخیص وجود خودرو در پارکینگ‌ها از اهمیت بالایی نیاز قابلیت همکاری در چنین برنامه‌هایی خبر می‌دهد. در دنیای سخت‌افزار و تکنولوژی‌های ارتباطی به دلیل رشد سریع دانش، تغییرات به سرعت رخ می‌دهند. برای مثال، اگر برنامه برای تشخیص فضاها یا خالی در پارکینگ از سنسور خاصی استفاده می‌کند، با ورود نوع پیشرفته‌تر سنسورها با عملکرد بهینه‌تر به بازار این دسته از برنامه‌ها باید از قابلیت همکاری سایر زیرسیستم‌ها با زیرسیستم مرتبط به سنسورهای جدید اطمینان حاصل کنند. از آنجا که بیشتر این سنسورها از پروتکل‌های معروف برای ارتباط استفاده می‌کنند در نتیجه شرکت‌های زیادی در این رابطه رقابت می‌کنند و هر پارکینگی می‌تواند از سنسورهای شرکت‌های متفاوتی استفاده کند این سنسورها بدلیل استفاده از تکنولوژی آلتراسونیک نسبت ارزان قیمت هستند و در نتیجه می‌تواند در ابعاد بالا از آن‌ها استفاده شود [۳۲]. در برنامه‌ی پارکینگ آنلاین ممکن است هر

¹Push

²Micro-Service

پارکینگ از تجهیزات متفاوتی استفاده کند و در نتیجه سرویس های متفاوتی ارائه شود؛ کاربران نهایی از یک برنامه ی یکتا برای جستجوی سرویس استفاده می کنند و فلذا این برنامه باید قابلیت جستجوی سرویس های فعال در یک پارکینگ را داشته باشد و با رابط های مناسب با آن ارتباط برقرار کند.

در رابطه با تاکسی رانی آنلاین و اشتراک گذاری خودرو و پارکینگ آنلاین برای دو قسمت پرداخت و اطلاعات جغرافیایی می توان سناریوهای زیر را نوشت:

سناریو: سیستم درخواست تاکسی یا اشتراک گذاری خودرو را دریافت کرده و با اطلاعات مکانی فعلی و مقصد ما راننده مورد نظر را معرفی و به سمت ما هدایت می کند.

- منبع محرک: سرور
- محرک: درخواست برای تبادل اطلاعات با گوگل مپ
- محصول: سیستمی که می خواهد تبادل اطلاعات کند.
- محیط: زمان اجرا، پیشتر سرویس ها با یکدیگر آشنا شده اند.
- پاسخ: قبول یا رد درخواست و حفظ تاریخچه درخواست توسط جزهای مشارکت کننده در این تبادل (اطلاعات جغرافیایی)
- اندازه گیری پاسخ: درصد اطلاعاتی که به درستی پردازش یا رد شده است.

سناریو: کاربر در هنگام پرداخت به درگاه پرداخت هدایت می شود و پس از انجام عملیات پرداخت یا تمام شدن مهلت کاربر را به اپ باز می گرداند. در این حین سیستم تاکسی رانی یا اشتراک خودرو با درگاه پرداخت در حال تبادل اطلاعات این که پرداخت صورت گرفته یا نه هستند.

- منبع محرک: سرور
- محرک: درخواست برای تبادل اطلاعات با درگاه پرداخت
- محصول: سیستمی که می خواهد تبادل اطلاعات کند.
- محیط: زمان اجرا، پیشتر سرویس ها با یکدیگر آشنا شده اند.
- پاسخ: قبول یا رد درخواست و حفظ تاریخچه درخواست توسط جزهای مشارکت کننده در این تبادل (گزارش پرداخت)
- اندازه گیری پاسخ: درصد اطلاعاتی که به درستی پردازش یا رد شده است.

فصل چهارم

اصلاح پذیری

۴-۱ تعریف اصلاح پذیری

اصلاح پذیری^۱ یا همان قابلیت اصلاح یکی از موارد مهمی است که تقریباً در تمامی مدل‌های کیفیت نرم افزار از جمله نرم افزارهای حوزه حمل و نقل درون شهری مورد سنجش قرار می‌گیرد. در این نیازمندی غیرعملکردی به اهمیت اصلاح پذیری نرم افزار با هزینه کم پرداخته می‌شود. برای اصلاح پذیری تعاریف زیادی می‌توان ارائه نمود که از جمله آن‌ها می‌توان به موارد زیر اشاره کرد [۴۵].

۱. چقدر نرم افزار قابل تغییر است؟

۲. هزینه تغییر

۳. میزان سهولت انجام تغییر

مطالعات زیادی نشان داده است که با پرداختن به ویژگی اصلاح پذیری می‌توان در حین توسعه به طور چشم‌گیر هزینه یک سیستم نرم افزاری را کاهش داد. همچنین به موجب پرداختن به این ویژگی به معمار نرم افزار این امکان داده می‌شود که هزینه‌های نگهداری را پیش‌بینی و ریسک و انعطاف پذیری نرم افزار را ارزیابی کند.

^۱Modifiability

۲-۴ سناریوی عمومی اصلاح پذیری

سناریوی عمومی قابلیت اصلاح پذیری به صورت زیر است:

- منبع محرک^۱ : هر یک از کاربران، توسعه دهندگان، ادمین‌ها؛ برای مثال در حمل و نقل درون شهری یا اشتراک گذاری خودرو کاربران نهایی درخواست دهنده سرویس تاکسی یا خودرو، ادمین‌ها، مدیران و توسعه دهندگان تمامی نرم افزارهای این حوزه از جمله مثال‌های منبع محرک هستند.

- محرک^۲ : اضافه، حذف یا مورد تغییر قرار دادن یک عملکرد، یا تغییر دادن یک ویژگی کیفیت یا ظرفیت یا تکنولوژی: برای مثال کاربران و ادمین‌های نرم افزارهای حوزه حمل و نقل درون شهری بسته به دسترسی‌هایی که به نرم افزار دارند می‌توانند داده خود یا بقیه را مورد تغییر قرار داده و توسعه دهندگان آن‌ها نیز می‌توانند کد را تغییر دهند.

- محصول^۳: کد، داده، مولفه‌ها و ...

- محیط^۴: در زمان کامپایل، زمان اجرا، زمان ساخت، زمان طراحی

- پاسخ^۵ : در پاسخ مناسب یک یا چند عملیات رخ می‌دهد. در این عملیات‌ها انجام و تست و استقرار تغییر انجام می‌شود.

- اندازه گیری پاسخ^۶: برای اندازه‌گیری پاسخ از معیارهای هزینه و تعداد و ساینز و پیچیدگی موارد مورد بررسی، تلاش و .. استفاده می‌شود.

تاکتیک‌های زیادی را می‌توان برای نرم افزارهای حوزه حمل و نقل درون شهری در نظر گرفت که بهبود این نیاز کمک کند که هدف همه آن‌ها کنترل پیچیدگی تغییرات و انجام تغییرات، تست و استقرار تغییرات در زمان و هزینه تعیین شده است.

یکی از این تاکتیک‌ها کم کردن حجم ماژول‌های برنامه است. اگر بتوان یک ماژول که را به ماژول‌های کوچکتر تبدیل کرد، آنگاه در این صورت هزینه تغییرات در آینده نیز به صورت میانگین کاهش پیدا می‌کند. چرا که هرچه حجم ماژول بیشتر باشد هزینه تغییر آن زیادتر است.

¹ Source of stimulus

² Stimulus

³ Artifact

⁴ Enviroment

⁵ Response

⁶ Response Measure

تاکتیک دیگر افزایش مقدار چسبندگی^۱ است. به این صورت که اگر دو وظیفه داخل یک ماژول یک هدف را دنبال نمی‌کنند آنگاه باید آن دو در ماژول‌های متفاوتی قرار بگیرند. این کار ممکن است موجب ایجاد ماژول‌های جدید یا جابه‌جایی وظایف به ماژول‌های موجود شود.

همچنین بر اساس اصل اتصال آزادانه^۲ تلاش می‌شود که تا حد ممکن وابستگی‌ها ماژول‌ها کاهش یابد و آن‌ها را محدود کرد [۲۱]. عملیات باز تولید^۳ را نیز بر اساس همین اصل می‌توان زمانی که دو ماژول تحت تاثیر یک تغییر، تاثیر می‌پذیرند انجام داد. چرا که در واقع قسمتی از آن دو کپی یکدیگر هستند. و در آخر باید گفت که هر چه عملیات اتصال را بتوان به تعویق بیندازیم تا انعطاف‌پذیری تغییر هر جز حفظ شود و هزینه تغییر خاص هر کدام کاهش یابد بهتر است.

به منظور بهره‌بری از این تاکتیک‌ها می‌توان با توجه به هریک از پروژه‌های تاکسی آنالین، اشتراک‌گذاری خودرو و مدیریت پارکینگ یک فهرست بازبینی تنظیم کرد. این موضوع به بهبود نیاز اصلاح‌پذیری پاسخ می‌دهد.

۳-۴ طراحی فهرست بازبینی برای اصلاح‌پذیری

۱-۳-۴ تخصیص مسئولیت‌ها

در حوزه نرم‌افزارهای حمل و نقل درون شهری ممکن است که تغییراتی در هر یک بخش‌های فنی، قانونی، اجتماعی، تجاری نرم‌افزار مربوطه رخ دهد. برای مثال نیاز به یک عملکرد جدید در بخش فنی تاکسی آنالین یا تغییر قانون و سیاست‌های این حوزه برای مدیریت پارکینگ‌ها یا بخش اجتماعی و بازخورد جامعه به نحوه اشتراک‌گذاری خودرو، در هر یک ممکن است که باعث شود، بخشی تغییر کند. باید دید که چه تغییراتی احتمال دارد رخ دهد و برای هر تغییر بالقوه مسئولیت‌هایی تعیین شود. همچنین مسئولیت‌هایی که تغییر می‌کنند باید در یک ماژول قرار بگیرند و مسئولیت‌هایی که در زمان‌های مختلف تغییر می‌کنند در ماژول‌های جداگانه قرار بگیرند تا به بهبود این نیاز کمک کنند.

۲-۳-۴ مدل هماهنگی

در این قسمت تعیین شود که کدام ویژگی یا ویژگی کیفیت می‌تواند در زمان اجرا تغییر کند و این موضوع چگونه بر هماهنگی تاثیر می‌گذارد. در حوزه نرم‌افزارهای حمل و نقل درون شهری ممکن است که هر یک از دستگاه‌ها، پروتکل‌ها و مسیرهای ارتباطی مورد استفاده برای هماهنگی تغییر پیدا کنند. برای مثال ممکن است پروتکل ارتباطی سنسورهای پارکینگ در زمان اجرا تغییر یابد و یا اطلاعات منتقل شده از کاربران در برنامه‌های

¹ cohesion

² loose coupling

³ refactor

درخواست تاکسی آنلاین یا اشتراک خودرو در زمان اجرا تغییر پیدا کند. به این منظور برای آن دستگاه ها، پروتکل ها و مسیرهای ارتباطی که محتمل است تغییر پیدا کنند اطمینان حاصل شود که تأثیر تغییرات فقط به مجموعه کوچکی از مازول ها محدود می شود.

۴-۳-۳ مدل داده

در این قسمت تعیین می شود که چه تغییراتی (یا دسته بندی تغییرات) در انتزاع داده ها، عملکردهای آنها یا خصوصیات آنها محتمل است که رخ دهد. در حوزه نرم افزارهای حمل و نقل درون شهری ممکن است که تغییر یا دسته تغییراتی در انتزاعات داده ها شامل ایجاد، شروع اولیه، ماندگاری، دستکاری، ترجمه یا تخریب هر یک رخ بدهد. به این منظور برای هر تغییر یا دسته تغییر، باید تعیین شود که تغییرات توسط چه منبع محرکی انجام می شود. همانگونه که در بخش قبل گفته شد کاربر نهایی، مدیر سیستم یا توسعه دهنده می تواند منبع تغییر باشد. در نهایت باید اطمینان حاصل کرد که ویژگی های مورد نیاز کاربر برای انجام تغییر قابل مشاهده است و کاربر دارای مجوزهای صحیح برای اصلاح داده ها، عملکردهای آن یا خصوصیات آن است. در نهایت تخصیص اطمینان حاصل شود تعداد تغییرات بالقوه و شدت اصلاحات در انتزاعات به حداقل ممکن برسد.

۴-۳-۴ نگاشت در میان عناصر معماری

در این قسمت تعیین می شود که آیا تغییر نحوه نگاشت قابلیت عناصر محاسباتی (به عنوان مثال فرآیندها، رشته ها، پردازنده ها) در زمان اجرا، زمان کامپایل، زمان طراحی یا زمان ساخت مطلوب است؟ در حوزه حمل و نقل درون شهری نیاز است که تغییرات لازم برای افزودن، حذف یا اصلاح یک تابع یا یک ویژگی کیفیت تعیین شود که این این تعیین به موارد زیر ممکن است وابسته باشد:

- اجرای وابستگی ها

- اختصاص داده به پایگاه داده

- اختصاص عناصر زمان اجرا به پردازش ها، رشته ها یا پردازنده ها

همچنین اطمینان حاصل شود که چنین تغییراتی با مکانیزمی که اتصال را به تاخیر می اندازد انجام شود.

۴-۳-۵ مدیریت منابع

در این قسمت تعیین می شود که چگونه اضافه کردن، حذف یا تغییر یک مسئولیت یا ویژگی کیفیت بر استفاده از منابع تأثیر می گذارد. به عنوان مثال در تاکسی آنلاین بر اساس رشد روزافزون کاربران، ممکن است سیستم نیاز به معرفی منابع جدید و حذف و جایگزینی منابع قبلی داشته باشد. یا ایجاد محدودیت ها روی منابع اشتراک گذاری

خودرو که باید اطمینان حاصل کرد که برای تامین نیازهای سیستم کافی هستند. یا ایجاد یک سری واسط برای منابع در مدیریت سیستم‌های پارکینگ تا دسترسی تنها از طریق آن‌ها به صورت کپسوله^۱ فراهم شود.

۶-۳-۴ زمان اتصال

در تمامی نرم‌افزارهای حوزه حمل و نقل درون شهری نیاز است که برای هر تغییر یا دسته تغییرات:

- آخرین زمانی که باید تغییر ایجاد شود تعیین شود.
- مکانیزم مناسب برای اتصال دیر هنگام انتخاب شود.
- هزینه معرفی مکانیزم و ایجاد تغییرات با استفاده از مکانیزم انتخاب شده تعیین شود.
- همچنین تعداد زیادی انتخاب اتصال معرفی نشود که خود این موضوع مانع تغییر شود، چرا که وابستگی ها در بین گزینه‌ها می تواند پیچیده و ناشناخته باشد.

۷-۳-۴ انتخاب فناوری

نیاز است که فناوری مناسب برای نرم‌افزارهای حوزه حمل و نقل درون شهری به گونه‌ای در نظر گرفته شود که انجام تست و استقرار پس از انجام اصلاحات را راحت‌تر کند و همچنین تعیین شود که تغییر خود فناوری چقدر راحت است؟ فناوری‌ها را باید به گونه‌ای انتخاب شوند که بیشترین پشتیبانی از تغییرات را داشته باشند.

۴-۴ مطالعات موردی

در اینجا مطالعات موردی در ۳ حوزه حمل و نقل درون شهری برای نیاز اصلاح‌پذیری انجام شده است. همانگونه که قبلاً بیان شد باید موارد محتملی که ممکن است تغییر پیدا کند شناسایی شوند و مسئولیت‌های مناسب همراه با زمان بندی برای آن‌ها تعیین گردد. در این راه چند تاکتیک مانند کوچک کردن مازول‌ها یا ترکیب مازول‌های مرتبط مطرح شد. از نرم‌افزارهای سنجش کد مانند سونارکیوب^۲ نیز می‌توان برای کنترل کردن حجم مازول‌ها و وابستگی آن‌ها نیز استفاده کرد. [۲۹]

۱-۴-۴ تاکسیرانی و حمل‌ونقل درون شهری

اصلاح‌پذیری را می‌توان یکی از زیر مشخصه‌های قابلیت نگهداری دانست. در مستندات پروژه MyWay عنوان شده است که معماری نرم‌افزار باید منعطف باشد، تا بتوان هر نمونه آن را در هر شهر به صورت محلی اجرا کرد و قابلیت اشتراک داده نیز بین آن‌ها فراهم باشد. همچنین در توسعه این نرم‌افزار برای بازتولید کد نسبت

¹Capsulated

²Sonarqube

به مواردی که توسط نرم افزارهای سنجش کد گزارش کرده اند، مواردی نیز در نظر گرفته شده که این بازتولید کد چنین مواردی در زمان توسعه موجب بهبود اصلاح پذیری می شود. همچنین در چنین سیستم هایی می توان به تنظیمات سطح کاربر اشاره کرد که انجام تغییرات و بازگرداندن اطلاعات به حالت اولیه به خوبی امکان انجام آن باید فراهم شود [۲۲].

۴-۴-۲ اشتراک خودرو و حمل و نقل درون شهری

در اپلیکیشن هایی که در اکوسیستم های تازه شکل گرفته فعالیت می کنند، یکی از مهم ترین دغدغه ها، عدم فرهنگ سازی در رابطه با چگونگی استفاده از اپلیکیشن است. عدم وجود فرهنگ تثبیت شده در جامعه سبب می شود این اپلیکیشن ها نیاز داشته باشند تا بسیار چابک عمل کرده و همراه با تغییرات به سرعت خود را اصلاح کنند [۲]؛ به عنوان مثال وقوع یک مشکل ممکن است قانون گذاران را مجاب کند تا قوانین بر نحوه ی ارائه ی خدمات توسط اپلیکیشن های این حوزه اعمال کنند و در چنین شرایطی معماری باید به خوبی از قابلیت اصلاح پذیری پشتیبانی کند.

۴-۴-۳ پارکینگ های عمومی و حمل و نقل درون شهری

در حوزه ی پارکینگ های عمومی ممکن است با پیشرفت سخت افزار و زیرساخت پارکینگ های شهری نیاز باشد تا رابط های سیستم به سرعت خود را با تغییرات جدید هماهنگ کنند فلذا اصلاح نقش پررنگی در چنین سیستم هایی نیز ایفا می کند؛ به عنوان یک مثال دقیق تر فرض کنید که پارکینگ ها در یک شهر شروع به نصب دستگاه هایی برای پرداخت هزینه ها از طریق تلفن های ساخت شرکت اپل کرده اند، معماری سیستم با برخورداری از اصلاح پذیری بالا می تواند به سرعت امکان استفاده از این نوع پرداخت را برای کاربران نهایی فراهم سازد. از روی موارد پشتیبانی اپلیکیشنی مثل ParkWhiz می توان چنین تغییراتی را در فرمت فایل ها، پروتکل ها و غیره دید [۳۱].

در رابطه با هر ۳ سیستم ها می توان سناریوهای زیر را نوشت:

سناریو: کاربر باید بتواند اطلاعات نمایه خود را مورد تغییر قرار بدهد و تغییر انجام شود.

• منبع محرک: کاربر نهایی

• محرک: اضافه، حذف و تغییر اطلاعات

• محصول: زمان اجرا

• محیط: داده

- پاسخ: عملیات مربوطه کاربر (اضافه، تغییر یا حذف) انجام شود.
- اندازه گیری پاسخ: میزان تلاش و مقدار زمان برای انجام تغییر
- سناریو: توسعه دهنده باید بتواند کد را تغییر دهد و هرچه هزینه این کار کمتر باشد بهتر است.
- منبع محرک: توسعه دهنده
- محرک: اضافه کردن ویژگی به کد
- محصول: کد
- محیط: زمان توسعه و استقرار
- پاسخ: تغییر انجام شود، تست و استقرار نیز انجام شود.
- اندازه گیری پاسخ: میزان هزینه و ریسک انجام تغییر
- که از این سناریو به طور خاص برای پارکینگ آنلاین می توان این سناریو را در نظر گرفت:
- سناریو: پارکینگ های سطح شهر به سیستم پرداخت با apple pay مجهز می شوند.
- منبع محرک: گسترش دهندگان سیستم
- محرک: افزودن قابلیت پرداخت با apple pay
- محصول: کد
- محیط: زمان ساخت
- پاسخ: افزودن قابلیت پرداخت با apple pay به کد و تست و کارگزاری^۱ در سیستم
- اندازه گیری پاسخ: پیچیدگی و مدت زمان مورد نیاز برای افزودن قابلیت پرداخت با apple pay توسط گسترش دهندگان سیستم

^۱Deploy

فصل پنجم

کارایی

۵-۱ تعریف کارایی

کارایی^۱ به توانایی سیستم در پاسخ گویی به نیازها در زمانی مشخص اشاره دارد. زمانی که رخدادی جدید نظیر خواسته ای از جانب کاربر یا سایر سیستم ها پیش می آید، سیستم باید در بازه ی زمانی مورد انتظار آن رخداد را بررسی کند و به نیاز آن پاسخ دهد. در عموم سیستم های حوزه حمل و نقل که به صورت برخط و مبتنی بر وب هستند ممکن است کارایی با گزاره هایی نظیر تعداد درخواست بر دقیقه بیان شود.

کارایی در بیشتر اوقات در کنار گسترش پذیری^۲ شنیده می شود، گسترش پذیری به افزایش ظرفیت پاسخ دهی با حفظ کارایی اشاره دارد و نباید مفهوم آن با کارایی یکسان در نظر گرفته شود.

۵-۲ سناریو عمومی کارایی

سناریو کارایی با فرارسیدن رویدادها به سیستم آغاز می شود. سیستم باید هم زمان با وجود سایر رویدادها، با تخصیص منابع به درخواست رویدادی که اکنون به سیستم رسیده نیز پاسخ دهد. رویدادها می توانند با الگوی مشخص و تبعیت از یک توزیع ریاضی رخ دهند و یا آن که غیر قابل پیش بینی باشند؛ الگوی فرارسیدن رویداد

¹Performance

²Scalability

ها می تواند به صورت دوره ای^۱، تصادفی^۲ و یا پراکنده^۳ باشد.

در سناریو عمومی کارایی، معیار سنجش پاسخ سیستم به محرک می تواند یکی از موارد زیر باشد:

- تاخیر^۴: زمان میان فرارسیدن رویداد و پاسخ سیستم را تاخیر سیستم می گویند. برای مثال زمانی که یک کاربر باید منتظر پاسخ جستجوی سفرهای مشابه در برنامه ی اشتراک خودرو بماند.

- مهلت های پردازشی^۵: یک رویداد ممکن زمان مشخصی را برای سیستم به عنوان حداکثر زمانی که باید پاسخ رویداد آماده باشد مشخص کند، تخطی از این زمان سبب کاهش کارایی سیستم خواهد شد.

- توان عملیاتی سیستم^۶: تعداد پردازش هایی سیستم در واحد زمان می تواند به پایان برساند به عنوان توان عملیاتی شناخته می شود. برای مثال، سرور یک برنامه تاکسی برخط توان پاسخ دادن به تعداد مشخصی درخواست سفر از سمت کاربران در لحظه را دارد.

همچنین معیار های دیگری نظیر واریانس تاخیر^۷ و تعداد رویداد های بلوکه شده نیز وجود دارند که کارایی سیستم به وسیله ی آن ها قابل سنجش است.

در ادامه به معرفی بخش های مختلف سناریو عمومی کارایی پرداخته می شود.

- منبع محرک^۸: یک منبع از داخل و یا خارج از سیستم که بتواند رویدادی را ایجاد و به سیستم ارسال کند، می تواند منبع محرک سناریوی کارایی باشد. برای مثال، کاربران نهایی و یا سایر زیرسیستم ها که سیستم های مورد مطالعه با آن ها همکاری می کنند.

- محرک^۹: فرا رسیدن یک رویداد برای مثال درخواست یک سفر به سیستم تاکسی رانی برخط.

- محصول^{۱۰}: سیستم یا بخشی از آن که مسئول رسیدگی به رویداد است.

- محیط^{۱۱}: سیستم می تواند در حالت های مختلف عملیاتی مانند حالت عادی^{۱۲}، اضطراری^{۱۳}، اوج بار^{۱۴} یا اضافه بار^{۱۵} باشد.

¹periodic

²Stochastic

³Sporadic

⁴Latency

⁵Deadlines in processing

⁶The throughput of the system

⁷Jitter

⁸Source of stimulus

⁹Stimulus

¹⁰Artifact

¹¹Environment

¹²Normal

¹³Emergency

¹⁴Peak Load

¹⁵OverLoad

- پاسخ^۱ : سیستم در پاسخ رسیدن یک رویداد با تخصیص منابع به آن پاسخ می دهد که ممکن است به تغییر محیط نیز منجر شود.
- اندازه گیری پاسخ^۲: معیار های سنجش کارایی سیستم پیش از این بررسی شدند. برای بررسی کارایی سیستم می توان به تاخیر، توان عملیاتی سیستم و مهلت های پردازشی اشاره کرد.

۳-۵ تاکتیک ها در کارایی

- هدف تاکتیک ها در کارایی تولید پاسخ یک رویداد با توجه به محدودیت های زمانی حاکم بر کارایی است. زمان پاسخ سیستم عموماً به زمان پردازش^۳ و یا زمان مسدود شدن^۴ ختم می شود.
- پردازش یک رویداد به منابع مختلفی نیاز دارد و توسط چندین مولفه^۵ سیستم، نیاز به رسیدگی دارد. مجموع زمان هایی که مولفه های مختلف سیستم با به کارگیری منابع به یک رویداد رسیدگی می کنند، زمان پردازشی رویداد به حساب می آید. در کنار زمان پردازش، زمان مسدود شدن نیز نقش مهمی در مجموع زمان تاخیر یک رویداد دارد؛ پردازش یک رویداد ممکن است به دلیل نیاز محاسباتی بخشی از پردازش به پردازشی دیگر و یا به دلیل در دسترس نبودن یک منبع مسدود شود. برای مثال، ممکن است زیرسیستم جستجوی نزدیک ترین راننده ها در برنامه ی تاکسی برخط منتظر دریافت جزئیات نقشه از زیرسیستم نقشه باشد.
- تاکتیک هایی که در کارایی وجود دارند به دو دسته ی زیر تقسیم می شوند:
- کنترل تقاضای منابع^۶ : این دسته از تاکتیک ها در سمت تقاضا فعالیت می کنند و سعی می کنند تا میزان تقاضا برای منابع را کاهش دهند.
 - مدیریت منابع^۷ : این دسته از تاکتیک ها سعی دارند تا کارها را با بازدهی بیشتری انجام دهند تا مصرف منابع از این طریق کاهش یابد.
- در ادامه به بررسی بیشتر این تاکتیک ها پرداخته می شود.

۱-۳-۵ کنترل درخواست منابع

یکی از راه های افزایش کارایی در حوزه نرم افزارهای حمل و نقل درون شهری، مدیریت دقیق تقاضای منابع است. این کار می تواند با کاهش تعداد رویدادهای پردازش شده با اعمال نرخ نمونه برداری یا با محدود کردن

¹Response

²Response Measure

³Processing time

⁴Blocking time

⁵Component

⁶Control resource demand

⁷Manage resources

سرعت پاسخگویی سیستم به رویدادها انجام شود.

- مدیریت نمونه برداری: با کاهش نرخ نمونه برداری از محیط هر چند مقداری از دقت کاسته خواهد شد اما میزان تقاضا رو می توان به مقدار خوبی کاهش داد. زمانی که جریان سازگار داده^۱ اهمیت بیشتری داشته باشد چشم پوشی از مقداری دقت می تواند راه حل خوبی برای کاهش تقاضا به سیستم باشد. برای مثال، در زیرسیستم نقشه‌ی برنامه‌ی تاکسی‌رانی برخط، ممکن است نرخ به روزرسانی محل کنونی خودرو را کاهش دهیم تا کارایی سرور در زمان‌هایی که بار زیادی وجود دارد، افزایش یابد.

- محدودیت پاسخ به رویدادها: زمانی که رویدادهای گسسته با نرخ بالایی به سیستم می‌رسند، باید در صف منتظر شروع پردازش بمانند. به دلیل گسسته بودن این رویدادها امکان کاهش نرخ نمونه برداری وجود ندارد، در چنین شرایطی سیستم می‌تواند انتخاب کند که با توجه به شرایطی نظیر اندازه‌ی صف و یا زمانی که تاکنون رویداد در پردازش به خود اختصاص داده است، حداکثر زمانی که یک رویداد می‌تواند پردازش بگیرد را محدود کند. با محدود کردن حداکثر زمان پردازشی که یک رویداد می‌تواند به خود اختصاص دهد، سیستم قادر خواهد بود تا رویدادهای قابل پیش‌بینی بیشتری را تضمین کند. در این روش زمانی که تصمیم گرفته می‌شود رویدادی رها^۲ شود باید سیاستی برای مدیریت این وضعیت از پیش تعیین شود: آیا رویدادهای رها شده ثبت می‌شود و یا به راحتی نادیده گرفته خواهد شد؟ آیا سایر سیستم‌ها و کاربران از رها سازی رویداد مطلع خواهند شد؟ در این زمینه می‌توان به مواقعی اشاره کرد که سیستم زمان زیادی را برای پیدا کردن سفرهای مشابه یک درخواستدر سیستم اشتراک خودرو صرف می‌کند و در چنین حالتی این تاکتیک توصیه می‌کند تا سرور این درخواست را رها کند تا با آزاد سازی منابع و پاسخ به درخواست‌های دیگر کارایی افزایش یابد.

- اولویت‌بندی رویدادها: اگر منابع کافی برای پاسخ دهی به تمامی رویدادها در اختیار سیستم قرار ندارد و یا اگر همه‌ی رویدادها از اهمیت یکسانی برخوردار نیستند استفاده از یک الگوریتم اولویت بندی می‌تواند تاثیر خوبی در کارایی سیستم داشته باشد. برای مثال، در برنامه‌ی یافتن پارکینگ‌های درون‌شهری سیستم می‌تواند درخواست مشتریان را بر اساس زمان پردازشی مورد نیاز اولویت‌دهی کند.

- کاهش سربار: هر چند وجود واسطه‌ها در افزایش اصلاح‌پذیری^۳ از اهمیت بالایی برخوردار است اما حذف این واسطه‌ها می‌تواند منجر به افزایش کارایی سیستم شود. معماری در این مساله با تقابل دو

¹Consistent Stream

²Drop

³Modifiability

نیاز کارایی و اصلاح‌پذیری دست و پنجه نرم می‌کند و باید بهترین تصمیم با توجه به نیازها و اهداف سیستم اخذ شود. برای مثال معماری می‌تواند بجای استفاده از چندین سیستم برای پاسخ به درخواست سفر توسط کاربر در برنامه‌ی تاکسی‌برخط از یک زیرسیستم یکپارچه استفاده کند و با حذف واسطه‌ها کارایی را در مجموع افزایش دهد؛ هرچند همان طور که بیان شد انجام چنین کاری سبب کاهش نیاز کیفی اصلاح‌پذیری خواهد شد.

- زمان اجرای محدود: محدودیتی در میزان استفاده از زمان اجرا برای پاسخگویی به یک رویداد تعیین کنید. برای الگوریتم‌های تکراری و وابسته به داده‌ها، محدود کردن تعداد تکرارها روشی برای محدود کردن زمان پردازش است که هر چند می‌تواند به پاسخی با دقت کمتر ختم شود اما در افزایش کارایی تاثیر مثبتی دارد. برای مثال، در برنامه‌ی پارکینگ‌های درون شهری، معماری می‌تواند زمانی که سرور برای پیدا کردن نزدیک‌ترین مسیر تا یک پارکینگ مشخص صرف می‌کند را با از دست دادن مقداری قابل چشم‌پوشی دقت، محدود کند.

- افزایش بازدهی منابع: بهبود الگوریتم‌های مورد استفاده در مناطق بحرانی تاخیر را کاهش می‌دهد.

۵-۳-۲ مدیریت منابع

هر چند ممکن است کنترل تقاضا برای منابع همواره ممکن نباشد اما مدیریت منابع همیشه در اختیار سیستم قرار دارد. گاهی می‌توان برای انجام یک پردازش از منابع مختلف استفاده کرد به عنوان مثال می‌توان در یک سیستم داده‌های میانی را در حافظه‌ی نهان^۱ نگهداری کرد و یا از حافظه‌های دیگر سیستم به منظور ذخیره سازی آن‌ها استفاده کنید. در ادامه چندین روش مدیریت منابع سیستم مطرح می‌شوند:

- افزایش تعداد منابع: افزایش کمیت و کیفیت منابع همواره یکی از راه‌های کاهش تاخیر رویدادهاست که البته دغدغه‌های مالی را بر سر راه خود دارد.

- استفاده از هم‌زمانی^۲: اگر درخواست‌ها قابلیت پردازش موازی را داشته باشند، استفاده از هم‌زمانی و پردازش موازی همراه با الگوریتم‌هایی جهت مدیریت پردازش رویدادها بر نخ‌های پردازشی متفاوت به جهت حفظ توان عملیاتی و عدالت می‌تواند کارایی سیستم را به مقدار خوبی افزایش دهد. برای مثال درخواست‌های کاربرهای یک سرویس تاکسیرانی آنلاین تا حد خوبی از یک دیگر مجزا هستند و می‌توان به صورت هم‌زمان انجام شود.

^۱Cache

^۲Concurrency

- حفظ چندین نسخه از محاسبات: وجود چندین سیستم مشابه برای پاسخ به رویدادها، در زمانی که چندین رویداد هم‌زمان با یکدیگر رخ می‌دهند می‌تواند کارایی سیستم را تا حد زیادی افزایش دهد. برای مثال وجود چنین سرور در معماری های کلاینت-سرور^۱ مانند آنچه که در سرویس تاکسیرانی آنلاین وجود دارد به همراه یک نرم افزار متعادل کننده ی بار^۲ می‌تواند به کارایی سیستم در پاسخ به رویدادها کمک زیادی کند.
- حفظ چندین نسخه از داده: تکثیر داده‌ها^۳ شامل نگهداری نسخه های جداگانه از داده‌ها برای کاهش نزاع میان رویدادها هنگام چندین دسترسی همزمان است. از آنجا که داده های تکثیر شده معمولاً یک کپی از داده های موجود است، حفظ کپی و هماهنگ سازی کپی ها به مسئولیتی تبدیل می‌شود که سیستم باید آن را بر عهده بگیرد. از این روش می‌توان در بیشتر نرم‌افزارهای این حوزه به خوبی استفاده کرد.
- محدودسازی اندازه ی صف: این روش که اصولاً در کنار محدودیت پاسخ به رویدادها استفاده می‌شود، اقدام به محدود سازی اندازه صف ورودی رویدادها می‌کند. در این روش باید سیاست مشخصی در برابر رویدادهایی که به دلیل محدودیت اندازه ی صف پذیرش نمی‌شوند، اتخاذ شود. برای مثال در اپلیکیشن پارکینگ زمانی که پارکینگ‌ها پر است و تعداد درخواست‌ها زیاد است می‌توان از محدود سازی صف استفاده کرد.
- برنامه‌ریزی منابع: در این روش هدف درک ویژگی های استفاده از هر منبع و انتخاب استراتژی برنامه ریزی سازگار با آن است.

۴-۵ طراحی فهرست بازبینی برای کارائی

در ادامه فهرستی برای پشتیبانی از روند طراحی و تجزیه و تحلیل نیاز کارایی ارائه شده است.

۴-۵-۱ تخصیص مسئولیت‌ها

در زمان تخصیص مسئولیت‌ها باید^۴ به موارد زیر توجه شود:

- مسئولیت‌های سیستم در شرایط مختلف نظیر بارسنگین^۵ یا نیاز بحرانی به پاسخ یک رویداد تعیین شود.

^۱Client-Server

^۲Load Balancer

^۳Data Replication

^۴Allocation of Responsibilities

^۵Heavy Load

برای این مسئولیت ها، الزامات پردازش هر مسئولیت شناسایی شده و تعیین شود که آیا ممکن است موجب گلوگاه^۱ های مختلف شود؟

- مسئولیت هایی که از وجود یک رشته کنترلی میان فرآیندها و یا مرزهای پردازشی حاصل می شوند، را شناسایی کنید.
- مسئولیت کنترل رشته های پردازشی در هنگام استفاده از روش های چندرشته ای^۲.
- اطمینان حاصل شود سیستم با وجود مسئولیت هایی که به عهده اش گذاشته شده است، می تواند به نیاز های کارایی پاسخ دهد.

۲-۴-۵ مدل هماهنگی

عناصر سیستم را که باید هماهنگ شوند - مستقیم یا غیرمستقیم - تعیین شود و رویکردهای ارتباطی و هماهنگی انتخاب شود که موارد زیر را انجام می دهند:

- پشتیبانی از هم زمانی رویدادها و روش های اولویت بندی و استراتژی های برنامه ریزی
- اطمینان حاصل شود که می توان پاسخ کارایی مورد نیاز را ارائه داد.
- سیستم در صورت لزوم می تواند ورودهای دوره ای، تصادفی یا پراکنده را پذیرش کند.
- از خصوصیات مناسب مکانیسم های ارتباطی برخوردار باشند.

۳-۴-۵ مدل داده

مدل داده^۳ ای آن بخش از داده ها که قرار است بار زیادی را متحمل شوند و در کارایی سیستم اثر گذار هستند مشخص شوند و موارد زیر در مورد آنها بررسی شود.

- آیا حفظ چندین نسخه از داده های کلیدی به نفع عملکرد است یا خیر؟
- آیا تقسیم^۴ داده ها به نفع کارایی است؟
- آیا کاهش نیاز های پردازشی برای عملیات هایی که نیاز به تغییر در داده ها دارند امکان پذیر است؟
- آیا امکان افزایش منابع به جهت کاهش گلوگاه ها در هنگام تغییر در داده ها امکان پذیر است؟

¹Bottleneck

²Multi-Threading

³Data Model

⁴Partitioning

۴-۴-۵ نقشه برداری در میان عناصر معماری

- در مواردی که بارگیری سنگین شبکه رخ می دهد، تعیین اینکه آیا قرارگیری همزمان برخی از اجزا باعث کاهش بارگذاری و بهبود کارایی کلی می شود؟
- اطمینان حاصل شود که مولفه ها با محاسبات سنگین تر به پردازنده هایی با بیشترین ظرفیت پردازش اختصاص داده شده اند.
- اطمینان حاصل شود استفاده از همزمانی ممکن است و آیا تاثیر مثبت قابل توجه ای بر روی کارایی سیستم دارد؟
- تعیین شود که آیا انتخاب رشته های کنترل و مسئولیت های مربوط به آنها گلوگاه هایی را ایجاد می کند یا خیر؟

۵-۴-۵ مدیریت منابع

- تعیین شود کدام منابع نقش حیاتی و مهمی در کارایی را ایفا می کنند. اطمینان حاصل شود این منابع در طول اجرا تحت شرایط عادی و بار کاری بالا تحت نظارت و مدیریت صحیح قرار می گیرند.

۶-۴-۵ زمان اتصال

موارد زیر برای مولفه هایی که پس از کامپایل^۱ به سیستم اضافه می شوند، تعیین شود:

- زمان لازم برای اضافه شدن مولفه به سیستم
 - میزان سربار اضافه ناشی از اضافه شدن دیر هنگام مولفه به سیستم
- باید اطمینان حاصل شود که این دو سربار جریمه خیلی زیادی بر روی سیستم نمی گذارند.

۷-۴-۵ انتخاب فناوری

آیا انتخاب فناوری شما امکان تنظیم موارد زیر را دارد:

- سیاست برنامه ریزی
- اولویت ها
- سیاست های کاهش تقاضا

¹Compile

- تخصیص بخشهایی از فناوری به پردازنده ها

- سایر موارد مربوط به کارایی

۵-۵ مطالعات موردی

در اینجا مطالعات موردی در ۳ حوزه حمل و نقل درون شهری برای نیاز کارایی انجام شده است.

۵-۵-۱ تاکسیرانی و حمل و نقل درون شهری

کارایی امروزه یکی از نیازهای مهم در هر نرم افزار است؛ برنامه هایی که کارایی لازم در مقایسه با رقبای خود دارا نیستند به راحتی بازار خود را به رقبای واگذار می کنند. کاربر انتظار دارد درخواست های او از سیستم در سریع ترین زمان ممکن پاسخ داده شود. همچنین در سیستم هایی مانند تاکسیرانی آنلاین اوپر، تمامی درخواست ها از اهمیت یکسانی دارا نیستند؛ سیستم باید با الویت دهی به درخواست هایی که محدودیت های سخت تری دارند کارایی سیستم را افزایش دهد [۳۷].

در برنامه های تاکسی آنلاین می توانیم با کاهش نرخ نمونه برداری تاکسی ها به درخواست های دیگر زیرسیستم ها منابع بیشتری را بدهیم. به عنوان نمونه، در برنامه MyWay، به صورت میانگین جستجوی سفرهای درون شهری حداکثر باید ظرف ۵ ثانیه انجام شوند و همچنین زمانی که کیفیت اینترنت در وضعیت عادی قرار دارد اکثر ارتباط های برنامه با سرور باید ظرف ۲ ثانیه صورت پذیرند [۲۲].

سناریو: کاربر با وارد کردن مقصد خود به دنبال راننده می گردد.

- منبع محرک: کاربر نهایی

- محرک: رویداد جستجوی راننده بر اساس مقصد

- محصول: سیستم

- محیط: محیط اجرای عادی

- پاسخ: راننده به کاربر معرفی می شود یا پاسخ داده می شود که راننده ای به این منظور وجود ندارد.

- اندازه گیری پاسخ: اندازه گیری تاخیر تا پاسخ دادن به کاربر (انتظار می رود تا ۵ ثانیه پیدا کند)

۵-۵-۲ اشتراک خودرو و حمل و نقل درون شهری

در برنامه های اشتراک خودرو یکی از زیرسیستم های اصلی جستجوی افراد با برنامه ی سفر و ویژگی های مشخص مشابه است. الگوریتمی که این زیرسیستم استفاده می کند باید از کارایی بالایی برخوردار باشد و

همچنین باید بتواند هم‌زمان به درخواست هزاران کاربر پاسخ دهد. شرکت‌هایی که در این زمینه فعالیت می‌کنند باید با استفاده از پردازش‌های چندرشته‌ای^۱ و تخصیص بهینه منابع به کارایی بالایی دست یابند. به عنوان مثال در سناریو عنوان شده سیستم پس از ۳۰ ثانیه باید نتایجی که به آن‌ها دست یافته است را به اطلاع کاربر نهایی برساند و منابع را برای پاسخ به سایر درخواست‌ها آزاد کند.

سناریو: کاربر با وارد کردن مقصد خود به دنبال همسفر می‌گردد.

- منبع محرک: کاربر نهایی
- محرک: رویداد جستجوی همسفر (راننده) بر اساس مقصد
- محصول: سیستم
- محیط: محیط اجرای عادی
- پاسخ: لیستی از همسفرها به کاربر پیشنهاد می‌شود
- اندازه‌گیری پاسخ: اندازه‌گیری تاخیر تا پاسخ دادن به کاربر (انتظار می‌رود تا ۳۰ ثانیه پیدا کند)

۵-۵-۳ پارکینگ‌های عمومی و حمل‌ونقل درون شهری

یکی از دغدغه‌های مهم در برنامه‌های این حوزه تغییرات زیاد در وضعیت فضا‌های خالی در پارکینگ‌هاست؛ برنامه‌هایی که چند ده هزار یا بیشتر پارکینگ را پوشش می‌دهند ممکن است در هر لحظه نیاز به هزاران نوشتن و خواندن از پایگاه داده داشته باشند. هر تغییر وضعیت در پارکینگ‌ها باید به سرعت و بدون تداخل در قالب تراکنش‌های اتمیک^۲ در پایگاه داده ثبت شوند. در چنین سیستم‌هایی خوب است که سیستم وضعیت پارکینگ‌هایی که بیشتر مورد استفاده قرار می‌گیرند را در حافظه‌هایی با سرعت بیشتر نسبت به سایر پارکینگ‌ها نگهداری کند.

سناریو: کاربر نهایی در یک شهر به جستجوی پارکینگ‌ها با ظرفیت خالی می‌گردد.

- منبع محرک: کاربر نهایی
- محرک: رویداد یافتن پارکینگ‌ها با ظرفیت خالی در سطح شهر
- محصول: سیستم
- محیط: محیط اجرای عادی

^۱Multi-Thread

^۲Atomic

- پاسخ: وضعیت پارکینگ ها با ظرفیت خالی توسط سیستم پردازش شده و به کاربر پاسخ داده می شود.
- اندازه گیری پاسخ: اندازه گیری تاخیر تا پاسخ دادن به کاربر (انتظار می رود تا ۵ ثانیه پیدا کند)

فصل ششم

امنیت

۶-۱ تعریف امنیت

امنیت^۱ یکی از نیازهای غیر عملکردی سیستم است که بوسیله آن می‌توان توانایی سیستم برای محافظت داده و اطلاعات از اشخاصی که مجوز دسترسی ندارند را نشان داد. در صورت وجود امنیت در سیستم‌های حوزه حمل‌ونقل درون شهری ممکن است که یک دسترسی بدون مجوز بخواهد که به داده کاربران دسترسی پیدا کند و آن را تغییر دهد یا باعث شود که سرویس برای بقیه کاربرهای دارای مجوز در دسترس نباشد.

امنیت ۳ مشخصه اصلی دارد که در ذیل خلاصه شده است [۳۳]:

۱. محرمانگی: در این معیار داده و سرویس‌ها از دسترس دسترسی‌های غیر مجوز دار حفظ می‌شود: برای مثال در سرویس تاکسی آنلاین داده‌های کاربران تنها در اختیار افراد معتبر خواهد بود و هرکسی به آن دسترسی نخواهد داشت.

۲. اصالت: در این معیار اصالت داده‌ها تضمین می‌شود و دست‌خوش تغییرات نمی‌شود: برای مثال در سیستم پارکینگ اگر فضای خالی وجود دارد و نرم‌افزار خالی بودن را نشان می‌دهد این داده اصالت دارد و تنها در صورتی که آن فضا پر شود داده نیز نشان دهنده پر بودن آن است و توسط افراد غیر مجاز دست‌خوش

¹ Security

تغییرات نباید بشود.

۳. در دسترس‌پذیری: در این معیار سیستم برای افراد دارای مجوز در دسترس خواهد بود و برای مثال استفاده از حمله منع سرویس نباید باعث شود که به در دسترس‌پذیری سرویس لطمه وارد شود: برای مثال ممکن است که یک حمله کننده تعداد زیادی درخواست به سرویس اشتراک خودرو دهد و سرویس آن بدلیل بار زیاد از دسترس کاربران دیگر خارج شود.

همچنین مشخصه‌های دیگری را نیز می‌توان برای امنیت لحاظ کرد که زیر مجموعه معیارهای قبلی هستند. در ذیل به ۳ مورد از آن‌ها اشاره شده است.

۱. احراز هویت: هویت افرادی که با سیستم درگیر هستند را تایید می‌کند: برای مثال در سیستم تاکسی آنلاین با استفاده از نام کاربری و رمز عبور و شماره همراه به تایید کاربر پرداخته می‌شود.

۲. عدم پذیرش مسئولیت: تضمین می‌کند که فرستنده در آینده فرستادن پیام خود را انکار نمی‌کند: برای مثال در سامانه اشتراک خودرو بین افرادی که یک خودرو را به اشتراک می‌گذارند عدم پذیرش مسئولیت پیام‌های رد و بدل شده بوجود نیاید که این موضوع ممکن است در آینده باعث غش در توافق بین آن‌ها بینجامد.

۳. احراز اصالت: به کاربر مجوزهای لازم برای انجام وظیفه مربوطه را اعطا می‌کند: برای مثال در سامانه پارکینگ افراد احراز هویت شده مجوز لازم برای رزرو جای پارک را داشته باشند.

۲-۶ سناریوی عمومی امنیت

سناریوی عمومی امنیت به صورت زیر است:

- منبع محرک^۱: انسان یا سیستمی که تشخیص داده شده است حال چه به صورت صحیح چه ناصحیح. برای مثال یک حمله کننده انسان خارج از شرکت سرویس تاکسی آنلاین
- محرک^۲: یک درخواست بدون مجوز برای نمایش داده، تغییر، حذف یا دسترسی به سرویس‌ها: یک درخواست بدون مجوز توسط یک فرد تایید نشده برای حذف رانندگان مجاز از سامانه اشتراک خودرو
- محصول^۳: سرویس‌های سیستم، داده‌های درون سیستم، مولفه‌ها و منابع سیستم: برای مثال تمام منابع موجود در سیستم‌های نرم‌افزاری حوزه حمل و نقل درون شهری

¹Source of stimulus

²Stimulus

³Artifact

• محیط^۱: سیستم که می‌تواند به صورت برون خط یا برخط باشد، به شبکه متصل باشد یا نباشد، پشت دیواره آتش باشد یا نباشد: برای مثال سیستم‌های برخط حوزه حمل و نقل درون شهری که پشت دیواره آتش هستند

• پاسخ^۲: بسته به نوع محرک و محصول پاسخ‌هایی بدین شکل تولید خواهد شد: داده و سرویس از دسترسی بدون مجوز محافظت شود، تغییر نیابد. منابع و سرویس‌ها در دسترس باشد و ... همچنین تمامی فعالیت‌ها ضبط شود تا بعداً بتوان در صورت بروز خرابی علت را پیگیری کرد.

• اندازه‌گیری پاسخ^۳: می‌توان معیارهای متفاوتی در نظر گرفت: چه مقدار طول کشیده است تا سیستم بعد از یک حمله موفق دوباره بازیابی شود، چه مقدار از مولفه‌های سیستم یا داده مورد خرابی قرار گرفته است یا چقدر طول کشیده است تا متوجه حمله شود.

برای دست‌یابی به هر نیازمندی مجموعه‌ای از تکنیک‌ها را به کار می‌بندیم که در مورد امنیت می‌توان به ۴ دسته شناخت، مقاومت، عکس‌العمل و بازیابی اشاره کرد.

برای شناخت حمله می‌توان از مقایسه کردن ترافیک شبکه یا سرویس یا مقایسه پترن‌های سیستم و پیدا کردن رفتارهای غیر معمول بهره برد. همچنین از روی چنین پترن‌هایی می‌توان به حمله منع سرویس نیز پی برد یا از روی دیرکرد پیام به آن مشکوک شد. برای چک کردن اصالت داده‌ها نیز می‌توان از روش‌های چک کردن اصالت داده مانند کدهای خطا^۴ و توابع هش^۵ استفاده کرد.

به منظور مقاومت می‌توان منابع ورودی به سیستم را شناسایی کرد و مطمئن شد که آن منبع دقیقاً همان کس یا چیزی است که باید باشد. به منابع مطمئن مجوزهای لازم برای تغییر و دسترسی به داده اعطا شود و تا حد ممکن دسترسی به منابع محدود شود. همچنین منابع را از نظر فیزیکی تا جای ممکن ایزوله باشد تا اگر یکی مورد حمله قرار گرفت دیگری را تحت تاثیر قرار ندهد. همچنین می‌تواند از مندهای رمزنگاری بر روی داده و ارتباط استفاده کرد و کاربر را مجبور کرد که تنظیمات خود را از حالت معمول خارج کرده و آن را تغییر دهد. عکس‌العمل مناسب در برابر حمله شامل اقدامات باطل کردن دسترسی حمله به منابع حتی برای کاربرهای مجاز است تا زمانی که حمله رفع شود. اگر تلاش‌ها مکرری برای دسترسی موفقیت‌آمیز نبود، دسترسی محدود و قفل شود. به اپراتورها و افراد مناسب در زمان اتفاق افتادن امری مشکوک یا تشخیص حمله اطلاع داده شود. به منظور بازیابی مناسب از تاکتیک‌های مربوط به دسترس‌پذیری استاده شود تا منابع بازیابی شوند و از تمامی

¹Environment

²Response

³Response Measure

⁴Checksum

⁵Hash

اتفاقات رکورد تهیه شود تا بعداً بتوان حمله کننده را تشخیص داد و آن را ردیابی کرد. به منظور بهره‌بری از این تاکتیک‌ها می‌توان با توجه به هریک از پروژه‌های تاکسی آنالین، اشتراک‌گذاری خودرو و مدیریت پارکینگ یک فهرست بازبینی تنظیم کرد. این موضوع به بهبود نیاز امنیت پاسخ می‌دهد.

۳-۶ طراحی فهرست بازبینی برای امنیت

در ادامه فهرستی برای پشتیبانی از روند طراحی و تجزیه و تحلیل برای معیار امنیت ارائه شده است.

۱-۳-۶ تخصیص مسئولیت‌ها

در این گام نیاز است که تعیین شود چه مسئولیت‌هایی از سیستم نیاز است که امن شود. برای این مسئولیت‌ها نیاز است که مطمئن شد مسئولیت‌های اضافه تخصیص یافته است:

- منبع محرک شناسایی شود.
- منبع محرک احراز اصالت شود.
- منبع محرک احراز هویت شود.
- مجوزهای لازم برای دسترسی به داده و سرویس‌ها اعطا شود.
- داده‌ها رمزگذاری شود.
- اگر دردسترس‌پذیری کاهش یافت تشخیص داده شود و اقدام مناسب مانند محدود کردن دسترسی و مطلع کردن اپراتور صورت گیرد.
- بعد از حمله بازبینی صورت گیرد.
- کدهای خطا و مقادیر هش تایید شود.

۲-۳-۶ مدل هماهنگی

در این گام نیاز است که مکانیزم‌های لازم برای ارتباط با بخش‌های دیگر تعیین شود. همچنین باید مطمئن شد که این این مکانیزم‌ها از قابلیت احراز هویت و اصالت بهره‌می‌برند و داده به صورت رمز شده میان بخش‌های مختلف انتقال پیدا می‌کند. در نهایت باید مطمئن شد که مکانیزم‌هایی برای مانیتور کردن و تشخیص تقاضا منابع زیاد وجود دارد و در صورت بروز تقاضا آن‌ها را محدود کرد.

۶-۳-۳ مدل داده

در این بخش حساسیت فیلدهای مختلف داده تعیین شود و موارد زیر اجرا گردد:

- باید مطمئن شد که داده‌های با سطح حساسیت متفاوت به صورت جدا ذخیره شده است.
- باید مطمئن شد که برای دسترسی به داده‌های با سطح حساسیت متفاوت، دسترسی‌هایی با سطح متفاوت نیاز است.
- باید مطمئن شد که درخواست دسترسی به داده‌های حساس ضبط می‌شود و از داده‌های ضبط آن نیز به طور مناسب محافظت می‌شود.
- باید مطمئن شد که داده‌ها به شکل مناسب محافظت شده است و کلیدهای آن نیز جدا از داده‌ها ذخیره و محافظت می‌شوند.
- باید مطمئن شد که اگر داده‌ها به طور نامناسب تغییر پیدا کرد قابل بازیابی باشد.

۶-۳-۴ نگاشت در میان عناصر معماری

در این قسمت تعیین می‌شود که چگونه نگاشت جایگزین اجزا معماری ممکن است که نحوه خواندن و نوشتن و تغییر دادگان و تنظیم دسترسی‌ها و کاهش دردسترس پذیری را تحت تاثیر قرار دهد. همچنین تعیین می‌شود که چگونه نقشه‌برداری ضبط داده‌های دسترسی به دادگان و سرویس‌ها را تحت تاثیر قرار می‌دهد یا تقاضاهای زیاد سیستم را تشخیص می‌دهد. برای هر یک از این نقشه‌برداری‌ها باید مطمئن شد که مسئولیت‌های عنوان شده در بخش تخصیص مسئولیت وجود دارد.

۶-۳-۵ مدیریت منابع

در این بخش نیاز است که منابعی از سیستم که نیاز است شناسایی، مانیتور، احراز اصالت و دسترسی خاص به منابع داشته باشند تعیین شود. همچنین نیاز است که منابعی که نیاز است تا احراز هویت، دادن مجوز، اطلاع افراد مناسب، رکورد دسترسی به داده‌ها، رمزگذاری داده‌ها و تشخیص تقاضای زیاد و محدود کردن دسترسی داشته باشند تعیین شود. برای این منابع نیاز است که تعیین شود که افراد خاص سازمان چقدر به منابع مخصوصا منابع حساس دسترسی داشته باشند. چگونه منابع مانیتور شوند و مطمئن شد که بخش‌های لازم عملیات‌های امنیتی مورد نیاز را انجام می‌دهند. همچنین باید مطمئن شد که از منابع مشترک برای انتقال داده‌های حساس از یک کاربری با حق دسترسی به یک کاربری بدون حق دسترسی استفاده نمی‌شود.

۶-۳-۶ زمان اتصال

مواردی را در نظر بگیرید که یک مولفه دیررس قابل اعتماد نباشد. به این منظور باید از مکانیزم‌های مناسب برای مدیریت و اعتبارسنجی آن استفاده کرد. دسترسی آن مولفه می‌تواند به سرویس‌ها و داده‌ها مسدود شود و دسترسی‌های ضبط شود. همچنین دادگان نیز رمزگذاری و کلید آن‌ها در جایی که اجزا دیررس دسترسی ندارند ذخیره شود. تمام این موارد انجام شود تا مولفه ارزیابی شود و صلاحیت آن اثبات شود.

۶-۳-۷ انتخاب فناوری

از فناوری‌هایی که می‌توان بوسیله آن از داده‌ها محافظت نمود و حق دسترسی‌ها را به خوبی مدیریت کرد و قابلیت احراز هویت کاربران را فراهم نمود باید استفاده کرد. همچنین باید دقت داشت که تکنولوژی مورد انتخاب تاکتیک‌های مورد نیاز برای بحث امنیت را پشتیبانی کند.

۶-۴ مطالعات موردی

در اینجا مطالعات موردی در ۳ حوزه حمل و نقل درون شهری برای نیاز امنیت انجام شده است. همانگونه که قبلاً بیان شد باید بسته به قسمت‌های حساس پروژه از رمزگذاری داده‌ها و ارتباط، شناسایی، احراز هویت و اصالت، کاهش دسترسی‌پذیری، بازیابی، ذخیره مناسب کلیدها و کدهای هش و خطا استفاده کرد تا امنیت مورد نیاز تامین شود.

در مستندات پروژه MyWay اعلام شده است که اطلاعات کاربر ناشناس می‌ماند، پروفایل‌های کاربر در سرور ذخیره می‌شود و حریم شخصی و مدیریت مناسب انجام می‌شود. ذخیره پروفایل‌های کاربر با قوانین محلی سازگار است، ارتباطات برای داده‌های حساس به صورت رمز شده است. قابلیت ردیابی برای عملیات‌های حساس انجام شده توسط کاربران وجود دارد، به خصوص برای ادمین‌ها. همچنین این اپلیکیشن صحت داده و مجوزهای لازم برای دسترسی به داده را تضمین می‌کند. ادمین‌ها دسترسی لازم برای مشاهده و تغییر داده‌های درونی را نیز دارند. در آخر باید گفت که کاربران در این اپلیکیشن احراز هویت می‌شود و سطح دسترسی‌ها و مجوزهای مختلفی میان نقش‌های اپلیکیشن وجود دارد. کاربر احراز هویت شده می‌تواند به نقش‌های مختلفی نگاشت شود [۲۲]. همچنین با بررسی‌ای که از برنامه Uber، ParkWhiz و Waze-Carpool به عمل آمد مشخص شد که این اپلیکیشن‌ها از قابلیت‌های احراز هویت با استفاده از نام کاربری و رمز عبور، نگه داشتن نشست کاربر و دنبال کردن فعالیت‌های کاربر، ضبط رکوردهای درخواست کاربر به سرور، استفاده از متدهای رمزنگاری مناسب، نگه داشتن تاریخچه مناسب از دادگان، نسبت دادن برخی توابع خاص تنها به بعضی از ماژول‌ها، محدود کردن ارتباط تنها در قسمت‌هایی از برنامه، چک کردن صحت دادگان برای متغیرهای حیاتی،

دنبال کردن خطاها، استفاده از ارتباط امن ssl برخوردار هستند.

در این رابطه تمام موارد مورد مطالعه می‌توان سناریوهای زیر را نوشت:

سناریو: اگر یک اتکر بخواهد به جای یک کاربر دیگر اقدام به درخواست سروی کند باید از این درخواست بدون مجوز جلوگیری شود.

- منبع محرک: اتکر

- محرک: تلاش بدون مجوز برای درخواست سرویس (به جای یک کاربری دیگر)

- محصول: سرویس

- محیط: سیستم آنلاین

- پاسخ: جلوگیری از درخواست بدون مجوز وضبط تغییرات و مطلع کردن کاربر از این که آیا این تغییر با درخواست شما صورت گرفته است یا خیر

- اندازه گیری پاسخ: مقدار زمانی که طول کشیده شده این حمله شناسایی شود و چه مقدار آسیب وارد شده است.

سناریو: ممکن است اتکر سعی در انجام حمله منع سرویس کند تا از در دسترس پذیری سرویس تاکی رانی آنلاین بکاهد.

سناریو: هکر سعی می‌کند با ارسال تعداد زیادی درخواست (حمله ی منع سرویس) سرویس ها را از دسترس خارج کند.

- منبع محرک: اتکر

- محرک: حمله منع سرویس به جهت کاهش دسترس پذیری

- محصول: سرویس های سیستم

- محیط: سیستم آنلاین

- پاسخ: داده و سرویس ها برای کاربران مجاز در دسترس قرار بگیرد و به کمک دیواره آتش اتکر شناسایی و دسترسی آن محدود شود.

- اندازه گیری پاسخ: مقدار زمانی که سیستم نسبت به حمله آسیب پذیر بوده است (چقدر طول کشیده تا به حالت نرمال برگردد)

فصل هفتم

قابلیت آزمون

۱-۷ تعریف قابلیت آزمون

طبق تخمین ها بین ۱۵ تا ۲۵ درصد هزینه تولید یک سیستم به آزمون سیستم اختصاص دارد و بعضی مدیران هزینه آن را تا ۵۰ درصد برآورد می کنند [۱۰]. قابلیت آزمون^۱ به سهولت ساخت نرم افزار برای نشان دادن عیب های خود از طریق آزمون اشاره دارد. به طور خاص، قابلیت آزمون احتمال آن است که اگر برنامه دارای خطایی باشد، در آزمون خطا خود را نشان دهد. به صورت شهودی، اگر یک سیستم به راحتی خطا های خود را آشکار سازد، دارای قابلیت آزمون است

معیارهای پاسخ دهی برای قابلیت آزمون به میزان موثر بودن آزمون ها در کشف عیب ها و مدت زمان انجام آزمایش ها تا رسیدن به حد مطلوبی از پوشش^۲ مربوط می شوند.

۲-۷ سناریوی عمومی قابلیت آزمون

سناریوی عمومی قابلیت استفاده به صورت زیر است:

¹Testability

²Coverage

- منبع محرک^۱: هر کسی که برای سیستم آزمون طراحی می کند می تواند جز محرک های سناریو عمومی باشد. آزمون های طراحی شده می توانند به صورت دستی یا اتوماتیک اجرا شوند.
- محرک^۲: اجرای مجموعه ی خاصی از آزمون ها که می تواند به یکی از دلایل اضافه شدن مولفه ای جدید یا تحویل به مشریان صورت پذیرفته باشد.
- محصول^۳: سیستم یا بخشی از آن که مورد آزمون قرار می گیرد.
- محیط^۴: زمان طراحی، زمان توسعه، زمان کامپایل، زمان ادغام^۵، زمان استقرار^۶، زمان اجرا
- پاسخ^۷: پاسخ سیستم می تواند زیر مجموعه ای از موارد زیر باشد.
 - اجرای مجموعه ی آزمون ها و دریافت پاسخ آن ها
 - جمع آوری فعالیت هایی که به خطا در آزمون منجر شده اند
 - کنترل و نظارت بر وضعیت سیستم
- اندازه گیری پاسخ^۸: یک یا چند مورد از این موارد را می توان به عنوان اندازه گیری استفاده نمود.
 - تلاش برای یافتن یک عیب یا کلاسی از عیب ها
 - تلاش برای دستیابی به درصد معینی از پوشش فضا^۹
 - احتمال بروز خطا در آزمون بعدی
 - زمان انجام آزمون ها
 - طول طولانی ترین زنجیره وابستگی در آزمون
 - مدت زمان آماده سازی محیط آزمون

۳-۷ تاکتیک ها در قابلیت آزمون

هدف تاکتیک های قابلیت آزمون آسان تر ساختن انجام آزمون ها پس از پایان فرآیند توسعه اولیه سیستم است. روش هایی که برای افزایش قابلیت آزمون وجود دارند در دو دسته از تاکتیک های قابلیت آزمون قرار می

¹Source of stimulus

²Stimulus

³Artifact

⁴Enviroment

⁵Integration Time

⁶Deployment Time

⁷Response

⁸Response Measure

⁹Space Coverage

گیرند؛ دسته ی اول به قابلیت کنترل و نظارت بر سیستم می افزایند و از این طریق انجام آزمون ها را آسان تر می سازند و دسته ی دیگر بر محدود کردن پیچیدگی های سیستم تمرکز می کنند.

۱-۳-۷ کنترل و نظارت بر وضعیت سیستم

ساده ترین شکل کنترل و نظارت بر یک سیستم نرم افزاری با مجموعه ای از ورودی ها است. اجازه می دهید سیستم کار خود با ورودی های داده شده را انجام دهد و سپس خروجی آن را مشاهده می کنید. به صورت خاص روش هایی که در این دسته بندی وجود دارند شامل موارد زیر هستند:

- رابط های تخصصی^۱ : برخورداری از یک رابط این قابلیت را می دهد که در زمان آزمون نرم افزار یا اجرا عادی به مقادیر متغیر ها در برنامه دسترسی داشته باشیم.
- ضبط/پخش^۲ : بازایجاد حالتی که خطا در آن رخ می دهد در بسیاری از مواقع دشوار است. ضبط حالت هنگام فراخوانی یک رابط باعث می شود تا از این حالت برای ایجاد مجدد خطا استفاده شود.
- محلی سازی ذخیره سازی^۳ : برای راه اندازی یک سیستم یا زیرسیستم از یک حالت آغازین تصادفی به منظور آزمون، بهترین روش این است که آن حالت در یک مکان واحد ذخیره شود.
- منابع داده انتزاعی^۴ : استفاده از منابع داده ای انتزاعی این امکان را به شما می دهد تا داده های آزمون را راحت تر جایگزین کنید. به عنوان مثال ، اگر پایگاه داده ای از داده های قابل آزمون را در اختیار داشته باشید، می توانید معماری خود را طوری طراحی کنید که به راحتی در مواقع آزمون سیستم از پایگاه داده ی مخصوص آزمون استفاده کند.
- جعبه شنی^۵ : منظور از جعبه شنی این است که نمونه ای از سیستم را از دنیای واقعی جدا کنید تا بتوانید آزمونی را انجام دهید بدون آن که نگرانی ای در مورد عواقب آزمون شما را نگران کند. با استفاده از جعبه شنی می توانید نسخه ای از منابعی را که رفتار آن تحت کنترل شماست، بسازید.
- ادعاهای قابل اجرا^۶ : با استفاده از این تاکتیک ، ادعاها معمولاً به صورت دستی کد زده می شوند و در مکان های دلخواه قرار می گیرند تا نشان دهند چه زمانی و در کجا برنامه در وضعیت معیوبی قرار دارد.

¹Specialized interfaces

²Record/Playback

³Localize state storage

⁴Abstract data sources

⁵Sandbox

⁶Executable assertions

۷-۳-۲ محدودسازی پیچیدگی های سیستم

هر چقدر یک نرم افزار پیچیده تر باشد، آزمون آن سخت تر خواهد بود زیرا بنا بر تعریف پیچیدگی، فضای حالت عملیاتی یک برنامه ی پیچیده بسیار بزرگ است و ایجاد مجدد یک حالت خاص در یک فضای حالت بزرگ دشوارتر از انجام این کار در فضای حالت کوچک است. به طور خاص دو روش زیر به محدودسازی پیچیدگی نرم افزار کمک می کنند.

- محدودسازی پیچیدگی ساختاری: این تاکتیک شامل اجتناب یا حل وابستگی های حلقوی میان مولفه های سیستم، جداسازی و کپسول سازی^۱ وابستگی ها به محیط خارجی و کاهش وابستگی بین اجزا به طور کلی است. در این روش می توانید عمق درخت ارث بری^۲ و تعداد فرزندان یک کلاس را محدود کنید.
- دوری از عدم قطعیت^۳: این تاکتیک شامل یافتن تمام منابع عدم قطعیت، مانند موازی کاری غیرقانونی و حذف هرچه بیشتر آن ها است.

۷-۴ طراحی فهرست بازبینی برای قابلیت آزمون

۷-۴-۱ تخصیص مسئولیت ها

در مورد تخصیص مسئولیت ها موارد زیر باید بررسی شوند:

- تعیین شود مهم ترین مسئولیت های سیستم چه مسئولیت هایی هستند زیرا این مسئولیت ها باید به دقت مورد آزمون قرار گیرند.
- اطمینان حاصل شود مولفه هایی در سیستم وظایف زیر را بر عهده دارند.
 - اجرای تست ها و دریافت نتایج حاصل از آن ها
 - ذخیره سازی لاگ^۴ ها رفتارهای غیر قابل پیش بینی سیستم و لاگ خطا های رخ داده در سیستم
 - کنترل و نظارت بر حالت های مرتبط با سیستم به جهت آزمون سیستم
- اطمینان حاصل شود که تخصیص عملکرد ها انسجام بالا، اتصال^۵ کم، جدایی شدید نگرانی ها^۶ و پیچیدگی ساختاری کم را فراهم می کند.

^۱Encapsulation

^۲Inheritance

^۳Nondeterminism

^۴Logs

^۵coupling

^۶separation of concerns

۲-۴-۷ مدل هماهنگی

از مکانیسم های هماهنگی و ارتباط سیستم اطمینان حاصل کنید:

- پشتیبانی از اجرای تست ها و دریافت پاسخ آن ها در یک یا میان چندین سیستم
- پشتیبانی از دریافت تاریخچه خطا های رخ داد در یک سیستم و یا میان چندین سیستم
- پشتیبانی از تزریق^۱ و نظارت^۲ بر وضعیت در کانال های ارتباطی برای استفاده در آزمون ها، درون یک سیستم یا بین سیستم ها

۳-۴-۷ مدل داده

در بخش باید انتزاعات اصلی داده را که باید برای اطمینان از عملکرد صحیح سیستم مورد آزمون قرار گیرند را تعیین کنید.

- اطمینان حاصل شود که امکان گرفتن مقادیر نمونه های این انتزاعات داده وجود دارد.
- اطمینان حاصل شود که می توان مقادیر نمونه های این انتزاعات داده را هنگام تزریق حالت به سیستم تنظیم کرد؛ بنابراین می توانیم حالت سیستم که منجر به خطا شده است، دوباره قابل ایجاد است.
- اطمینان حاصل شود که هر گونه تغییر در این داده های انتزاعی ممکن و قابل آزمون است.

۴-۴-۷ نقشه برداری در میان عناصر معماری

نحوه آزمون نگاشت های احتمالی عناصر معماری (به ویژه نگاشت فرآیندها به پردازنده ها، رشته ها برای پردازش ها و ماژول ها به اجزا) را تعیین کنید تا پاسخ آزمون مورد نظر حاصل شود و وجود وضعیت رقابتی^۳ مشخص شود.

۵-۴-۷ مدیریت منابع

در این بخش باید از وجود منابع کافی جهت اجرای آزمون های اطمینان حاصل کنید و همچنین مطمئن شوید محیطی که آزمون ها در آن به اجر در می آیند نمود محیط واقعی که سیستم در آن فعالیت می کند است. در بحث مدیریت منابع باید به موارد زیر توجه شود:

- محدودیت منابع آزمون

¹Injection

²Monitoring

³Race Condition

- ثبت دقیق مصرف منابع در هنگام رویداد ها علی الخصوص خطا ها
- وضع محدودیت های جدید بر روی منابع برای آزمون شرایط خاص
- ایجاد منابع مجازی با هدف آزمون

۶-۴-۷ زمان اتصال

در این بخش باید اطمینان حاصل شود مولفه هایی که پس از زمان کامپایل دیر هنگام به سیستم متصل می شوند از قابلیت آزمون برخوردار هستند.

۷-۴-۷ انتخاب فناوری

انتخاب فناوری باید به گونه ای باشد که سناریوهای قابلیت آزمون را بر روی سیستم بتوان اجرا کرد.

۵-۷ مطالعات موردی

هر ۳ حوزه مطرح شده دارای نرم افزارهای آنلاین به صورت حداقل موبایل و وب هستند. در هنگام توسعه، توسعه دهندگان این برنامه ها از یونیت تست^۱ برای پوشش آزمون های هر مولفه نرم افزار استفاده می کنند که این پروسه تولید نرم افزار بر اساس معیارهای TDD^۲ طراحی و انجام می شود.

همچنین برای پوشش کیفیت تست ها از نرم افزارهای سنجش کد مانند سونارکیوب^۳ همراه با جنکینز^۴ و خط لوله های^۵ تولید نرم افزار استفاده می شود. همچنین برای آزمون عملکرد برنامه های تحت وب نیز از سلنیوم^۶ استفاده می شود.

همچنین موارد دیگری از تست نیز وجود دارد که به نام تست A/B معروف هستند و فیدبک کاربر را نسبت به تغییرات جدید می سنجند معمولاً هر آپدیت شرکت هایی مثل اوبر قبل از عرضه نهایی به همه کاربران از این نوع آزمون نیز عبور می کند [۳۸].

سناریوهای زیر را می توان در نظر گرفت:

سناریو: کاربران نهایی که در تست A/B آخرین تغییر اپلیکیشن تاکسی رانی آنلاین شرکت کردند و شکست آپدیت جدید را گزارش می کنند.

- منبع محرک: کاربران نهایی که به منظور تست انتخاب شدند

¹Unit Test

²Test-driven development

³Sonarqube

⁴Jenkins

⁵Pipeline

⁶Selenium

- محرک: فیدبک شکست
- محصول: زمان اجرا
- محیط: اپ تغییر یافته
- پاسخ: ثبت خطاها و فیدبک‌ها
- اندازه گیری پاسخ: احتمال اینکه با توجه به تعداد کاربرانی که در این آزمون شرکت کردند خطاها و دلایل شکست کشف شود.
- سناریو: تست توسط ابزارهای اتومات تست کل سیستم اشتراک گذاری خودرو در هنگام استقرار
- منبع محرک: ابزار اتومات تست
- محرک: پیاده سازی کامل سیستم
- محصول: زمان استقرار
- محیط: سیستم
- پاسخ: انجام تست‌ها و خروجی نهایی و کنترل حالت‌های سیستم
- اندازه گیری پاسخ: طول بلندترین زنجیره تست، زمان انجام تست‌ها
- سناریو: تست مجموعه‌ای از موارد آزمون توسط tester unit برای بخش apple pay پارکینگ آنلاین که توسعه آن به تازگی به پایان رسیده است.
- منبع محرک: tester unit
- محرک: مجموعه‌ای از تست‌ها
- محصول: قسمتی از سیستم (apple pay) که در حال تست است.
- محیط: زمان توسعه
- پاسخ: تست‌ها انجام می‌شود و تعداد تست‌های موفقیت آمیز و شکست خورده و زمانی که هر تست طول کشیده گزارش می‌شود.
- اندازه گیری پاسخ: میزان تلاش برای پیدا کردن خطا

فصل هشتم

قابلیت استفاده

۸-۱ تعریف قابلیت استفاده

قابلیت استفاده^۱ به این تعریف می‌پردازد که چقدر برای کار انجام دادن یک وظیفه دلخواه راحت است. این ویژگی یکی از ارزان و راحت‌ترین ویژگی‌هایی است که می‌توان به وسیله آن کیفیت سیستم را افزایش داد. قابلیت استفاده شامل موارد زیر است:

- ویژگی‌ها سیستم یادگیری
- استفاده موثر از یک سیستم
- به حداقل رساندن تاثیر خطاها
- تطبیق سیستم با نیازهای کاربر
- افزایش اعتماد به نفس و رضایت

^۱Usability

۸-۲ سناریوی عمومی قابلیت استفاده

سناریوی عمومی قابلیت استفاده به صورت زیر است:

- منبع محرک^۱ : کاربر نهایی
- محرک^۲ : کاربر نهایی که سعی می‌کند از سیستم به طور موثر استفاده کند و آن را یاد بگیرد و تاثیر خطاها را کمینه کند، با سیستم تطبیق پذیرد و آن را تنظیم کند.
- محصول^۳: سیستم یا بخشی از سیستم که کاربر در حال تعامل با آن است.
- محیط^۴: در مرحله اجرا یا در مرحله کانفیگ
- پاسخ^۵ : سیستم باید ویژگی‌های مورد نیاز کاربر را فراهم کند یا با توجه به نیازهای کاربر آن را پیش‌بینی کند.
- اندازه گیری پاسخ^۶: یک یا چند مورد از این موارد را می‌توان به عنوان اندازه گیری استفاده نمود. زمان تسک، تعداد خطاها، تعداد تسک‌های تکمیل شده، میزان رضایت کاربر، درصد موفقیت‌آمیز عملیات‌ها نسبت به کل عملیات‌ها یا میان زمان یا داده از دست رفته زمانی که خطا رخ داده است.
- محققان حوزه ارتباط انسان و کامپیوتر از اصطلاحات ابتکار کاربر^۷ ، ابتکار سیستم و یا ابتکار درهم استفاده می‌کنند تا توصیف کنند که کدام جفت انسان- کامپیوتر ابتکار عمل خاصی را انجام می‌دهند و تعامل گونه پیش می‌رود.
- از تمایز بین ابتکار کاربر و سیستم برای بحث در مورد تاکتیک‌های قابلیت استفاده می‌توان بهره برد و سناریوهای مختلف را بررسی کرد. هدف نهایی این تاکتیک‌ها آن است که به کاربر فیدبک و یاری مناسب داده شود.
- دسته اول تاکتیک‌ها پشتیبانی ابتکار کاربر است. که می‌توان آن‌ها را به صورت زیر طبقه بندی کرد.
- کنسل: سیستم باید به درخواست کنسل کردن گوش دهد و دستوری که کنسل شده متوقف شود و منابع آن آزاد گردد و مولفه‌های وابسته نیز اطلاع داده شوند.

¹Source of stimulus

²Stimulus

³Artifact

⁴Environment

⁵Response

⁶Response Measure

⁷User Initiative

- توقف/ادامه: سیستم در صورت توقف باید به صورت موقت منابع را آزاد کند که ممکن است بوسیله تسک‌های دیگر دوباره تخصیص داده شوند.
- بازگشت: باید این امکان وجود داشته باشد که یک یک حالت قبلی به درخواست کاربر بازگردانی شود.
- ترکیب: امکان ترکیب اشیا با سطح پایین به گروه. در این صورت کاربر می‌تواند یک عملیات را بر روی یک گروه اعمال کند و نه بر روی تک تک اشیا سطح پایین.
- دسته دوم تاکتیک‌ها پشتیبانی ابتکار سیستم است. که می‌توان آن‌ها را به صورت زیر طبقه بندی کرد.
- حفظ مدل تسک: زمینه را تعیین می‌کند تا سیستم بتواند از آنچه کاربر در تلاش است ایده ای داشته باشد و به او کمک کند.
- حفظ مدل کاربر: صریحا دانش کاربر از سیستم، رفتار کاربر از نظر زمان پاسخ مورد نظر و غیره را نشان می‌دهد.
- حفظ مدل سیستم: سیستم یک مدل صریح از خود را حفظ می‌کند. که از آن به منظور تعیین رفتار سیستم استفاده می‌شود تا برخورد مناسب به کاربر ارائه شود.

۸-۳ طراحی فهرست بازبینی برای قابلیت استفاده

۸-۳-۱ تخصیص مسئولیت‌ها

اطمینان حاصل شود که مسئولیت‌های اضافی در سیستم در صورت لزوم برای کمک به کاربر در موارد زیر در نظر گرفته شده است:

- یادگیری نحوه استفاده از سیستم
- دستیابی موثر به وظیفه موجود
- تطبیق‌پذیری و پیکربندی سیستم
- بازیابی از خطاهای کاربر و سیستم

۸-۳-۲ مدل هماهنگی

در این قسمت باید تعیین شود که عناصر سیستم بر چگونگی یادگیری کاربر برای استفاده از سیستم، کامل کردن تسک‌ها، تنظیم سیستم و بازیابی خطاها و سیستم، رضایت و اعتماد به نفس کاربر تاثیر می‌گذارد. برای مثال آیا

می‌توان سیستم به رویدادهای حرکت کاربر بر روی منوها به خوبی پاسخ دهد و در زمان واقعی بازخورد مناسب بدهد؟

۸-۳-۳ مدل داده

در این قسمت انتزاعات عمده داده را که با رفتار قابل درک کاربر درگیر هستند تعیین می‌شود. باید اطمینان حاصل کرد که این انتزاعات اصلی داده‌ها، عملکردهای آنها و خصوصیات آنها برای کمک به کاربر در دستیابی به وظیفه پیش رو، سازگاری و پیکربندی سیستم، بازیابی از خطاهای کاربر و سیستم، یادگیری نحوه استفاده از سیستم و افزایش رضایت کاربر طراحی شده است. به عنوان مثال، انتزاعات داده باید به گونه‌ای طراحی شوند که از عملیات بازگشت و لغو پشتیبانی را پشتیبانی کند: و ریزدانی این عملیات‌ها به گونه‌ای باشد که بیش از حد طولانی نشود.

۸-۳-۴ نقشه برداری در میان عناصر معماری

در این مرحله باید تعیین شود که چه نقشه برداری از بین عناصر معماری برای کاربر نهایی قابل مشاهده است (به عنوان مثال میزان آگاهی کاربر نهایی از اینکه کدام سرویس‌ها محلی و کدام یک از راه دور هستند). برای آن‌هایی که قابل مشاهده هستند، تعیین شود که این قابل مشاهده بودن چگونه بر روی سهولت یادگیری کاربر برای استفاده از سیستم، دستیابی به وظیفه پیش رو، سازگاری و پیکربندی سیستم، بازیابی از خطاهای کاربر و سیستم و اعتماد به نفس و رضایت کاربر تاثیر می‌گذارد.

۸-۳-۵ مدیریت منابع

در این قسمت تعیین می‌شود که کاربر چگونه می‌تواند از منابع سیستم استفاده کند و پیکربندی را انجام دهد. باید اطمینان حاصل شود که محدودیت‌های منابع تحت تنظیمات تمام کنترل شده توسط کاربر، احتمال رسیدن کاربران به وظایفشان را کمتر نخواهد کرد. به عنوان مثال، از تنظیماتی که منجر به زمان پاسخ بیش از حد طولانی می‌شود، جلوگیری باید کرد. همچنین اطمینان حاصل شود که سطح منابع بر توانایی کاربران در یادگیری نحوه استفاده از سیستم تأثیر نگذارد، یا میزان اطمینان و رضایت آنها از سیستم را کاهش ندهد.

۸-۳-۶ زمان اتصال

در این بخش تصمیمات مربوط به زمان اتصال باید تحت کنترل کاربر باشد و اطمینان حاصل شود که کاربران می‌توانند تصمیماتی بگیرند که به قابلیت استفاده کمک می‌کند. به عنوان مثال، اگر کاربر می‌تواند در زمان اجرا، پیکربندی سیستم یا پروتکل‌های ارتباطی آن یا عملکرد آن را از طریق افزونه‌ها انتخاب کند، باید اطمینان

حاصل کرد که چنین گزینه هایی بر توانایی کاربر در یادگیری ویژگی های سیستم، استفاده موثر از سیستم، به حداقل رساندن تأثیر خطاها، سازگاری و پیکربندی بیشتر سیستم یا افزایش اطمینان و رضایت تأثیر منفی نمی گذارد.

۸-۳-۲ انتخاب فناوری

انتخاب فناوری باید به گونه ای باشد که سناریوهای قابلیت استفاده را بر روی سیستم بتوان اجرا کرد. باید اطمینان پیدا کرد که تکنولوژی انتخاب شده تأثیر برعکس ندارد و به ویژگی های یادگیری سیستم خللی وارد نمی کند.

۸-۴ مطالعات موردی

در اینجا مطالعات موردی در ۳ حوزه حمل و نقل درون شهری برای نیاز قابلیت استفاده انجام شده است.

۸-۴-۱ تاکسیرانی و حمل و نقل درون شهری

واسط منطقی باید به راحتی قابل فهم و استفاده از آن راحت باشد. برای جلوگیری از بروز دادن بسیاری از خطاها راهنمایی به کاربر و محدود کردن فیلدها به موارد مجاز انجام شود. کلیدهای بازگشت و لغو کردن وجود داشته باشد از ریزدانگی مناسب برخوردار باشد. امکان گروه کردن و انتخاب موارد در برنامه وجود داشته باشد، برای مثال حذف لیست سفرها

در مقاله [۳۶] نسبت به بررسی قابلیت استفاده اپلیکیشن موبایل uber پرداخته شده است، به طور خلاصه در این مقاله بیان شده است که چون اکثر کاربران از تنها انگشت شست خود برای کاربر کردن با برنامه های موبایل استفاده می کنند محل ثبت نام و ورود برنامه در مکانی که برای این انگشت راحت تر است گذاشته شده همچنین اگر برنامه به لوکیشن شما دسترسی داشته باشد در هنگام استفاده از نقشه به منظور تعیین لوکیشن ابتدا به محل لوکیشن شما می رود تا مبدا را راحت تر پیدا کنید. همچنین برنامه مطمئن می شود که در هیچ صفحه ای مقدار زیادی اطلاعات به شما ندهد که شما با حجم عظیمی از اطلاعات گیج شوید. همچنین اسلایدبار در کنار صفحه وجود دارد که به راحتی هم گزینه کمک را در آنجا می توانید پیدا کنید و هم زمانی که در صفحات خود برنامه هستید. فیدبک مناسب از زمان سفر شما را بهتون خواهد داد و از شما در مورد سرفرتان نظرخواهی خواهد کرد. همچنین وقتی تاکسی شما در راه رسیدن به شماست اطلاعات کاملی از راننده و ماشین آن ارائه می دهد تا شما احساس بهتری داشته باشید که تاکسی شما در راه رسیدن به شماست.

سناریو: خطاهای کاربر در هنگام ورود به اپلیکیشن تاکسی آنلاین و کار با منوها کاهش یابد.

- منبع محرک: کاربر نهایی
- محرک: تاثیر خطاهای کاربر کمینه شود.
- محصول: سیستم، در کلاینتی که کاربر در حال حاضر با آن کار می‌کند.
- محیط: زمان اجرا
- پاسخ: گزینه فراموشی و بازگرداندن رمز عبور، استفاده از اسلایدبار، عملیات لغو، بازگشت
- اندازه گیری پاسخ: تعداد خطاها، تعداد مشکلات حل شده

۸-۴-۲ اشتراک خودرو و حمل و نقل درون شهری

استفاده راحت و قابل فهم، در هنگام پیدا کردن بهترین مسیر با استفاده از یک نوار پیشرفت کاربر باید از میزان پیشرفت مطلع شود. از وضعیت کاربر مطلع بوده و پیشنهادات مناسب به کاربر داده شود تا از پیچیدگی انجام تسک‌ها توسط کاربر بکاهد. کلیدهای لغو درخواست و معوق کردن آن‌ها وجود داشته باشد. همانند uber ، waze-carpool نیز از قابلیت استفاده بالایی برخوردار است، ثبت نام و ورود آن در نقاط مناسبی جایگذاری شده، دکمه‌های راهبر نیز به خوبی نمایان هستند و با علامت‌های مناسب نظیر چرخنده برای تنظیمات به کاربر مفهوم را منتقل می‌کنند. در ۳ گام پشت سر هم می‌توان نسبت به برنامه ریزی سفر، پیدا کردن رانندگان و مطلع شدن از برنامه سفر با آن کار کرد. برای اطلاع دادن به کاربر از پوش نوتیفیکیشن استفاده می‌کند که کار را خیلی راحت‌تر می‌کند.

سناریو: کاربر نهایی طی چند مرحله برنامه‌ی سفر را در برنامه اشتراک خودرو وارد می‌کند که ناگهان تصمیم به تغییر یکی از قسمت‌های برنامه‌ی سفر می‌گیرد.

- منبع محرک: کاربر نهایی
- محرک: درخواست بازگشت به مراحل قبل وارد کردن اطلاعات برنامه سفر
- محصول: زیر سیستم مربوط به برنامه ریزی سفر
- محیط: زمان اجرا
- پاسخ: سیستم از حالت چند گامه برخوردار بوده و امکان بازگشت به عقب و ویرایش برنامه مسافرت را به او می‌دهد، همچنین از گزینه‌های با شکل مفهومی استفاده شده که بیشترین فیدبک به کاربر از گزینه‌ها داده شود.

- اندازه گیری پاسخ: میزان هدر رفت زمان و داده ها در هنگام رخداد خطا چقدر است؟ میزان رضایت کاربر از عملکرد چقدر است؟

۸-۴-۳ پارکینگ های عمومی و حمل و نقل درون شهری

نرم افزار به راحتی قابل استفاده باشد و با استفاده از علامت ها و نشانه ها و نمایش گذاری به فهم بهتر کاربر کمک کند. امکان لغو درخواست و بازگشت به عقب و فیدبک دادن به کاربر وجود داشته باشد.

در مقاله [۲۴] نسبت به بررسی طراحی اپلیکیشن Parkwhiz پرداخته شده است. علاوه بر رعایت نکات پایه علامت ها و نشانه ها وجود دکمه های راهنما و حرکت موارد خاصی در این برنامه نیز وجود دارد. در این برنامه کاربران می توانند آدرس مقصد و میزان ساعتی که پارکینگ را در دسترس می خواهند به عنوان گزینه وارد کنند. نه تنها این اپلیکیشن گزینه های پارک برای کاربر را بر اساس قیمت مقدار ساعت وارد شده نشان می دهد بلکه زمانی که نیاز است تا از آن پارکینگ به آن مقصد کاربر برسد را نیز نشان می دهد. این ویژگی منحصر به فرد باعث می شود که دیگر نیازی به مقایسه زمان بر و طاقت فرا بین گزینه ها کاربر نبپردازد و سریع گزینه مطلوب را انتخاب کند. این اپ با google-map و waze یکپارچه ست و برای رسیدن به پارکینگ نزدیک مقصد نیازی نیست که کاربر به صورت دستی از آن ها استفاده کند و مستقیم به آن ها هدایت می شود. همچنین اطلاعاتی از پارکینگ و این که چه زمانی حدودا به پارکینگ می رسید نیز خیلی برای کاربرانی که اولین بار است از این اپ استفاده می کنند مفید است. همچنین پرداخت را با ساخت یک بارکد راحت می کند و مکان هایی که تابحال در آن ها پارک کرده اید را راحت می توانید از صفحه اصلی دوباره-رزرو کنید.

سناریو: کاربر جدید باید بتواند در فرصت کوتاهی بتواند ویژگی های سیستم را یاد بگیرد.

- منبع محرک: کاربرنهایی
- محرک: کاربر می خواهد رزرو پارکینگ به کمک این سیستم را یاد بگیرد و به طور موثر از آن ها استفاده کند.

- محصول: سیستم، در کلاینتی که کاربر در حال حاضر با آن کار می کند.

- محیط: زمان اجرا

- پاسخ: اینترفیس آشنا و شبیه بقیه سرویس هایی که با نقشه کار می کنند باشد و اطلاعاتی از پارکینگ و این که چه زمانی حدودا به پارکینگ می رسد به او بازگردانده شود.

- اندازه گیری پاسخ: میزان رضایت کاربر

فصل نهم

معماری نمونه مطالعاتی تاکسی آنالاین

۹-۱ معماری میکروسرویس دامنه‌گرا

اوبر^۱ دارای بیش از ۲۲۰۰ میکروسرویس است [۱۴] و وجود تعداد زیادی میکروسرویس باعث پیدایش پیچیدگی زیادی در نگهداری این سرویس‌ها و توسعه سرویس‌های جدید می‌شود تا جایی که Uber سعی کرد با تغییراتی در معماری میکروسرویس، معماری میکروسرویس دامنه‌گرا^۲ را ارائه دهد؛ در معماری میکروسرویس دامنه‌گرا سعی شده است تا با حفظ مزیت‌های معماری میکروسرویس، از پیچیدگی کل سیستم کاسته شود.

در معماری میکروسرویس، سرویس‌ها با عملکرد محدود به یک حوزه بر روی یک شبکه مستقر می‌شوند و از طریق رابط‌های تعریف شده به درخواست‌های که به صورت Remote Procedure Call ارسال می‌شوند پاسخ می‌دهند؛ اوبر به دلایل زیر در سال ۲۰۱۲ معماری خود را از monolithic به micro-service تغییر داد:

- ریسک‌های دردسترس‌پذیری^۳ : یک خطا در سیستم monolithic می‌توانست منجر به از دست خارج شدن کل سیستم Uber شود.

- استقرار^۴ های پرخطر و زمان‌بر

^۱Uber

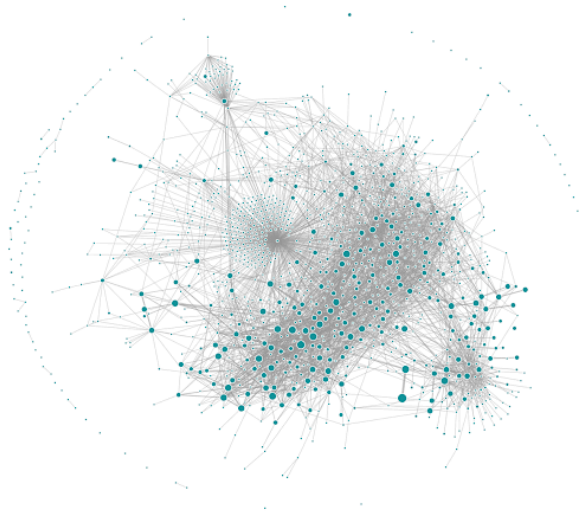
^۲Domain-Oriented Microservice Architecture

^۳Availability

^۴Deployment

- تفکیک ضعیف نگرانی‌ها^۱ : باوجود یک پایگاه کد بزرگ تفکیک مرز میان منطق تجاری و مولفه‌ها در یک سیستم با رشد بسیار سریع به خوبی صورت نمی‌پذیرد.
- اجرای ناکارآمد : وجود مشکلاتی که به چندین تیم وابستگی دارد، سبب می‌شود تا کارایی در اجرا بسیار پایین باشد.

هر چند مهاجرت از سیستم با معماری monolithic به microservice در زمانی که اندازه شرکت اوپر به صدها مهندس رسیده بود بسیاری از مشکلات را حل کرد، اما زمانی که تعداد میکروسرویس‌ها افزایش یافت شرکت متوجه پیچیدگی روزافزون معماری میکروسرویس شد؛ به عنوان مثال گاهی نیاز است برای یافتن ریشه‌ی یک خطا چندین میکروسرویس از تیم‌های مختلف مورد بررسی قرار گیرند. همچنین وابستگی میان سرویس‌ها گاه سبب می‌شود تا تاخیر پاسخ یک سرویس دیگر قابل قبول نباشد. همان طور که از پیچیدگی شبکه Uber در سال ۲۰۱۸ ۹-۱ مشخص است میکروسرویس‌ها شدیداً به یکدیگر وابسته هستند.



به جهت پیاده‌سازی یک امکان جدید در Uber یک تیم باید با سرویس‌های متفاوت متعلق به تیم‌های متفاوت کار کند و این عمل به جلسات زیادی بر روی طراحی و بازبینی کد نیاز دارد. همچنین مزیت معماری میکروسرویس در مورد داشتن خطوط مشخص مالکیت سرویس، هنگامی که تیم‌ها در سرویس‌های یکدیگر کد ایجاد می‌کنند، مدل‌های داده یکدیگر را اصلاح می‌کنند و حتی از طرف دارندگان سرویس نسخه‌های جدید را استقرار^۲ می‌دهند، به خطر می‌افتد. در نتیجه با افزایش تعداد سرویس‌ها گویا با یک سیستم Network Monolithic سر کار داریم که یکی از نتایج آن این است که چندین میکروسرویس به ظاهر مستقل نیاز دارند تا همزمان در سامانه مستقر شوند تا سامانه بدون نقص به عملکرد خود ادامه دهد.

^۱Concerns

^۲deployment

در ”معماری سرویس دامنه‌گرا“ به طور عمده از روشهای تثبیت‌شده ساختاردهی کد مانند طراحی مبتنی بر دامنه^۱ [۴۶]، معماری تمیز^۲ [۴۷]، معماری سرویس‌گرا^۳ [۴۸] و الگوهای طراحی شی‌گرا^۴ و رابطه گرا^۵ استفاده می‌شود.

اصول اصلی و اصطلاحات مرتبط با معماری سرویس دامنه‌گرا به شرح زیر است:

- با تعریف Domain به جای شکل‌گیری پیرامون یک به یک میکروسرویس‌ها، پیرامون مجموعه‌ای از میکروسرویس‌های مرتبط شکل گرفته است.
 - معماری Uber همچنین مجموعه دامنه‌هایی را ایجاد می‌کند که لایه نامیده می‌شوند. لایه‌ای که دامنه به آن تعلق دارد مشخص می‌کند که میکروسرویس‌های موجود در آن دامنه چه وابستگی‌هایی را دارند.
 - معماری برای دامنه‌ها رابط‌هایی مشخص ارائه می‌دهد که به عنوان یک نقطه واحد ورود به مجموعه رفتار می‌کنند و به آن‌ها دروازه^۶ گفته می‌شود.
 - هر دامنه باید برای دامنه‌های دیگر ناشناخته باشد، به عبارت دیگر، یک دامنه نباید منطق مربوط به دامنه دیگری را داشته باشد که در داخل کد یا مدل داده‌ای دامنه دیگر وجود دارد. از آنجا که گاهی تیم‌ها نیاز دارند تا منطق را در دامنه تیم دیگری قرار دهند، ما یک معماری تکمیلی برای پشتیبانی از نقاط توسعه یافته تعریف شده در دامنه ارائه می‌دهیم.
- دامنه‌ها به واسطه گردهمایی سرویس‌هایی که منطق تجاری مشابهی دارند، ایجاد می‌شوند و تعداد سرویس‌های درون یک دامنه متغیری است که دامنه‌ی گسترده‌ای دارد و دامنه‌ها می‌توانند شامل یک تا ده‌ها سرویس باشند؛ به عنوان مثال Uber Maps به سه دامنه تقسیم می‌شود که این ۳ دامنه در مجموع ۸۰ میکروسرویس را در بر دارند و ۳ gateway بر سر راه آن‌ها تعبیه شده است.
- معماری Uber برای پاسخ به این سوال که ”چه سرویس‌هایی می‌توانند چه سرویس‌های دیگری را فراخوانی کنند“ دست به ایجاد لایه‌ها در سطح معماری زده است. با وجود لایه‌های مختلف مدیریت وابستگی در مقیاس صورت می‌پذیرد و نگرانی‌های مختلف معماری در لایه‌های مختلف از یکدیگر تفکیک می‌شوند. به کمک لایه‌ها با در نظر گرفتن شعاع انفجار شکست^۷ دامنه‌ها، دامنه‌هایی که در پایین هرم قرار می‌گیرند وابستگی‌های بیشتری

¹Domain-driven Design

²Clean Architecture

³Service-Oriented Architecture

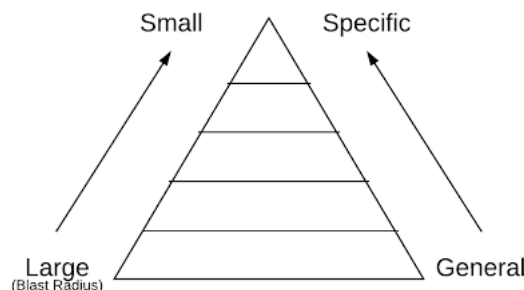
⁴object-oriented design

⁵interface-oriented design

⁶gateway

⁷failure blast radius

داشته و مجموعه‌ی بزرگتری از عملکردهای تجاری را ارائه می‌دهند. شکل ۹-۱ به خوبی ایده‌ی لایه‌بندی را نشان می‌دهد:



شکل ۹-۱: لایه‌ها

لایه‌های پایینی عملکردهایی نظیر مدیریت حساب کاربران را پشتیبانی می‌کنند در حالی که در قله هرم به عملکردهای جزئی‌تر نظیر تجربه کاربری یک امکان در موبایل پرداخته می‌شود. امکانات ممکن است با پخته‌تر شد تعمیم داده شوند و به لایه‌های پایین تر هرم مهاجرت کنند.

شرکت Uber از ۵ لایه‌ی زیر در معماری لایه‌های خود استفاده می‌کند:

- لایه زیرساخت^۱: پاسخ سوالاتی که هر واحد مهندسی می‌تواند از آن استفاده کند نظیر فضای ذخیره‌سازی و شبکه در این لایه داده شده است.
- لایه تجاری^۲: عملکردهای تجاری که سازمان عبور می‌تواند استفاده کند اما منحصرًا مربوط به یک محصول خاص نظیر Uber Ride یا Uber Eat نیستند در این لایه قرار می‌گیرد.
- لایه محصول^۳: عملکردهای تجاری که مربوط به یک محصول و خط کسب‌وکار است توسط این لایه تامین می‌شود اما این لایه نیازهای مختص به یک پلتفرم خاص را پیاده‌سازی نمی‌کند؛ به عنوان به مثال به نحوه پیاده‌سازی "درخواست سفر" در اپلیکیشن موبایل کاری ندارد.
- لایه نمایش^۴: نیازهای عملکردی که کاربران در زمان استفاده از یک برنامه و پلتفرم خاص نیاز دارند را برطرف می‌کند.
- لایه لبه^۵: لایه لبه امکانات Uber را به دنیای بیرون عرضه می‌کنند و اپلیکیشن‌های موبایل نیز از ویژگی‌های

¹Infrastructure Layer

²Business Layer

³Product Layer

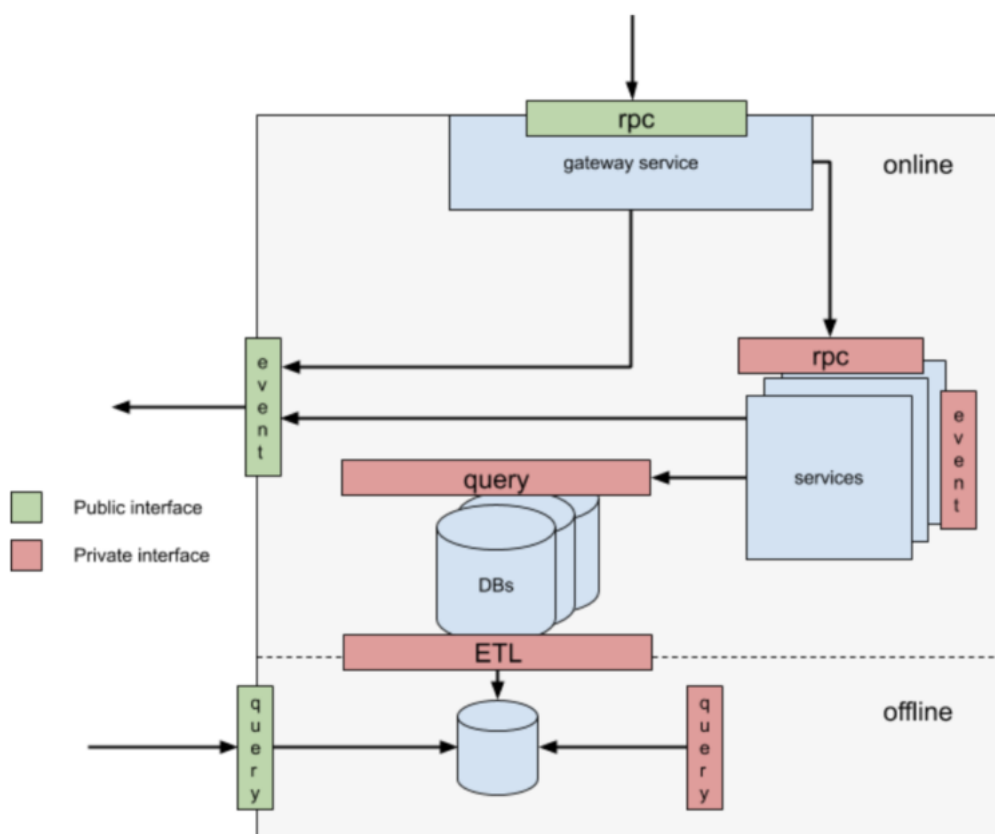
⁴Presentaion layer

⁵Edge Layer

این لایه استفاده می‌کنند.

همان‌طور که مشاهده می‌شود هر چه به لایه‌های بالاتر می‌رویم میزان شعاع انفجار شکست کاهش می‌یابد و عملکرد دامنه‌ها خاص‌تر می‌شود.

دروازه‌های تعریف شده در معماری میکروسرویس دامنه‌گرا بجای اینکه به یک میکروسرویس مربوط باشد به مجموعه‌ای از میکروسرویس‌ها که ما از آن‌ها با نام دامنه یاد می‌کنیم مربوط هستند. شکل ۹-۲ به خوبی ارتباط را نمایش می‌دهد.



شکل ۹-۲: دروازه‌ها

اگر به معنای طراحی شی‌گرا به دروازه‌ها نگاه شود، دروازه‌ها در واقع تعاریف واسط هستند، که به ما امکان می‌دهد هر کاری را که می‌خواهیم از نظر ”پیاده‌سازی” در پایگاه کد انجام دهیم.

۹-۲ قابل مشاهده بودن در مقیاس بزرگ

در هنگام تصور سازمانی به بزرگی Uber یکی از نیازهای مهم قابلیت مشاهده و مانیتورینگ بسیاری از نیازهای کیفی همراه با رشد و گسترش سازمان است. برای حفظ ثبات در معماری میکروسرویس دامنه‌گرا که در بخش

۹-۱ به آن اشاره شد تیم مشاهده^۱ شرکت Uber اقدام به ساخت خط لوله‌هایی به منظور تشخیص، هشدار و کاهش عواقب مشکلات در میکروسرویس‌های ساخته‌شده در تیم‌های مهندسی مختلف کرده‌است؛ به عنوان مثال دو نمونه از این خط لوله‌ها به جهت مراقبت از دیتاست‌ها با نام‌های uMonitor و Neris ساخته شده‌اند. uMonitor سیستم هشدار مبتنی بر معیارها در Uber است که بر اساس پلتفرم M2 [۱۷] مراکز داده را بررسی می‌کند، در حالی که Neris در درجه اول به دنبال هشدارها در زیرساخت‌های سطح میزبان است.

۹-۳ مدیریت منابع در Uber

یکی زیرساخت‌های نرم افزاری رایج در بین شرکتهای فناوری کلاستر^۲ ها هستند. کلاستر مجموعه‌ای از منابع با میزبان‌های فیزیکی متفاوت است که با تبدیل به یک منبع مشترک منطقی در جهت تقویت توان محاسباتی و استفاده انعطاف پذیر از سخت افزار مراکز داده فعالیت می‌کند. در Uber، مدیریت کلاسترها یک لایه انتزاعی برای کارهای مختلف فراهم می‌کند [۲۵].

با افزایش گستره تجارت Uber استفاده کارآمد از منابع کلاسترها بسیار مهم می‌شود. پشته محاسباتی Uber به دلیل وجود کلاسترهای اختصاصی برای موارد استفاده دسته‌ای^۳، با وضعیت^۴ و بی‌وضعیت^۵ به مدیریت ویژه‌ای نیازمند است و همچنین ماهیت دینامیک کسب‌وکار Uber این مساله را سخت‌تر نیز می‌سازد؛ به عنوان مثال دامنه تقاضا برای سفرها با توجه به تعطیلات و ایام هفته بسیار متغیر است. هر چند Uber سعی کرده است تا با فراهم آوردن سخت‌افزار بیش از حد نیاز تا جای ممکن در پاسخ به نیاز مشتریان کاستی نداشته باشد اما گاهی در هنگام اوج بار یک محصول و خط کسب‌وکار، درست در زمانی که بخشی از کلاسترها اضافه بار دارند، کلاستر سایر محصولات ممکن است فضای خالی بدون استفاده داشته باشند و این تکیه بر کلاسترهای اختصاصی نیز به این معنی است که Uber نمی‌تواند منابع محاسبه‌ای را بین آنها به اشتراک بگذارد.

برای استفاده بهتر از منابع، Uber باید این بارهای کاری را در یک سیستم مدیریت بار واحد محاسبه کند. افزایش کارایی‌های حاصل باعث کاهش هزینه فنی هر سفر در زیرساخت‌های Uber می‌شود و باعث افزایش حاشیه سود سفرها در Uber و درنهایت به نفع رانندگان و مسافران خواهد شد.

راه حلی که Uber برای مدیریت کلاسترها ارائه کرده است نرم‌افزار مدیریت یکپارچه کلاسترها به نام Peloton است [۲۵]؛ Peloton یک برنامه‌ریز یکپارچه است که برای مدیریت منابع در بارهای متمایز طراحی شده است و مدیریت کلاسترها در Uber را تجمیع می‌کند. Peloton با استفاده از یک پلتفرم مشترک، توازن

¹ Observability team

² cluster

³ Batch

⁴ Stateful

⁵ Stateless

در استفاده از منابع، به اشتراک گذاری گسترده منابع و پیش‌بینی استفاده از منابع برای نیازهای آینده در Uber را پشتیبانی می‌کند.

از سایر محصولات مشابه با Peloton می‌توان به Google Borg [۵۰]، kubernetes [۱۵]، Hadoop YARN [۹] و Apache Mesos/Aurora [۶][۱۸] اشاره کرد.

۹-۴ منابع ذخیره‌سازی داده در Uber

پیش از سال ۲۰۱۶ شرکت Uber از پایگاه داده Postgres استفاده می‌کرد [۲۶]، اما امروز با استفاده از روشی بدون طرح^۱ [۸] با ترکیب تکنولوژی‌هایی نظیر Riak [۲۸] و cassandra [۷] به همراه MySQL داده‌هایی که برای طولانی مدت نیاز به نگهداری دارند، را ذخیره سازی می‌کند. در طول زمان Postgres و Mysql جای خود را به موتورهای NoSQL دادند و همچنین به جهت دستیابی هر چه بهتر به ویژگی کیفی کارایی در بسیاری از کاربرد ها با افزایش حجم داده‌ها Riak جایگاه خود را به cassandra می‌دهد. مهندسانی که ابزارهای کاربردی و چند منظوره را برای پذیرش در سراسر Uber ایجاد می‌کنند، از Cassandra و Go با شدت بیشتری نسبت به تیم های دیگر در Uber استفاده می‌کنند که دلیل اصلی آن سرعت است. همچنین Uber از Redis [۲۷] هم برای ذخیره سازی داده در حافظه‌های سریع و هم برای صف بندی استفاده می‌کند و با استفاده از Twemproxy [۳۴] مقیاس پذیری لایه حافظه پنهان را بدون از بین بردن نسبت برخورد^۲ حافظه پنهان از طریق الگوریتم هش کردن سازگار فراهم می‌کند.

برای ذخیره سازی توزیع شده و تجزیه و تحلیل داده های پیچیده، از انبار داده Hadoop [۹] به جهت ذخیره بهینه داده ها استفاده می‌شود.

۹-۵ ثبت وقایع در Uber

در Uber، از Apache Kafka [۳] به عنوان یک گذرگاه پیام برای اتصال قسمت های مختلف اکوسیستم استفاده می‌شود. Uber لاگ^۳ مربوط به سیستم و برنامه ها و همچنین داده های رویداد را از برنامه های رانندگان و مسافران را جمع آوری می‌کند و سپس این داده‌ها را از طریق کافکا^۴ در دسترس مصرف کنندگان پایین دست قرار می‌دهد. داده ها در کافکا توسط خطوط لوله real-time و خطوط دسته‌ای^۵ تغذیه می‌شوند. داده های خطوط لوله

^۱Schemaless

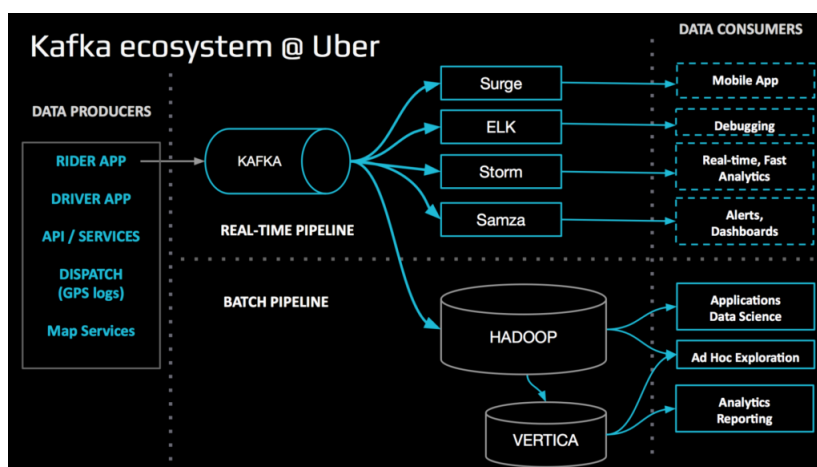
^۲Hit Ratio

^۳Log

^۴Kafka

^۵batch

مربوط به فعالیت هایی مانند محاسبه معیارهای تجاری، رفع باگ^۱، هشدار و داشبورد است. شکل ۹-۳ نمایی از نحوه ارتباط کافکا با سایر اجزای معماری رد Uber را نشان می دهد. [۳]



شکل ۹-۳: کافکا

۹-۶ تست در Uber

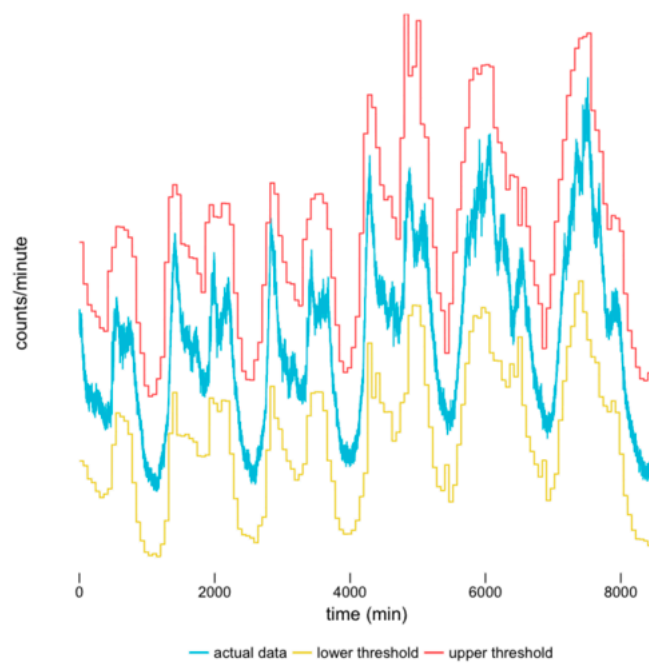
برای اطمینان از اینکه سرویس ها قادر به پاسخگویی به تقاضای محیط واقعی هستند، مهندسين در Uber دو نرم افزار uDestroy و Hailstorm را برای تست میکروسرویس ها ایجاد کرده اند. Hailstorm تست های ادغام را انجام می دهد و زمان اوج بار را نیز شبیه سازی می کند، در حالی که uDestroy عمداً کارها را خراب می کند تا سیستم بتواند در کنترل خرابی های غیر منتظره بهتر عمل کند.

کارمندان Uber قبل از اینکه برنامه به دست کاربران برسند، از نسخه بتا برنامه برای آزمایش مداوم تحولات جدید استفاده می کنند و در طی استفاده از نسخه بتا باگ ها را گزارش و ثبت می کنند. [۳۵]

۹-۷ قابلیت اطمینان و مشاهده پذیری در Uber

شرکت Uber برای نظارت از سامانه ی Nagios [۲۳] استفاده می کند، که به یک سیستم هشدار برای اعلان ها متصل است. در صورتی که کدی در بک اند منجر به شکست سیستم شود به مهندسان مربوطه اعلان ارسال می شود. شرکت Uber به منظور مانیتورینگ شرایط بخش های مختلف سازمان نرم افزار M3 [۱۶] را گسترش داده است به صورتی که هر سخت افزار و کد توسط این نرم افزار تحت نظر قرار دارد. همچنین این شرکت از نرم افزارهای تشخیص ناهنجاری نظیر Argos [۵] برای تشخیص ناهنجاری ها بر اساس مدلی پیش بینی کننده مبنی بر داده های گذشته استفاده می کند که در شکل ۹-۴ نمایی از آن را مشاهده می شود.

^۱Bug



شکل ۹-۴ : Argos

فصل دهم

معماری نمونه مطالعاتی برنامه اشتراک سفر

۱-۱۰ زمینه برنامه اشتراک سفر

هدف ایجاد یک بازار برخط میان رانندگان و مسافران است تا از یک سمت نیاز مسافران برای یافتن رانندگانی که مقاصد و شرایط نزدیک به دلخواه آن برای سفر دارند رفع شود و از سوی دیگر رانندگان قادر باشند تا مسافران باب میل خود را از میان مسافرین انتخاب کنند. در این میان برنامه باید اطلاعات سفرها را لحظه به لحظه رصد کند تا از بروز هر گونه مشکل جلوگیری شود.

معماری پیشنهادی برای چنین زمینه^۱ ای معماری micro-service [۱۹] است تا با شکستن برنامه بر اساس عملکردها^۲ به میکرو سرویس های متناظر، نیازهای کیفی و غیرکیفی را مرتفع سازد. به این صورت با شکستن برنامه به مولفه های تقریباً مستقل بر حسب منطق تجاری به برنامه قابلیت حمل، انعطاف پذیری، رشد و گسترش پذیری بیشتری داده می شود. نمونه معماری مورد نظر برای برنامه اشتراک گذاری خودرو را در شکل ۱-۱۰ نمایش داده شده است [۲۰]. در مورد بحث در دسترس پذیری در معماری میکروسرویس، تاب آوری^۳ مهم ترین فاکتور است؛ حفظ دسترس پذیری با مدیریت و مانیتورینگ صحیح سرویس ها امکان پذیر خواهد بود. گسترش پذیری در

¹Context

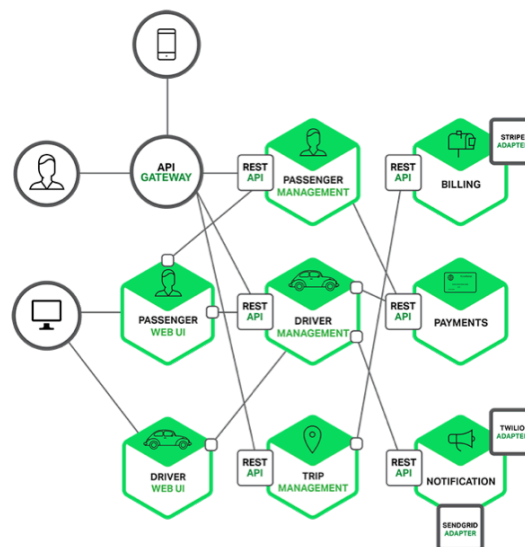
²Functionalities

³Resilience

معماری میکروسرویس در گرو تخصیص بهینه منابع به سرویس ها توسط مولفه توزیع بار^۱ است.

۲-۱۰ معماری پیشنهادی

الگو معماری میکروسرویس ها روشی برای توسعه برنامه در قالب سرویس های کوچک است که هر کدام به صورت تقریباً مستقل فعالیت می کنند. این الگو امکان تحویل/استقرار مداوم^۲ برنامه های پیچیده و بزرگ را فراهم می کند و همچنین به یک سازمان امکان می دهد تا به تدریج پشته فناوری خود را توسعه دهد.



شکل ۱۰-۱: معماری میکروسرویس برنامه اشتراک سفر، عکس از [۲۰]

طبق معماری تعیین شده برنامه های سمت کاربر از طریق API Gateway درخواست های خود را ارسال می کنند و سرویس ها در ارتباط با یکدیگر به درخواست های کاربران پاسخ می دهند.

۳-۱۰ نیازمندی های پوشش داده شده توسط معماری

در ادامه به تجزیه و تحلیل مشخصات معماری رایج برای الگوی معماری میکروسرویس ها و میزان توانایی این الگو در رفع نیازهای کیفی پرداخته می شود.

چابکی و توانایی پاسخ سریع به تغییرات محیطی در کسب و کارهای رقابتی نظیر صنعت حمل و نقل اهمیت بالایی دارد. با توجه به مفهوم واحدهای مستقل و کوچک در الگوی میکروسرویس برنامه با چنین الگویی می تواند با کمترین هزینه تغییرات لازم را در خود اعمال کند و از قابلیت تغییرپذیری^۳ بالایی برخوردار است. به دلیل تفکیک عملکرد و دغدغه های تجاری در برنامه ها با الگوی میکروسرویس، می توان تست را محدود

^۱ Load Balancer

^۲ Continuous Delivery/Development

^۳ Modifiability

کرد، و این امر باعث می شود که آزمایشات هدفمندتر انجام شوند. همچنین آزمون برای یک سرویس خاص بسیار آسان تر و عملی تر از نوشتن آزمون برای یک معماری Monolithic است و چون در معماری میکروسرویس مولفه ها loosely coupled هستند امکان این که یک تغییر در یک سرویس منجر به شکست سایر سرویس ها شود، کم است.

به دلیل ذات توزیع شده و نیاز به پیام رسانی تحت شبکه در الگوی میکروسرویس نوشتن سرویس ها با کارایی بالا در خروجی مناسب بسیار اهمیت دارد و یکی از چالش های استفاده از الگوی میکروسرویس حفظ کارایی در هنگام گسترش برنامه است اما در مقابل چالش کارایی گسترش پذیری برنامه تحت این الگو به راحتی صورت می پذیرد و با ساختار خود پیچیدگی مسائل رو با شکستن به ریزسرویس ها کاهش می دهد. همچنین وجود امکان ارتباط راحت با سرویس های دیگر نیز به قابلیت همکاری در این معماری نیز کمک می کند.

پویایی معماری میکرو سرویس این امکان را می دهد که بتوان به چالش های امنیت نیز پاسخ داد، با استفاده از این معماری می توان تمامی ارتباطات را رمز کرد، تمامی درخواست ها را احراز هویت کرد و به چالش های رمزگذاری و دنبال کردن نشست کاربران و جلوگیری از حملات منع سرویس با استفاده از فیلترها نیز پاسخ داد [۱۱].

از آنجا که در این معماری می توان رنج وسیعی از API ها را برنامه ریزی کرد بنابراین در صورت طراحی مناسب API ها می توان به نیازمندی هایی که موجب افزایش قابلیت استفاده کاربر می شود نیز پاسخ داد.

فصل یازدهم

معماری نمونه مطالعاتی برنامه پارکینگ

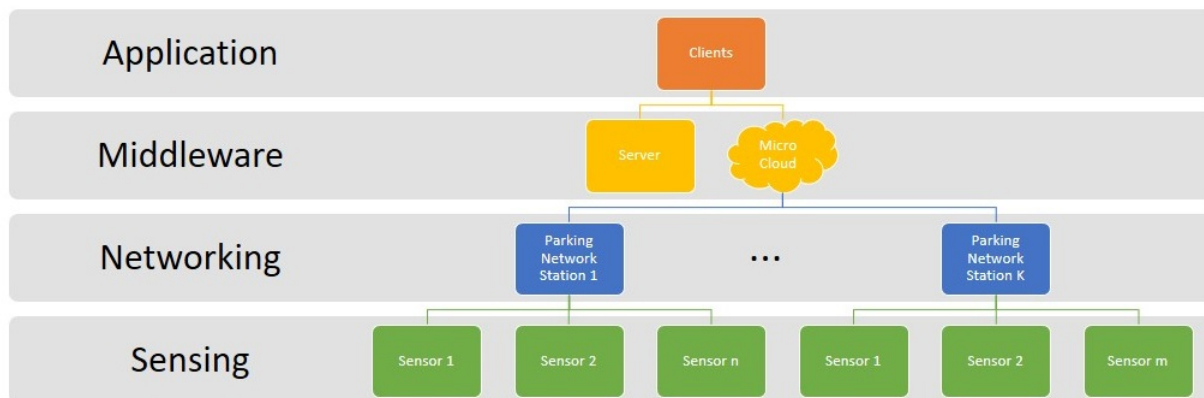
۱-۱۱ زمینه برنامه پارکینگ

هدف از برنامه برقراری ارتباط کاربران با پارکینگ‌های مختلف و فضاهای خالی در هر پارکینگ است؛ کاربران در این برنامه ها باید بتوانند از مکان‌های خالی در پارکینگ‌های سطح شهر آگاه شوند و امکان رزرو مکان‌های خالی را داشته باشند. در این برنامه تعداد زیاد پارکینگ‌ها در سطح شهر و تغییرات زیاد در وضعیت پارکینگ‌ها سبب پیچیدگی مدیریت سنسورها می‌شود، در حالی که این پیچیدگی نباید خود را در پایگاه کد و سایر مولفه‌های سیستم نشان دهد. به منظور تفکیک دغدغه‌ها، معماری پیشنهادی برای این برنامه معماری لایه‌ای است.

در معماری لایه‌ای، لایه‌ها مستقل هستند بنابراین گروهی از تغییرات در یک لایه بر لایه‌های دیگر تأثیر نمی‌گذارد. با توجه به این ویژگی هر گونه تغییر در ساختار هر لایه این امکان را به ما می‌دهد تا مستقل از سایر لایه‌ها به بررسی تغییرات انجام شده بپردازیم. به عنوان مثال تغییر در پروتکل مورد استفاده در سنسورها در معماری لایه‌ای صدمه‌ای به ساختار لایه‌های بالاتر نمی‌زند.

۲-۱۱ معماری پیشنهادی

با توجه به زمینه مطرح شده برای برنامه پارکینگ‌های شهری، معماری لایه ای پیشنهادی باید دارای لایه‌های شکل ۱-۱۱ باشد. این شکل با توجه به مقاله [۴۲] کشیده شده است. در معماری لایه‌ای پیشنهادی شده حیطه



شکل ۱-۱۱: معماری لایه‌ای پیشنهادی برای برنامه پارکینگ‌های شهری

عملکردی هر لایه به شرح زیر است:

- لایه‌ی برنامه^۱: برنامه‌های سمت کاربر شامل برنامه‌ی تلفن همراه و نسخه‌های وب در این لایه پیاده سازی می‌شوند و از طریق رابط‌ها از لایه‌ی پایینی خود خدمات را دریافت می‌کنند. کاربران قادر به جستجو در مکان‌های مورد نظر خود برای پارکینگ هستند و می‌توانند فضای مورد نظر خود را رزرو کنند.
- لایه میان‌افزار^۲: در این لایه اطلاعات پارکینگ‌ها به صورت لحظه‌ای دریافت می‌شود و پس از پردازش خدماتی که کاربران در لایه‌ی برنامه نیاز دارند، در اختیارشان قرار داده می‌شود. خود سرور در این لایه از معماری میکرو سرویس تبعیت می‌کند که پیشتر ویژگی‌های آن مطرح شد.
- لایه شبکه^۳: لایه‌ی شبکه به منظور ارتباط آسان تر سنسورها با لایه‌ی میان افزار تعبیه شده است. اگر قرار بود هر یک از سنسورها خود مستقلاً به لایه‌ی میان‌افزار وصل شوند پیچیدگی شبکه و مدیریت آن بسیار دشوار بود. لایه‌ی شبکه با تجمع اطلاعات سنسورها ضمن کشف خرابی یا اطلاعات ناصحیح در میان سنسورها با پروتکل‌های مطمئن تری ضمن هزینه کمتر اطلاعات را در اختیار لایه میان‌افزار قرار می‌دهد. لایه شبکه ارتباط بی نقص بین مراکز مختلف پارکینگ و سیستم میان‌افزار و در نهایت کاربران را تضمین می‌کند. داده‌های کاربر و مرکز پارک از طریق این لایه به سیستم میان‌افزار منتقل می‌شود. این لایه شامل

¹Application

²Middleware

³Networking

انواع مختلف فن آوری های ارتباطی نظیر شبکه LAN و WAN است.

- لایه سنسورها^۱ : سنسورها از نوع های مختلف در این لایه قرار دارند و این لایه با محدودسازی پیچیدگی های سخت افزاری، مانع از انتشار آن به لایه های بالاتر خواهد شد.

۱۱-۳ نیازمندی های پوشش داده شده توسط معماری

در معماری لایه ای از آنجا که هر لایه مستقل از دیگری است موجب اصلاح پذیری بالا و قابلیت آزمون در این سیستم می شود. چرا که می توان برای مثال در لایه سنسورها، پروتکل آن ها و دیگر موارد را تغییر داد یا هر لایه را مجزا آزمون کرد. همچنین در این معماری می توان امنیت را بر روی ارتباطات بین کاربران تا سرور، یا سنسورها تا میکروکلاد به راحتی داشت.

از آنجا که خود سرور از معماری میکرو کرنل استفاده می کند نیز خصوصیات معماری میکرو کرنل نیز به آن نیز اضافه می شود.

به راحتی می توان نیازمندی های مختلف را با API ها برآورده کرد که به پوشش نیازمندی های احراز هویت و پیاده سازی قابلیت استفاده پاسخ می دهد. آزمون راحت سرویس ها و تغییر آن ها نیز به دلیل ویژگی loosely coupled مولفه ها نیز امکان پذیر است.

همچنین بدیل ذات توزیع شده معماری میکرو کرنل و مستقل بودن معماری لایه گسترش پذیری نیز امکان پذیر است که از نتایج این گسترش پذیری می توان به افزایش کارایی و دردسترس پذیری این سرویس نیز اشاره داشت. وجود امکان ارتباط راحت با سرویس های دیگر نیز به قابلیت همکاری در این معماری نیز کمک می کند.

^۱ Sensing

References

- [1] 24/7 access to parkwhiz. https://help.parkwhiz.com/en_us/247-access-HJag1lgZWPW. Accessed: 2021-06-02.
- [2] 6 advantages and disadvantages of carpooling. <https://connectusfund.org/6-advantages-and-disadvantages-of-carpooling>. Accessed: 2021-06-02.
- [3] Apache kafka. <http://kafka.apache.org/>. Accessed: 2021-06-02.
- [4] Apache kafka at uber. <https://eng.uber.com/kafka/>. Accessed: 2021-06-02.
- [5] Argos. <https://eng.uber.com/argos-real-time-alerts/>. Accessed: 2021-06-02.
- [6] aurora. <http://aurora.apache.org/>. Accessed: 2021-06-02.
- [7] cassandra. <https://cassandra.apache.org/>. Accessed: 2021-06-02.
- [8] Designing schemaless, uber engineering's scalable datastore using mysql. <https://eng.uber.com/schemaless-part-one-mysql-datastore/>. Accessed: 2021-06-02.
- [9] hadoop. <https://hadoop.apache.org/docs/r2.7.1/hadoop-yarn/hadoop-yarn-site/YARN.html>. Accessed: 2021-06-02.
- [10] How do we measure the costs of software testing. <https://www.softwaretestinggenius.com/how-do-we-measure-the-costs-of-software-testing/>. Accessed: 2021-06-02.
- [11] How to secure microservices architecture. <https://securityintelligence.com/posts/how-to-secure-microservices-architecture/>. Accessed: 2021-06-02.
- [12] Integrating waze with arcgis. <https://working-with-waze.hub.arcgis.com/>. Accessed: 2021-06-02.
- [13] Interoperability and the digital payment ecosystem. <https://valid.com/digital-payment-ecosystem>. Accessed: 2021-06-02.
- [14] Introducing domain-oriented microservice architecture. <https://eng.uber.com/microservice-architecture>. Accessed: 2021-06-02.
- [15] kubernetes. <https://kubernetes.io/>. Accessed: 2021-06-02.

- [16] M3. <https://www.youtube.com/watch?v=89H48IwFeV4>. Accessed: 2021-06-02.
- [17] M3: Uber's open source, large-scale metrics platform for prometheus. <https://eng.uber.com/m3/>. Accessed: 2021-06-02.
- [18] mesos. <http://mesos.apache.org/>. Accessed: 2021-06-02.
- [19] Microservice architecture. <https://microservices.io/>. Accessed: 2021-06-02.
- [20] Microservices: an agile architecture for carpooling application. <https://donetechno.com/en/microservices-an-agile-architecture-part-1/>. Accessed: 2021-06-02.
- [21] Modular agile: Loosely coupled, highly cohesive ceremonies. <https://dzone.com/articles/modular-agile-loosely-coupled>. Accessed: 2021-06-02.
- [22] Myway: Requirement specification and analysis. <https://www.sonarsource.com/why-us/code-quality/>. Accessed: 2021-06-02.
- [23] Nagios. <https://www.nagios.org/>. Accessed: 2021-06-02.
- [24] Parkwhiz: A design review. <https://medium.com/@corresrachel/parkwhiz-a-design-review-98c15033af36>. Accessed: 2021-06-02.
- [25] Peloton: Uber's unified resource scheduler for diverse cluster workloads. <https://eng.uber.com/resource-scheduler-cluster-management-peloton/>. Accessed: 2021-06-02.
- [26] Project mezzanine: The great migration. <https://eng.uber.com/mezzanine-codebase-data-migration/>. Accessed: 2021-06-02.
- [27] redis. <https://redis.io/>. Accessed: 2021-06-02.
- [28] Riak. <https://riak.com/>. Accessed: 2021-06-02.
- [29] Sonarqube, code quality. <https://www.sonarsource.com/why-us/code-quality/>. Accessed: 2021-06-02.
- [30] Starnavpilot, navigation software. https://www.aragon-technologies.com/en/starnavpilot/interoperability_with_google_maps.php. Accessed: 2021-06-02.
- [31] Support terms of parkwhiz. <https://www.parkwhiz.com/support/terms/>. Accessed: 2021-06-02.
- [32] The technological race to find you a place to park. <https://www.nytimes.com/2017/11/30/business/car-parking-apps.html>. Accessed: 2021-06-02.
- [33] Three tenets of information security. <https://www.lbmc.com/blog/three-tenets-of-information-security/>. Accessed: 2021-06-02.
- [34] Twemproxy. <https://github.com/twitter/twemproxy>. Accessed: 2021-06-02.
- [35] The uber engineering tech stack, part i: The foundation. <https://eng.uber.com/tech-stack-part-one-foundation/>. Accessed: 2021-06-02.
- [36] Uber usability analysis. <https://medium.com/@monikaadarsh/does-uber-app-have-an-uber-user-experience-e054189e60e5>. Accessed: 2021-06-02.
- [37] Uber's real-time push platform. <https://eng.uber.com/real-time-push-platform/>. Accessed: 2021-06-02.
- [38] Under the hood of uber's experimentation platform. <https://eng.uber.com/xp/>. Accessed: 2021-06-02.
- [39] Using waze and scoop apps for carpool commuting. <https://cubiclettherapy.com/waze-scoop-carpool/>. Accessed: 2021-06-02.
- [40] Waze carpool. <https://www.waze.com/carpool/>. Accessed: 2021-06-02.
- [41] Wikipedia: Availability. <https://en.wikipedia.org/wiki/Availability>. Accessed: 2021-06-02.

- [42] AHMED, S., RAHMAN, M. S., RAHAMAN, M. S., ET AL. A blockchain-based architecture for integrated smart parking systems. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (2019), IEEE, pp. 177–182.
- [43] ATAIE, E., BABAEIAN, M., AND AGHAEI, F. Improving software quality by pattern driven software architecture. In *5th Symposium on Advance in Science & Technologies* (2011).
- [44] BASS, L., CLEMENTS, P., AND KAZMAN, R. *Software architecture in practice*. Addison-Wesley Professional, 2003.
- [45] CIUBĂNCAN, D., AND ȚIRBAN, P. Empirical validation of oo metrics impact on changeability. *Studia Universitatis Babes-Bolyai, Informatica* 65, 1 (2020).
- [46] EVANS, E., AND EVANS, E. J. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.
- [47] MARTIN, R. C. *Clean architecture: a craftsman's guide to software structure and design*. Prentice Hall, 2018.
- [48] PERREY, R., AND LYCETT, M. Service-oriented architecture. In *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on* (2003), IEEE, pp. 116–119.
- [49] SUDMANN, M., TIEDE, D., LANG, S., AND BARALDI, A. Semantic and syntactic interoperability in online processing of big earth observation data. *International journal of digital earth* 11, 1 (2018), 95–112.
- [50] VERMA, A., PEDROSA, L., KORUPOLU, M., OPPENHEIMER, D., TUNE, E., AND WILKES, J. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems* (2015), pp. 1–17.