

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ



دانشگاه شهرستان

دانشکده مهندسی برق و کامپیوتر

پایان نامه کارشناسی مهندسی کامپیوتر-نرم افزار

تحلیل فنی پرداخت مبتنی بر کیف پول الکترونیکی و
طراحی و پیاده سازی نمونه ای از پرداخت

نگارش:

محمد حسام مدبری

استاد راهنما:

دکتر مرتضی دری گیو

بهمن-۱۳۹۵

این جانب (این جانب)..... بدین وسیله انقدر می دارم که محتوای علمی این نوشتار با عنوان

که به عنوان پیمان نامه کارشناسی مهندسی برق گرایش به دلگذره برق و کامپیوتر - دانشگاه سمنان ارزان شده،

وارایی اصالت پژوهشی بوده و حاصل فایل علمی این جانب (این جانب) است.

این جانب (این جانب) می دانم که اگر خلاف ادعای بالا در عرضه مجز شود، کیفر حقوق مسترد برای این جانب (این جانب) سلب شده و مرتب قانونی مرطبه آن نیز از طرف مراجع ذمی بربط قائل بکسری است.

نام و نام خانوادگی - شماره و اشخاصی

کارخانه و اصناف

کارخانه و اصناف

این فرم به صورت دستنویس تکمیل می شود.



دانشگاه شهرستان

دانشکده‌ی مهندسی برق و کامپیوتر

تأیید دفاع از پایان‌نامه‌ی کارشناسی
پایان‌نامه‌ی محمد حسام مدبری

برای اخذ درجه‌ی کارشناسی مهندسی کامپیوتر-نرم‌افزار

با عنوان:

تحلیل فنی پرداخت مبتنی بر کیف پول الکترونیکی و
پیاده‌سازی نمونه‌ای از پرداخت

در تاریخ ۱۸ بهمن ۱۳۹۵ دفاع شد و مورد تأیید قرار گرفت.

تأییدکنندگان:

- | | |
|------|---------------------------------------------------------|
| امضا | (۱) استاد محترم داور جناب آقای مهندس ابوالمعالی |
| امضا | (۲) استاد محترم راهنمای جناب آقای دکتر دری گیو |
| امضا | (۳) مدیر محترم گروه کامپیوتر جناب آقای دکتر فدایی اسلام |

چکیده

با توجه به مشکلات امنیتی موجود در زمینه‌ی پرداخت الکترونیکی، استفاده از راه حل کیف پول الکترونیکی به منظور رفع این مشکلات حائز اهمیت است. به سبب اینکه پرداخت‌های امروزی، اطلاعات کارت بانکی را در معرض خطر قرار می‌دهند، پنهان بودن اطلاعات پرداخت و امن‌تر کردن تراکنش‌ها به علاوه‌ی سهولت پرداخت، از هداف اصلی کیف پول الکترونیکی می‌باشد.

در فصل‌های ابتدایی تکنولوژی Tokenisation مورد مطالعه قرار خواهد گرفت که نقش اصلی در ایمن‌سازی کیف‌پول‌های الکترونیکی را دارد. در باقی فصل‌ها به بررسی بخش‌های مختلف پروژه‌ی پیاده‌سازی شده پرداخته می‌شود.

پروژه‌ی انجام شده، پیاده‌سازی مدلی از پرداخت است که با کمک تکنولوژی iBeacon به منظور پیدا کردن موقعیت کاربر مورد استفاده قرار گرفته است.

کلید واژگان: iBeacon ، پرداخت الکترونیک ، کیف‌پول الکترونیکی

فهرست مطالب

۱	۱	مقدمه
۲	۱.۱	معرفی سرویس‌های برجسته‌ی کیف پول الکترونیکی
۲	۱.۱.۱	سرویس Apple Pay
۲	۲.۱.۱	روش‌های پرداخت سرویس Apple Pay
۳	۳.۱.۱	مراحل کار با سرویس Apple Pay
۳	۱.۳.۱.۱	پرداخت درون فروشگاه
۴	۲.۳.۱.۱	پرداخت درون نرم‌افزار
۵	۳.۳.۱.۱	پرداخت درون وب سایت
۵	۲.۱	نرم‌افزارهای برجسته‌ی پرداخت موبایل
۵	۱.۲.۱	Starbucks
۶	۲.۲.۱	Royal Bank of Canada (RBC)
۶	۳.۲.۱	ANZ Bank
۷	۴.۲.۱	CaixaBank
۸	۵.۲.۱	Capital One
۹	۶.۲.۱	SnapScan
۹	۳.۱	سرویس Amazon Go
۹	۴.۱	معرفی نرم‌افزار پیاده‌سازی شده
۱۰	۱.۴.۱	تکنولوژی‌های استفاده شده
۱۰	۲.۴.۱	نحوه‌ی کار نرم‌افزار

۲ تکنولوژی Tokenisation و زیرساخت‌ها

۱۷	تعریف اولیه	۱.۲
۱۷	بانک صادر کننده	۱.۱.۲
۱۷	پذیرنده	۲.۱.۲
۱۸	بانک پذیرنده	۳.۱.۲
۱۸	دارنده‌ی کارت	۴.۱.۲
۱۸	خدمات پرداخت	۵.۱.۲
۱۸	ابزار پرداخت	۶.۱.۲
۱۸	کارت بانکی	۷.۱.۲
۱۸	شماره‌ی کارت (PAN)	۸.۱.۲
۱۹	شبکه‌ی پرداخت	۹.۱.۲
۱۹	پردازنده‌ی پرداخت	۱۰.۱.۲
۱۹	تاریخچه تکنولوژی Tokenisation	۲.۲
۱۹	مقدمه‌ای بر تکنولوژی Tokenisation	۳.۲
۲۰	خلاصه‌ی کلی	۱.۳.۲
۲۱	اکوسیستم Tokenisation	۴.۲
۲۴	Token Service Provider	۱.۴.۲
۲۴	Token Vault	۱.۱.۴.۲
۲۴	درخواست دهنده‌ی Token	۲.۱.۴.۲
۲۵	Secure Element	۵.۲
۲۶	Host Card Emulation	۶.۲
۲۷	پیدایش تکنولوژی HCE	۱.۶.۲
۲۷	فواید استفاده از تکنولوژی HCE	۲.۶.۲
۲۷	استقلال	۱.۲.۶.۲
۲۷	یکی شدن آسان‌تر با شخص ثالث	۲.۲.۶.۲
۲۷	هزینه‌های پایین‌تر	۳.۲.۶.۲
۲۸	استفاده از چندین کارت	۴.۲.۶.۲

۲۹	Android Pay و Apple Pay	بررسی نحوه کار سرویس‌های	۳
۲۹	Apple Pay	۱.۳
۳۰	اجزای	۱.۱.۳
۳۰	Secure Element	۱.۱.۱.۳
۳۰	NFC Controller	۲.۱.۱.۳
۳۰	Wallet	۳.۱.۱.۳
۳۰	Secure Enclave	۴.۱.۱.۳
۳۱	سرورهای	۵.۱.۱.۳
۳۱	چگونگی استفاده از Apple Pay	۲.۱.۳
۳۱	چگونگی استفاده از NFC Controller	۳.۱.۳
۳۲	نحوه اضافه شدن کارت‌های اعتباری و بدهی	۴.۱.۳
۳۳	اضافه کردن کارت‌ها بصورت دستی	۱.۴.۱.۳
۳۴	اجازه‌ی پرداخت	۵.۱.۳
۳۴	کد امنیتی پویای مخصوص هر تراکنش	۶.۱.۳
۳۵	پرداخت از راه NFC	۷.۱.۳
۳۵	پرداخت درون نرم‌افزاری	۸.۱.۳
۳۶	Android Pay	۲.۳
۳۸	طراحی و پیاده‌سازی نرم‌افزار پرداخت سمت سرور	۴	
۳۸	ساختار کلی	۱.۴
۳۹	Web Service	۲.۴
۴۱	تشریح نحوه پیاده‌سازی	۳.۴
۴۱	بسته Controllers	۱.۳.۴
۴۱	کلاس WhereIsBeacon	۱.۱.۳.۴
۴۵	Hibernate	۴.۴
۴۶	اجزای برنامه‌نویسی	۱.۴.۴
۴۶	شیء Configuration	۱.۱.۴.۴

۴۶	SessionFactory	شیء	۲.۱.۴.۴
۴۶	Session	شیء	۳.۱.۴.۴
۴۶	Transaction	شیء	۴.۱.۴.۴
۴۷	Query	شیء	۵.۱.۴.۴
۴۷	Criteria	شیء	۶.۱.۴.۴
۴۷	hibernate.cfg.xml		۲.۴.۴
۴۸	POJO	کلاس	۳.۴.۴
۵۰	Mapping configuration	فایل	۴.۴.۴
۵۱	نحوه‌ی استفاده	نحوه‌ی استفاده	۵.۴.۴

۵ طراحی و پیاده‌سازی نرم‌افزار پرداخت سمت کالاینت

۵۳	Beacon	۱.۵
۵۳	ساختار داخلی Beacon	۱.۱.۵
۵۴	موارد استفاده	۲.۱.۵
۵۴	پروتکل‌ها	۳.۱.۵
۵۴	iBeacon	۱.۳.۱.۵
۵۵	Eddystone	۲.۳.۱.۵
۵۵	iBeacon	۴.۱.۵
۵۶	دستگاه‌های دارای تکنولوژی iBeacon	۵.۱.۵
۵۶	سیگنال‌های iBeacon	۶.۱.۵
۵۷	رابطه‌ای نرم‌افزاری	۷.۱.۵
۵۷	حریم خصوصی	۱.۷.۱.۵
۵۷	iBeacon	۲.۷.۱.۵
۵۷	دقت	۲.۷.۱.۵
۵۷	Monitoring	۸.۱.۵
۵۸	Ranging	۹.۱.۵
۵۹	محدودیت‌های فیزیکی	۱۰.۱.۵
۵۹	برنامه‌نویسی در iOS	۲.۵

۵۹	اجزای اصلی	۳.۵
۵۹	AppDelegate.swift	۱.۳.۵
۶۰	@UIApplicationMain	۲.۳.۵
۶۱	Storyboard	۳.۳.۵
۶۱	الگوی طراحی MVC	۴.۵
۶۲	UIView	۱.۴.۵
۶۲	UIViewController	۲.۴.۵
۶۳	استفاده از رابط برنامه‌نویسی نرم‌افزار Core Location	۵.۵
۶۴	Monitoring	۱.۵.۵
۶۶	Ranging	۲.۵.۵
۷۰	UserDefaults	۶.۵
۷۱	Model	۷.۵

فصل ۱

مقدمه

امروزه اکثر خریدها و به موجب آن پرداخت‌های بانکی صورت می‌گیرد، به همین خاطر توجه سارقان و کلاهبرداران این حوزه را به خود جلب کرده است. سرویس‌های کیف‌پول الکترونیکی این اجازه را به کاربران می‌دهند که کارت‌های بانکی خود را بصورت دیجیتالی در دستگاه‌های دیجیتالی مانند تلفن‌های هوشمند ذخیره کنند. بدین طریق کاربران در زمان پرداخت به جای استفاده از کارت‌های بانکی حقیقی از کارت دیجیتالی ذخیره شده بر روی دستگاه دیجیتالی‌شان استفاده می‌کنند. تدبیر امنیتی مناسبی برای ایمن کردن اینگونه پرداخت‌ها دیده شده که مهم‌ترین آن‌ها تکنولوژی Tokenisation می‌باشد. در فصل ۲ به توضیح این تکنولوژی پرداخته خواهد شد.

برای خرید یک کالا و یا یک سرویس مدل‌های مختلفی وجود دارد که کاربران می‌توانند با استفاده از آن‌ها پرداخت را انجام دهند. برای مثال کاربران می‌توانند خریدهایشان را از راه ترمینال‌های مجهز به تکنولوژی NFC^۱ به صورت پرداخت غیرتماسی^۲، و یا از راه نرم‌افزارهای پرداخت به راههای مختلف انجام دهند. در ادامه‌ی این فصل به چند نمونه از مدل‌های پرداخت که در حال حاضر موجود هستند، خواهیم پرداخت.

لازم به ذکر است، کیف‌پول الکترونیکی که به منظور نگهداری بیت کوین‌ها^۳ و یا پول الکترونیکی می‌باشد، مورد بحث نخواهد بود.

^۱Near Field Communication

^۲Contactless payment

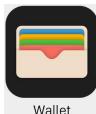
^۳Bitcoin

۱.۱ معرفی سرویس‌های برجسته‌ی کیف پول الکترونیکی

از جمله‌ی سرویس‌های کیف‌پول، می‌توان Apple Pay را نام برد که اپل در ماه سپتامبر ۲۰۱۴ با معرفی iPhone 6 آن را عرضه کرد. شرکت گوگل نیز سرویس Android Pay را در ماه فوریه‌ی ۲۰۱۵ معرفی کرد. سرویس‌های کیف‌پولی که این شرکت‌ها ارائه کرده‌اند، بر پایه‌ی تکنولوژی Tokenisation می‌باشند، و از این تکنولوژی برای مدیریت کارت‌های بانکی استفاده می‌کنند. لازم به ذکر است که سرویس Apple Pay به عنوان امن‌ترین روش پرداخت موجود در جهان شناخته شده است [۱]. بنابراین در ادامه تمرکز بیشتر بر روی توضیح سرویس Apple Pay خواهد بود، از لحاظ رابط کاربری تفاوت چشم‌گیری با یکدیگر ندارند، اما از لحاظ فنی جای بحث خواهد داشت. در فصل ۳ به مقایسه‌ی دو سرویس ارائه شده توسط این دو شرکت و علت برتری سرویس Apple Pay پرداخته خواهد شد.

۱.۱.۱ سرویس Apple Pay

برای استفاده از سرویس Apple Pay از نرم‌افزار Wallet که بصورت پیش‌فرض بر روی دستگاه‌های iPhone 6 و مدل‌های بالاتر و Apple Watch نصب شده است، استفاده می‌شود. شکل ۱.۱ آیکون نرم‌افزار Wallet را نشان می‌دهد.



شکل ۱.۱: آیکون نرم‌افزار Wallet

۲.۱.۱ روش‌های پرداخت سرویس Apple Pay

این سرویس برای پرداخت سه راه را به کاربران ارائه می‌دهد.

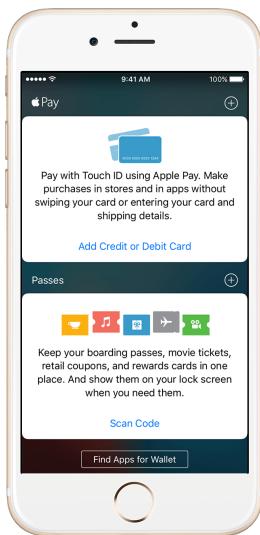
- پرداخت درون فروشگاه

- پرداخت درون نرم‌افزار

- پرداخت درون وبسایت

۳.۱.۱ مراحل کار با سرویس Apple Pay

برای استفاده از این سرویس در ابتدا باید کارت بانکی را که به منظور انجام خریدها استفاده می‌شود، تعریف شود. لازم به ذکر است که کارت مورد نظر باید جزء کارت‌های پشتیبانی شده توسط Apple Pay باشد. در شکل ۲.۱ محیط نرم‌افزار برای اضافه کردن نشان داده شده است.



شکل ۲.۱: محیط نرم‌افزار Apple Pay

با کلیک نمودن بر روی آیکون «+» در بالای صفحه، وارد مراحل اضافه کردن کارت بانکی خواهید شد. پس از احراز هویت و تایید آن توسط بانک، Apple Pay برای انجام خریدها آماده است. در ادامه مراحل پرداخت برای سه روش ارائه شده توضیح داده خواهد شد. پرداخت‌های فوق هم از طریق گوشی‌های هوشمند iPhone و هم از طریق Apple Watch امکان‌پذیر است.

۳.۱.۱ پرداخت درون فروشگاه

کاربر پس از اتمام خرید در فروشگاه، می‌تواند از این روش پرداخت برای تکمیل خرید خود اقدام کند. پس از ارائه‌ی صورت حساب توسط صندوق دار، کاربر گوشی و یا Apple Watch خود را نزدیک به میدان NFC ترمینال می‌کند، و سپس اجازه‌ی پرداخت را با استفاده از اثر انگشت خود و یا کلمه‌ی عبور می‌دهد. بر روی Apple Watch کاربر لبه‌ی انتهایی آن را دو بار لمس می‌کند. ترمینال‌های مجاز برای استفاده از این روش پرداخت دارای نشان شکل ۳.۱ می‌باشند.

ترمینال‌های مورد پشتیبانی همانند شکل ۴.۱ می‌باشند، که دارای نماد Apple Pay هستند.



شکل ۳.۱: نماد NFC Apple Pay

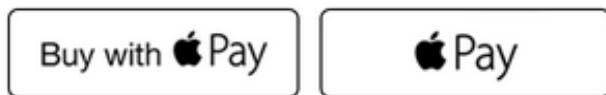


شکل ۴.۱: ترمینال پشتیبانی کننده از Apple Pay

۲.۳.۱.۱ پرداخت درون نرم افزار

کاربر با استفاده از iPhone، iPad و Watch Apple Pay را به عنوان روش پرداخت برای پرداختهای درون نرم افزار استفاده کند. برای انجام پرداخت از این راه مراحل زیر لازم است:

۱. لمس دکمه Apple Pay و یا لمس Buy with Apple Pay به عنوان روش پرداخت در زمان بازبینی نهایی خرید (شکل ۵.۱).



شکل ۵.۱: آیکون های Apple Pay

۲. بازبینی اطلاعات صورت حساب، اطلاعات تماس و نحوه ارسال خرید، برای حصول اطمینان از صحت اطلاعات. اگر کاربر مایل به پرداخت با کارت متفاوتی غیر از کارت پیش فرض باشد، باید علامت فلش لمس شود.

۳. اگر نیاز به وارد کردن اطلاعات تماس، صورت حساب و یا نحوه ارسال محموله باشد، کاربر اطلاعات

را وارد می‌کند. Apple Pay این اطلاعات را ذخیره می‌کند تا نیازی به وارد کردن دوباره‌ی آن‌ها نباشد.

۴. بر روی iPad و iPhone کاربر با استفاده از اثر انگشت اجازه‌ی پرداخت را می‌دهد. بر روی Apple Watch کاربر لبھی انتهایی آن را دو بار لمس می‌کند.

۳.۲.۱.۱ پرداخت درون وب سایت

اپل پشتیبانی از پرداخت درون وب سایت خود را به تازگی در پاییز ۲۰۱۵ به همراه مک بوک جدیدش معرفی کرد. پرداخت از این راه فقط از طریق مرورگر Safari قابل انجام است. مراحل انجام پرداخت همانند پرداخت درون نرم افزار می‌باشد. به جز اینکه در مک بوک جدید اپل قابلیت خواندن اثر انگشت اضافه شده. در مک بوک‌های سری قبل کاربر مجبور به اتصال گوشی و مک از راه بلوتوث، و انجام پرداخت از راه گوشی بود.

۲.۱ نرم افزارهای بر جسته‌ی پرداخت موبایل

بانک‌ها در سراسر جهان در حال عرضه‌ی پلتفرم‌های جدید بوده و شرکت‌های فعال در این حوزه در حال انتشار در بازارهای کشورها هستند. مصرف کنندگان نیز اشتیاق استفاده از این محصولات را نشان داده‌اند. به نقل از Juniper Research، پیش‌بینی شده است که پرداخت‌های صورت گرفته از راه موبایل و پرداخت بدون تماس در طی سال ۲۰۱۶ بالغ بر ۳.۶ تریلیون دلار خواهد بود. بنابراین آینده‌ی پرداخت موبایل بسیار امیدبخش و خوب بنظر می‌آید [۲].

با چنین انتظار بالایی، در ادامه لیستی از نرم افزارهای ابداعی پرداخت موبایل در بازار کنونی آورده شده است. بعلاوه ویژگی بر جسته‌ی هر نرم افزار که آن را از بقیه‌ی نرم افزارها متمایز کرده است، بیان شده است.

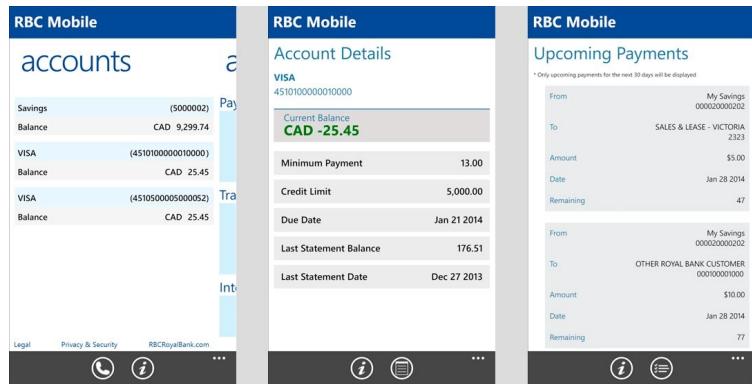
Starbucks ۱.۲.۱

ویژگی: طرح‌های وفاداری، سرویس پرداخت از پیش

هتمیشه نسبت به رقبا در این عرصه برتری داشته است. از سال ۲۰۰۱ جزء اولین‌ها در ارائه‌ی WiFi درون فروشگاه بود و از سال ۲۰۰۹ جزء اولین پذیرنده‌گان پرداخت موبایل بوده است. شمار تراکنش‌های موبایلی در فروش فروشگاه‌ها ۲۵ درصد کل ترانکش‌ها می‌باشد که با استفاده از آسان تر کردن راههای پرداخت برای مصرف کنندگان و ارائه‌ی سرویس سفارش از قبل، پرداخت‌ها و دریافت طرح‌های وفاداری و جایزه را آسان‌تر کرده است.

Royal Bank of Canada (RBC) ۲.۱

ویژگی: کارت‌های هدیه، رسیدهای دیجیتالی بزرگ‌ترین صادرکننده کارت ^۴ کانادا با ۶.۵ میلیون کارت بانکی، بانک شناخته شده‌ی پیشتاز در عرصه‌ی پرداخت می‌باشد. نرمافزار این بانک بر پایه‌ی تکنولوژی HCE ^۵ می‌باشد، بدین معنی که کاربر نیازی ندارد حتی سیم‌کارت خاصی را داشته باشد (در فصل ۲ در مورد تکنولوژی HCE صحبت خواهد شد و ضعف و قوت‌های این تکنولوژی بیان خواهد شد).



شکل ۱.۶: نرمافزار موبایل RBC

در این نرمافزار کاربر می‌تواند کارت‌های اعتباری ^۶ و کارت بدهی ^۷ را در کیف‌پول ذخیره کند و با استفاده از آن پرداخت درون فروشگاه انجام دهد، می‌تواند کارت‌های جایزه را نیز به کیف‌پول خود اضافه کند و یا کارت‌های جایزه‌ی جدید را از طرح‌های وفاداری مختلف بخرد و اضافه کند، و آن‌ها را به دوستان خود ارسال کند.

ANZ Bank ۲.۲

ویژگی: انتخاب روش پرداخت بانک ANZ استرالیا اخیرا در زمینه‌ی پرداخت بدون تماس ^۸ خبرساز بود. نرمافزار این بانک روش‌های

^۱card issuer

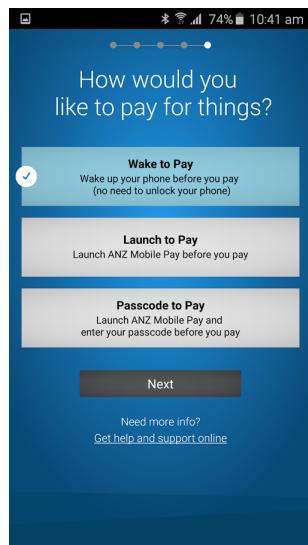
^۲Host Card Emulation

^۳credit card

^۴debit card

^۵Contactless Payment

مختلفی برای اینکه کاربر چگونه فرآیند پرداخت خود را بر روی دستگاه خود آغاز کند، ارائه می‌دهد. نرم‌افزار اجازه‌ی انتخاب نحوه‌ی فعال سازی پرداخت را به کاربر می‌دهد. بدین شکل که کاربر به منظور انجام پرداخت، تلفن‌همراه را بدون نیاز به باز کردن قفل آن فقط روشن کند و یا قبل از انجام پرداخت، کاربر نرم‌افزار بانک ANZ را اجرا کند، و یا کاربر نرم‌افزار ANZ را باز کرده و یک کلمه‌ی عبور از قبل تعیین شده را به منظور انجام پرداخت وارد کند. بدین ترتیب نرم‌افزار به کاربر اجازه می‌دهد که روشی را که برایش راحت‌تر است، انتخاب کند. در شکل ۷.۱ محیط نرم‌افزار بانک ANZ نشان داده شده است.



شکل ۷.۱: نرم‌افزار بانک ANZ

CaixaBank ۴.۲.۱

ویژگی: پرداخت با ابزارهای پوشیدنی CaixaBank صاحب شهرت در عرصه‌ی پرداخت موبایل، و بازار بزرگی از پرداخت‌های بدون تماس در کشور اسپانیا می‌باشد. در سال ۲۰۱۵ پرداخت موبایل بر پایه‌ی تکنولوژی HCE خود را ارائه کرد، و در سال ۲۰۱۶ ImaginBank را معرفی کرد، که با استفاده از نرم‌افزار ImaginPay با بهره‌گیری از تکنولوژی NFC و گوشی‌های هوشمند و ابزارهای پوشیدنی، کاربر اقدام به پرداخت می‌کند. ابزارهای نرم‌افزار در شکل ۸.۱ نشان داده شده‌اند.

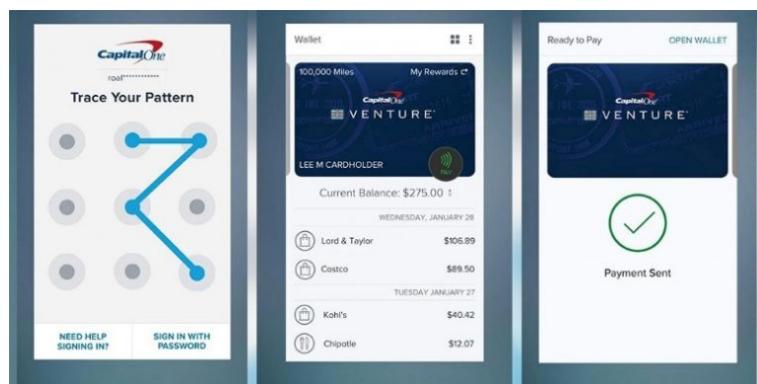


شکل ۸.۱: ابزارهای نرمافزار بانک CaixaBank

Capital One ۹.۲.۱

ویژگی: امنیت با استفاده از تکنولوژی Tokenisation

اواخر سال میلادی ۲۰۱۵ اولين بانک آمریکایی بود که پرداخت از طریق NFC درون نرمافزارهای موبایلش را عرضه کرد. مورد قابل توجه تر این است که این نرمافزار از تکنولوژی Tokenisation به عنوان مکانیزم امنیتی خود استفاده کرده است تا از اطلاعات کارتی و حساس کاربران استفاده‌ای نشوند و امن بمانند. در شکل ۹.۱ محیط نرمافزار Capital One نشان داده شده است.



شکل ۹.۱: نرمافزار بانک Capital One

SnapScan ۶.۲.۱

ویژگی: استفاده از تکنولوژی Beacon SnapScan بیش از ۳۰۰۰۰ پذیرنده^۹ را تحت پشتیبانی خود دارد و با تمامی بانک‌های آفریقای جنوبی همکاری می‌کند. این نرمافزار در زمان انجام پرداخت از بارکدهای دو بعدی^{۱۰} برای پیدا کردن فروشگاهی که کاربر در آن حاضر است، استفاده می‌کند. اما به تازگی قابلیتی به نام SnpBeacons اضافه کرده است. با استفاده از این قابلیت مکان کاربر بدون استفاده از هیچ‌گونه اسکن و فقط با حضور کاربر در فروشگاه مشخص می‌شود. در فصل ۵ به معرفی این تکنولوژی و طرز کار آن به طور کامل پرداخته خواهد شد.

Amazon Go ۳.۱

Amazon Go نوع جدیدی از فروشگاه بدون نیاز به چکنها^{۱۱} است. شرکت Amazon پیشرفته‌ترین نوع پرداخت در جهان را بوجود آورده به این شکل که نیازی به ماندن در صف برای انجام پرداخت وجود ندارد. به گفته‌ی آمازون عرضه‌ی کامل این سرویس اوایل سال ۲۰۱۷ خواهد بود.

طرز استفاده از این سرویس به این شکل خواهد بود که کاربر از نرمافزار Go برای ورود به فروشگاه استفاده می‌کند، سپس خرید خود را در فروشگاه انجام می‌دهد و با خروج از فروشگاه چندی بعد آمازون رسید پرداخت را به کاربر می‌دهد.

آمازون از تکنولوژی‌هایی که در ماشین‌های خودران^{۱۲} استفاده می‌شود بهره می‌برد: computer vision، sensor fusion، deep learning و Just Walk Out گذاشته است [۳].

۴.۱ معرفی نرمافزار پیاده‌سازی شده

در این بخش به معرفی مختصر نحوه‌ی کار و قسمت‌های مختلف نرمافزار پیاده‌سازی شده پرداخته می‌شود.

^۹Merchant

^{۱۰}QR-code

^{۱۱}checkout

^{۱۲}self-driving car

۱.۴.۱ تکنولوژی‌های استفاده شده

این نرمافزار دارای دو بخش، سمت کلاینت و سمت سرور می‌باشد. نرمافزار سمت کلاینت بر روی پلتفرم iOS پیاده‌سازی شده است و قابل استفاده و بر روی تلفن‌های همراه iPhone می‌باشد. در این نرمافزار برای پیدا کردن موقعیت فعلی کاربر از تکنولوژی iBeacon و برای اضافه کردن کالاها به لیست خرید از بارکدهای دوبعدی استفاده شده است.

نرمافزار سمت سرور متشکل از وب‌سرویس‌ها می‌باشد که از پلتفرم جاوا بهره می‌برند. برای پیاده‌سازی وب‌سرویس‌ها، Servlet Framework و برای ارتباط با پایگاهداده Hibernate Framework مورد استفاده قرار گرفته است. همچنین برای تایید شماره‌ی تلفن همراه کاربر، از سرویس پیام‌کوتاه استفاده شده است.

۲.۴.۱ نحوه کار نرمافزار

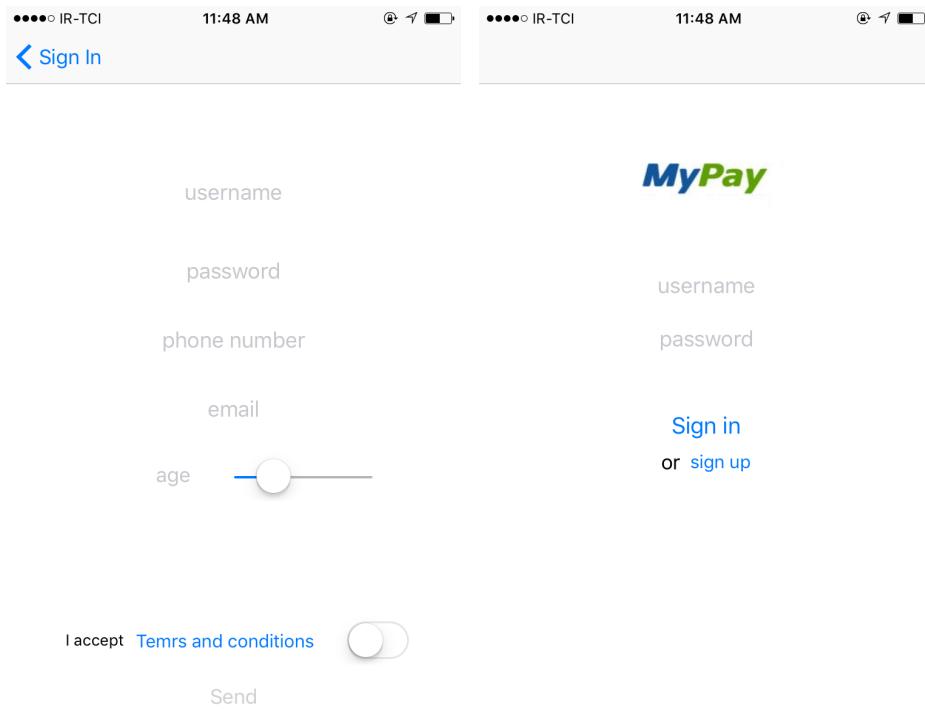
مراحل کار با نرمافزار به شکل زیر می‌باشد:

۱. در صورت اینکه کاربر از قبل دارای حساب کاربری باشد کاربر تنها اقدام به ورود می‌کند،^{۱۳} در غیر این صورت اقدام به ثبت نام می‌کند.

بعد از ثبت‌نام کاربر پیامکی حاوی یک کد چهار رقمی مبنی بر تایید شماره‌ی تلفن دریافت کرده و سپس آنرا وارد می‌کند.

شکل ۱(آ) نشان دهنده‌ی صفحه‌ی ورود و شکل ۱(ب) نشان دهنده‌ی صفحه‌ی ثبت نام کاربر می‌باشد.

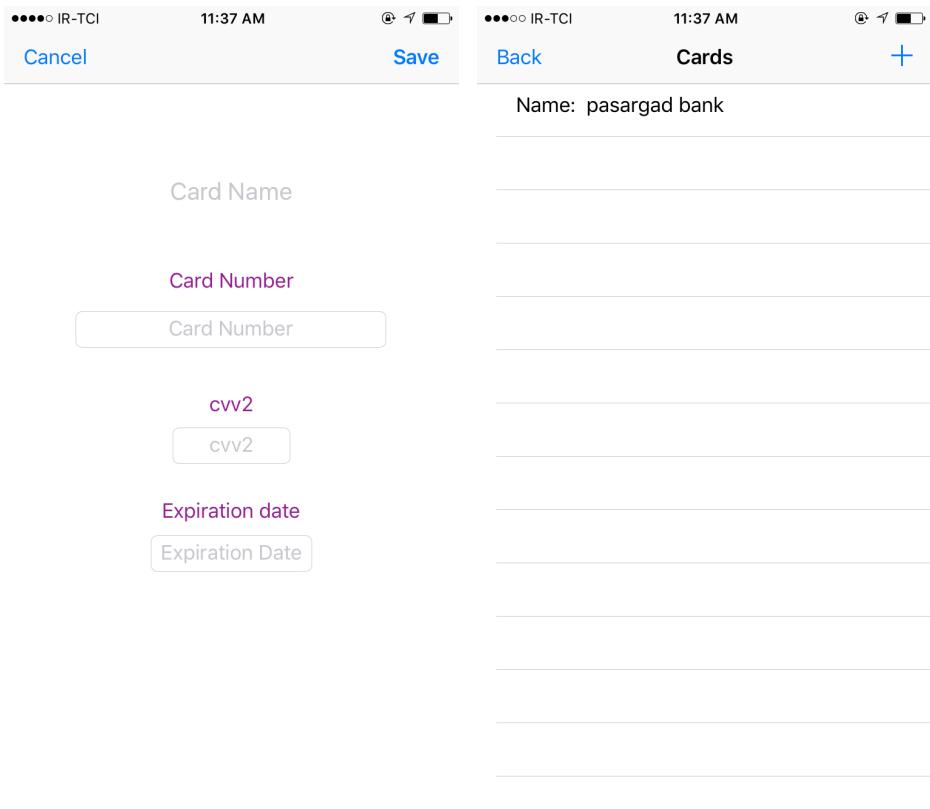
^{۱۳}sign-in



(ب) صفحه‌ی ثبت‌نام (ا) صفحه‌ی ورود

شکل ۱۰.۱: صفحات ورود و ثبت نام

۲. قبل از انجام خرید کاربر به منظور تکمیل فرآیند خرید نیازمند به فراهم کردن اطلاعات کارت بانکی می‌باشد. با لمس کردن اضافه کردن کارت کاربر وارد صفحات اضافه کردن کارت بانکی می‌شود و این عملیات را به انجام می‌رساند. شکل ۱۱.۱(آ) نشان دهنده‌ی صفحه‌ی لیست کارت‌ها و شکل ۱۱.۱(ب) نشان دهنده‌ی صفحه‌ی اضافه کردن کارت‌ها می‌باشد.

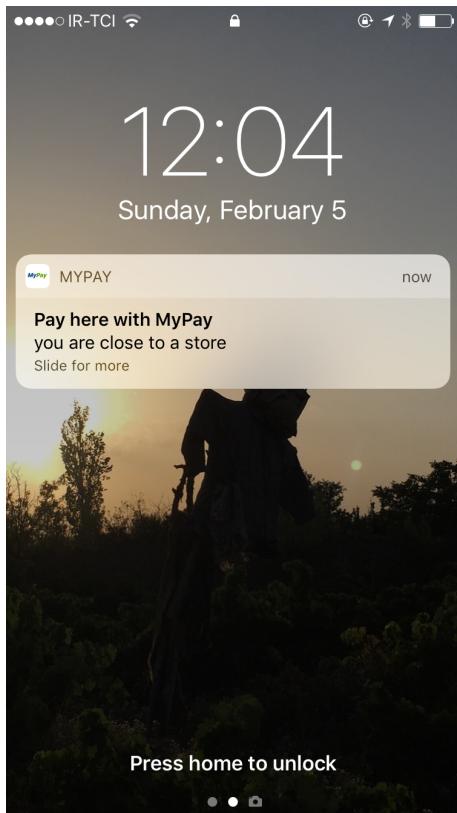


(ب) اضافه کردن کارت‌ها

(ا) لیست کارت‌ها

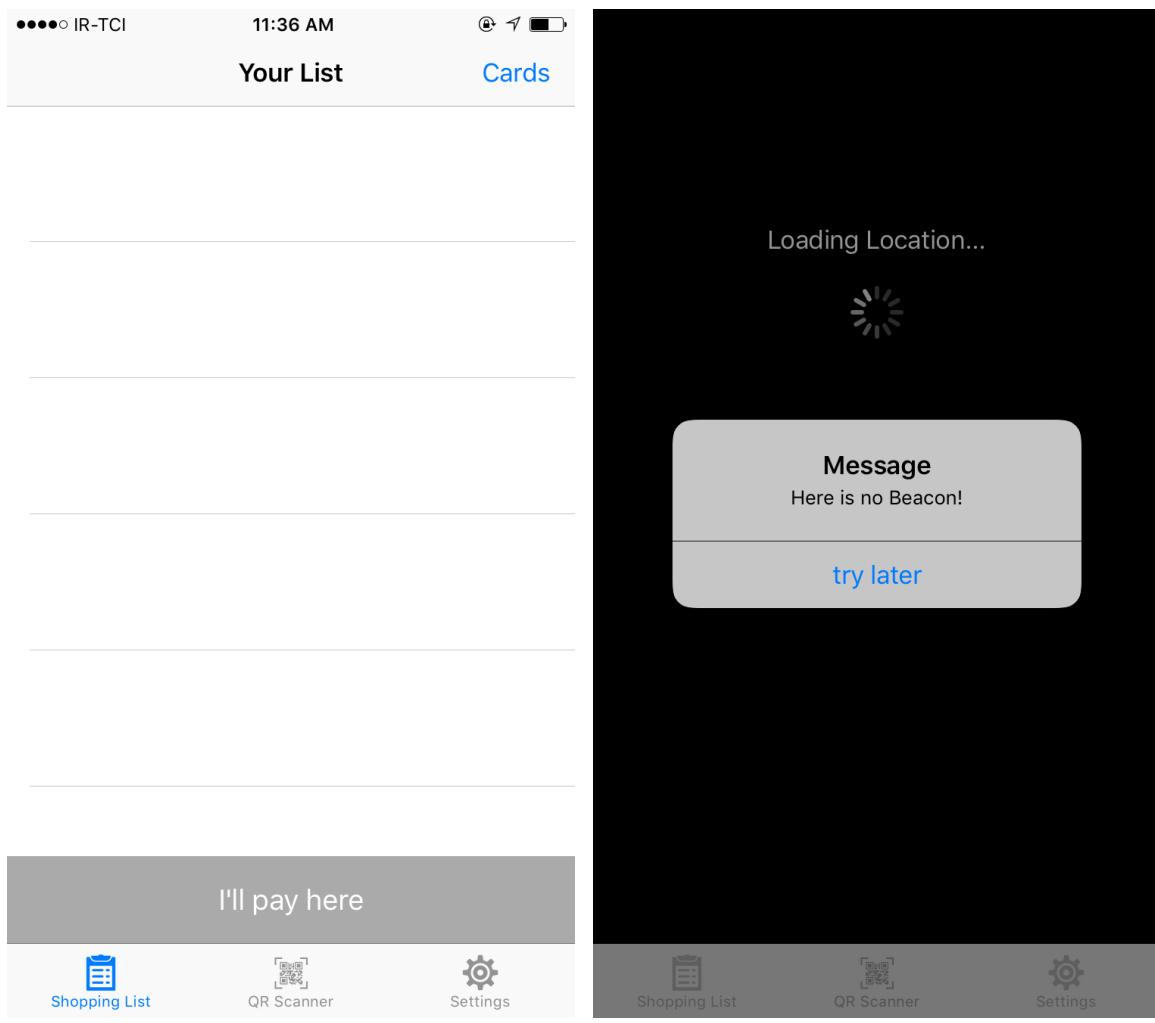
شکل ۱۱.۱: صفحات مدیریت کارت‌ها

۳. با حضور کاربر در فروشگاه‌های تحت پشتیبانی، نرمافزار، مکان فعلی کاربر را تشخیص داده و یک اعلانی به کار بر نشان می‌دهد. با باز کردن نرمافزار کاربر می‌تواند اقدام به خرید کند. شکل ۱۲.۱ نشان دهنده‌ی اعلان نشان داده به کاربر می‌باشد. در صورتی که فروشگاه تحت پشتیبانی در نزدیکی کاربر نباشد امکان انجام خرید نخواهد بود. شکل ۱۳.۱ نشان دهنده‌ی وضعیتی است که کاربر در فروشگاهی حضور ندارد.



شکل ۱۲.۱: اعلان نشان داده شده

۴. با تشخیص مکان کاربر توسط نرمافزار، مکان حاضر به کاربر اعلان می‌شود و امکان اسکن کردن کدهای کالاها فراهم می‌شود، بدین منظور کاربر با استفاده از دوربین تلفن همراه کد QR را اسکن می‌کند. شکل ۱۴.۱ نشان دهندهٔ تشخیص حضور کاربر در فروشگاه می‌باشد.



(ب) لیست خرید

(آ) صحده اسکن

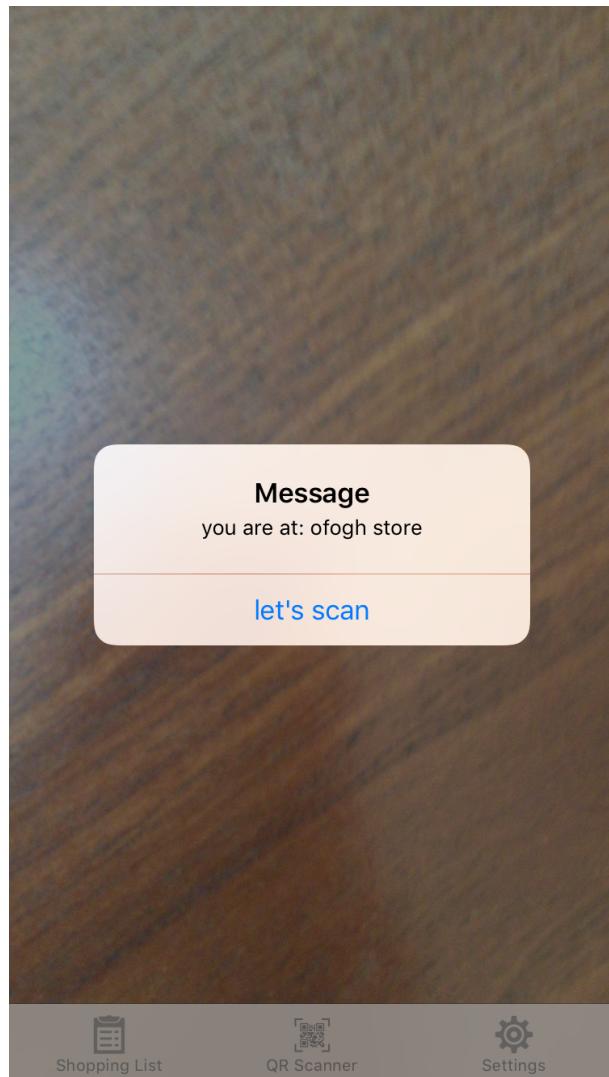
شکل ۱۳.۱: وضعیت نرم‌افزار زمانی که کاربر در فروشگاه حضور ندارد

۵. پس از اسکن بارکد دو بعدی اطلاعات کالا بعلاوه مبلغ کالا به کاربر نمایش داده می‌شود. شکل ۱۴.

۱۴.۱) اطلاعات نمایش داده شده برای کالا را نشان می‌دهد.

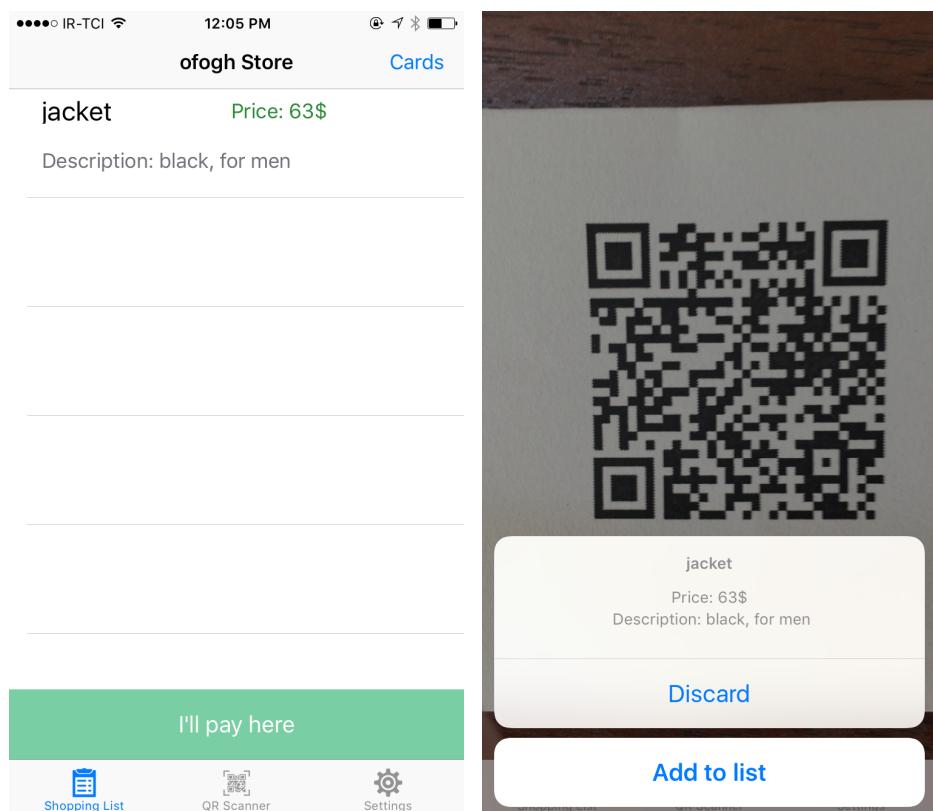
۶. کاربر اضافه به لیست خرید را لمس می‌کند و کالا به لیست خرید اضافه می‌شود. شکل ۱۴.۲)

۱۴.۲) لیست خرید بعد از اضافه کردن کالا را نمایش می‌دهد.



شکل ۱۴.۱: تشخیص حضور کاربر در فروشگاه

۷. در صفحه‌ی لیست کالاهای کاربر پرداخت در اینجا را می‌زند و وارد مراحل پرداخت می‌شود.



(ب) لیست خریدها

(أ) اطلاعات کالا

شكل ١٥.١: صفحات پس از اسکن بارکد

فصل ۲

تکنولوژی Tokenisation و زیرساخت‌ها

در این فصل به بررسی تکنولوژی‌ها و زیرساخت‌های مورد نیاز در پرداخت‌های امروزی همچون Tokenisation ، Secure Element ، Host Card Emulation ، پرداخته خواهد شد.

۱.۲ تعاریف اولیه

در ابتدا نیاز به بیان تعاریف اولیه در زمینه‌ی بانکی می‌باشد، که در این بخش به تعریف این مفاهیم خواهیم پرداخت.

۱.۱.۲ بانک صادر کننده

بانک صادر کننده^۱، بانکی است که کارت بانکی را طبق مقررات و ضوابط بانک مرکزی برای مشتریان خود صادر می‌کند.

۲.۱.۲ پذیرنده

پذیرنده^۲، شخص حقیقی یا حقوقی است که با پذیرش کارت بانکی و با استفاده از ابزار پذیرش نسبت به فروش کالا و یا ارایه خدمات به دارندگان کارت اقدام می‌نماید.

¹ Issuer Bank

² Merchant

۳.۱.۲ بانک پذیرنده

بانک پذیرنده^۳ ، بانکی است که حساب پذیرنده به منظور واریز وجهه مربوط به داد و ستدہای کارت‌های بانکی نزد آن مفتوح است.

۴.۱.۲ دارنده‌ی کارت

دارنده‌ی کارت^۴ ، شخص حقیقی است که کارت بانکی توسط بانک به نام وی صادر شده باشد.

۵.۱.۲ خدمات پرداخت

خدمات پرداخت^۵ ، عبارتند از فراهم ساختن سختافزار، نرمافزار و تجهیزات مورد لزوم و ایجاد شبکه به نحوی که پردازش تراکنش پرداخت دارنده کارت به پذیرنده را میسر سازد.

۶.۱.۲ ابزار پرداخت

به هرگونه وسیله‌ای اطلاق میگردد که دارنده آن می‌تواند نسبت به انتقال وجهه از طریق آن اقدام نماید.

۷.۱.۲ کارت بانکی

کارت بانکی^۶ نوعی ابزار پرداخت است که بانک براساس مقررات برای مشتریان خود صادر می‌کند [۴].

۸.۱.۲ شماره‌ی کارت (PAN)

شماره‌ی کارت بانکی و یا PAN^۷ ، یک کد ۱۳ تا ۱۹ رقمی مطابق با استاندارد ISO 7812، است. این شماره توسط بانک صادرکننده تولید می‌شود و بر روی کارت‌های بانکی منقش می‌شود.

^۱Acquirer Bank

^۲Card Holder

^۳Payment Services

^۴Bank Card

^۵Primary Account Number

۹.۱.۲ شبکه‌ی پرداخت

شبکه‌ی پرداخت^۸، یک سیستم الکترونیکی به منظور پذیرش، انتقال و یا پردازش تراکنش‌های ایجاد شده توسط کارت‌های پرداخت برای، کالاهای و خدمات وغیره است. این سیستم به منظور انتقال اطلاعات و وجوده در میان بانک‌های پذیرنده، پردازنده‌ها و دارندگان کارت می‌باشد.

۱۰.۱.۲ پردازنده‌ی پرداخت

پردازنده‌ی پرداخت^۹، یک موجودیت به منظور فراهم کردن خدمات پردازش پرداخت‌ها برای بانک‌های پذیرنده و یا صادرکننده می‌باشد.

۲.۲ تاریخچه تکنولوژی Tokenisation

در ماه مارس سال ۲۰۱۴ کنسرسیوم EMV^{۱۰} مشخصات فنی تکنولوژی را معرفی کرد. به دنبال آن شرکت‌های فعال در این زمینه با بهره‌وری از این تکنولوژی سرویس‌های کیف‌پول الکترونیکی خود را معرفی کردند.

۳.۲ مقدمه‌ای بر تکنولوژی Tokenisation

ذینفعان تکنولوژی Tokenisation عبارت‌اند از: بانک‌های پذیرنده، پردازنده‌ها، بانک‌های صادرکننده، و دارندگان کارت بانکی که از این تکنولوژی بهره می‌برند.

مسائلی همچون درخواست Token، صدور و فراهم کردن Token و پردازش تراکنش‌ها از مسائل مهم در زمینه‌ی Tokenisation می‌باشد. همچنین تعریف مواردی مانند، مسئولیت‌های کلیدی، تعریف اکوسیستم موجود، اصطلاحات فنی و تعریف هر موجودیت در اکوسیستم ضروری می‌باشد. در بخش ۱.۲ قسمتی از تعاریف مورد نیاز آورده شد.

^۸Payment Network

^۹Payment Processor

^{۱۰}Europay MasterCard Visa consortium (EMVco)

۱.۳.۲ خلاصه‌ی کلی

صنعت پرداخت در حال تحول و فراهم آوردن فاکتورهای جدید به منظور مقابله با کلاهبرداری‌ها در زمینه‌ی پرداخت همچون، account misuse و دیگر فرم‌های کلاهبرداری در این زمینه می‌باشد. در حالی که تراشه‌های EMV بر روی کارت‌های بانکی، محافظت قابل توجهی را برای تراکنش‌هایی که کارت بانکی در آن‌ها حضور دارند فراهم می‌آورد، نیاز مشابهی نیز به منظور کمینه کردن دسترسی‌های غیر مجاز به داده‌های حساب دارنده‌ی کارت برای تراکنش‌هایی که کارت بانکی در آن‌ها حضور فیزیکی ندارند وجود دارد. سیستم‌های پرداخت مبتنی بر PAN پاسخگوی این نیازها می‌باشد.

Token‌های پرداخت، مقادیری جایگزین برای مقدار واقعی PAN‌ها در اکوسیستم پرداخت می‌باشند. به این فرآیند Tokenisation گفته می‌شود. در این فصل به چگونگی ساخت و ملزومات ساخت این Token‌ها خواهیم پرداخت.

Token‌های پرداخت ممکن است توسط تمامی متدهای احراز هویت دارنده‌ی کارت (CVM^{۱۱}) استفاده شوند، از جمله استفاده این روش‌ها، می‌توان به امضای دیجیتالی، پین‌های آنلاین و آفلاین، و بدون CVM اشاره نمود.

به علاوه در زمان صدور Token پرداخت، مراحلی به منظور احراز هویت دارنده‌ی کارت و اطمینان حاصل کردن از صحت و اعتبار PAN لازم می‌باشد. این فرآیند به عنوان شناسایی و تشخیص صحت (ID&V^{۱۲}) شناخته می‌شود و هر بار که درخواست Token پرداخت داده می‌گیرد. انواع مختلف ID&V می‌تواند اعمال شود، که در نتیجه‌ی آن بر روی پارامتر سطح اطمینان Token^{۱۳} تأثیر دارد. برای مثال اگر با مقدار کمینه و یا بدون ID&V مورد استفاده قرار گیرد، منجر به سطح پایینی از اطمینان Token پرداخت خواهد شد، و همینطور بالعکس.

فواید زیادی برای ذینفعانی که از این اکوسیستم پرداخت استفاده می‌کنند وجود دارد که کمک به تشویق آن‌ها برای اقتباس این تکنولوژی می‌کند:

- صادرکنندگان کارت و دارندگان کارت بانکی می‌توانند از راههای جدید و امن برای پرداخت استفاده

^{۱۱}Cardholder Verification Method

^{۱۲}Identification and Verification

^{۱۳}Token Assurance

کنند، و اگر اطلاعات آنها با استفاده از کلاهبرداری‌ها فاش شود، اطلاعات به درد نخوری مثل Token به جای اطلاعات حساس مثل PAN فاش می‌شود.

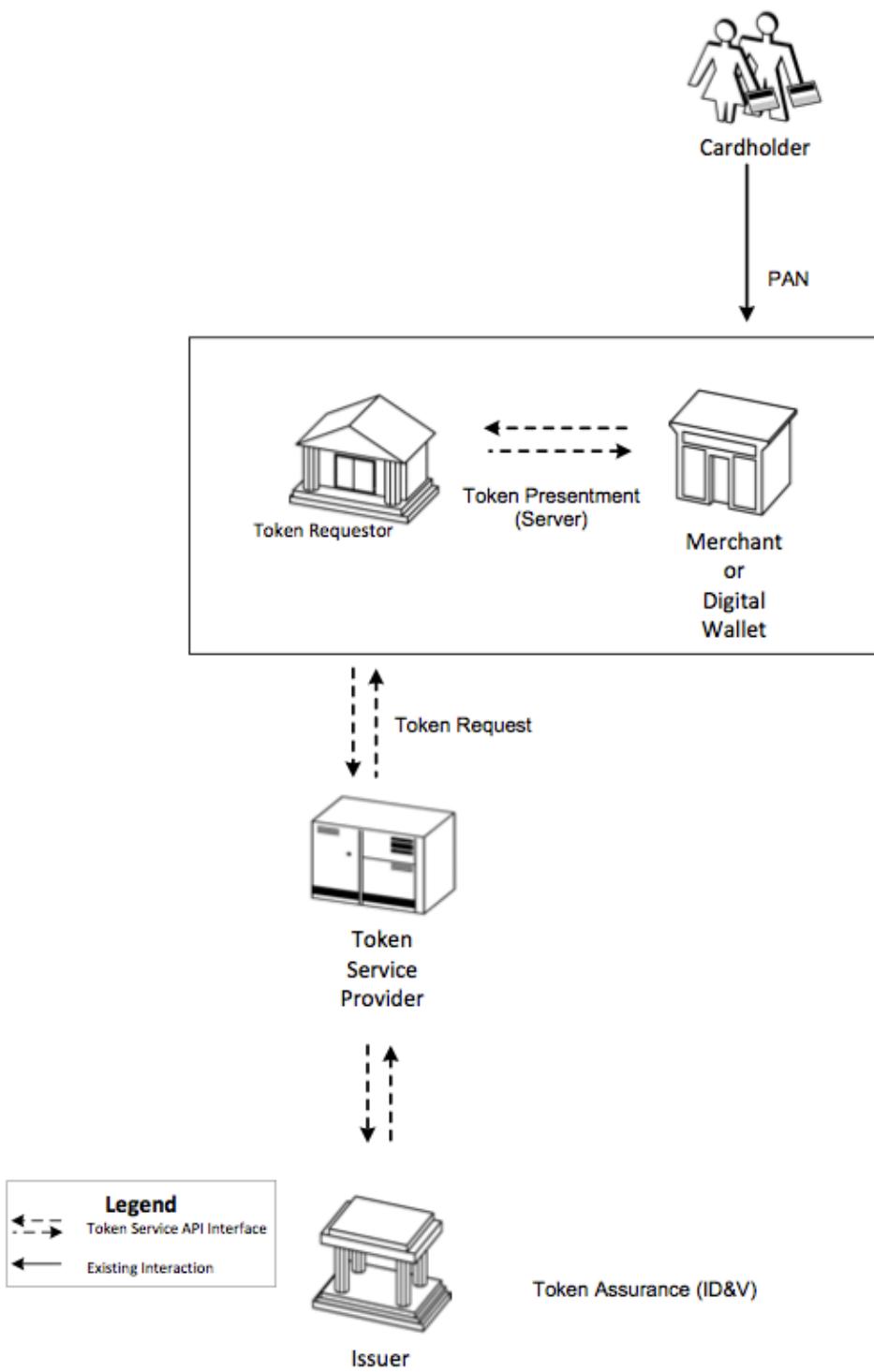
- بانک‌های پذیرنده و پذیرنده‌ها، تجربه‌ی کمتری در برابر تحدیدهای آنلاین و از دستدادن داده‌ها خواهند داشت.

- شبکه‌های پردازش پرداخت قادر خواهند بود که یک دستورالعمل و مشخصات باز را اقتباس کنند. این موضوع به بهبود قابلیت همکاری کمک می‌کند و باعث کاهش ملزمات محافظت از داده برای شبکه‌ی پرداخت و شرکای آنها می‌شود.

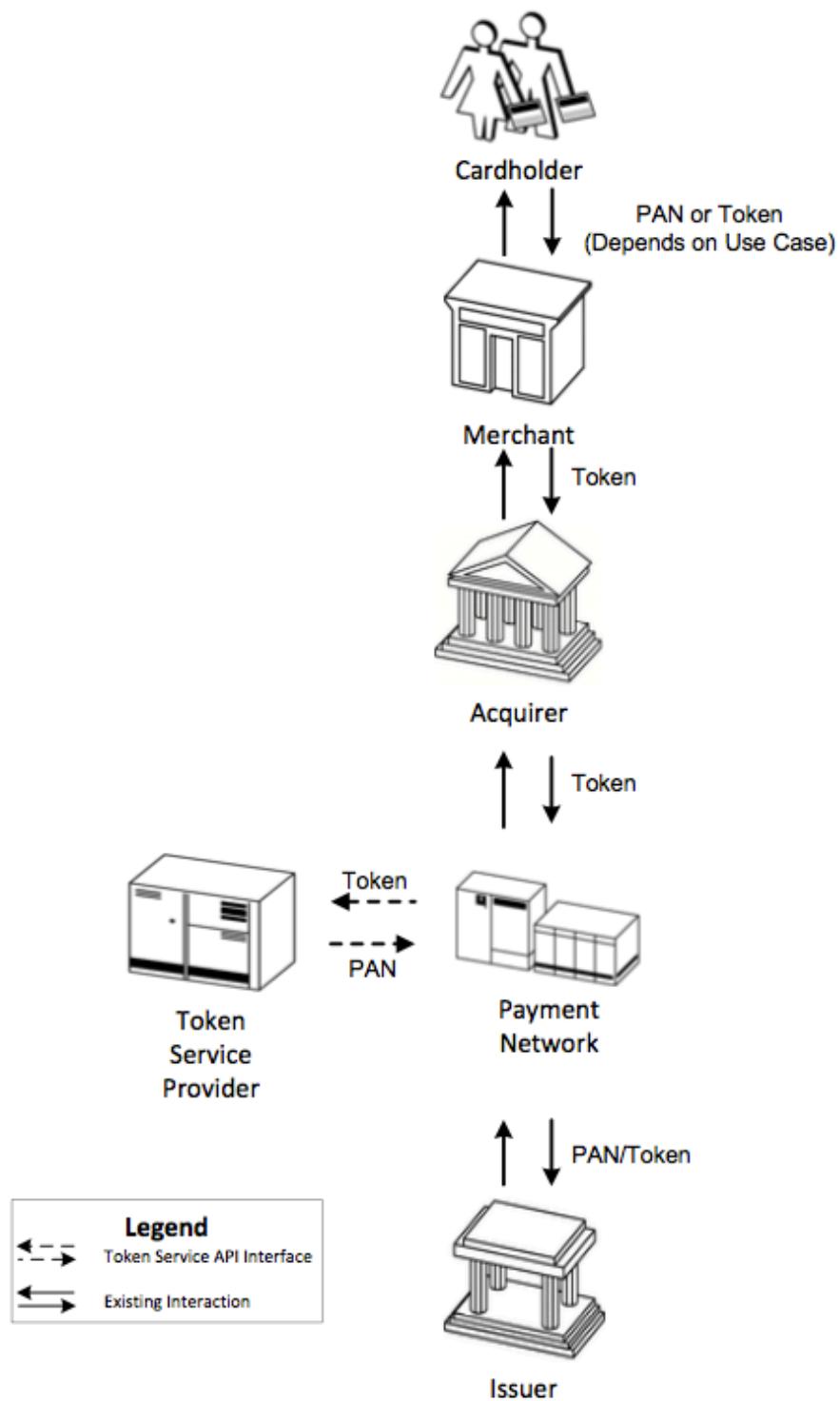
٤.٢ اکوسیستم Tokenisation

پیاده‌سازی راه حل‌های Token پرداخت مستلزم وجود نقش‌هایی درون این اکوسیستم می‌باشد. بعضی از این نقش‌ها در صنعت پرداخت سنتی حاضر موجود هستند، و برخی دیگر نقش‌های جدیدی هستند که در این اکوسیستم تعریف می‌شوند.

در دیاگرام‌های شکل ۱.۲ و ۲.۲ نقش‌های مختلف در این اکوسیستم بصورت خلاصه آورده شده است. نقش‌های موجود در بخش‌های پیش رو توضیح داده می‌شوند.



شكل ١.٢: فراهم کردن Token پرداخت



شكل ٢.٢: تراکنش Token پرداخت

Token Service Provider ۱.۴.۲

Token Service Provider (TPS) ها موجودیت‌های درون اکوسیستم Tokenisation هستند که موظف به فراهم کردن Token های پرداخت به درخواست‌دهنده‌های Token ثبت شده هستند.

Token Vault ۱.۱.۴.۲

یک انبار برای نگهداری Token های پرداخت که توسط اکوسیستم Tokenisation پیاده‌سازی شده است. از این انبار اطلاعات برای مرتبط کردن Token های مربوطه به مقادیر PAN نیز استفاده می‌شود. Vault ممکن است مقادیر دیگری از درخواست‌دهنده‌ی Token که در زمان ثبت تعیین می‌شود و ممکن است توسط TSP به منظور اعمال محدودیت‌های دامنه‌ای و یا کنترل بر روی پردازش تراکنش‌ها انجام گیرد، را ذخیره کند.

TSP ها مسئول بخش‌های مختلفی در این اکوسیستم به عنوان یک موجودیت تأیید شده برای صدور Token های پرداخت می‌باشند. برخی از این مسؤولیت‌ها شامل:

- محافظت و اداره کردن Token Vault

- تولید Token پرداخت و انتشار آن

- اختصاص و فراهم کردن Token ها

- وظایف مربوط به ثبت درخواست‌دهنده‌های Token

TSP ها مسئول ساخت و مدیریت، رابطه‌ای نرم‌افزاری درخواست‌دهنده، Token Vault، پلتفرم‌های اختصاص و ثبت‌کننده‌های Token می‌باشند.

درخواست‌دهنده‌ی Token ۲.۱.۴.۲

درخواست‌دهنده‌های Token پرداخت می‌توانند شریک‌های سنتی در صنعت پرداخت و یا شریک‌های جدیدی باشند. برخی از درخواست‌دهنده‌های Token عبارت‌اند از:

- پذیرنده‌های کارت بر روی فایل^{۱۴}

^{۱۴}Card-on-file Merchants

- بانک‌های پذیرنده، پردازنده‌های بانک‌های پذیرنده، و دروازه‌هایی به جای پذیرنده
- توانمندسازهای پرداخت مثل، تولیدکننده‌های دستگاه original equipment manufacturer (OEM)
- فراهم‌کننده‌های کیف‌پول الکترونیکی
- بانک‌های صادرکننده

در خواستدهنده‌های Token نیاز به ثبت در TSP خواهد داشت و با ملزومات ثبت، سیستم‌ها و فرآیندها مطابقت داشته باشند. بعد از یک ثبت موفق توسط یک Token، به درخواستدهنده‌ی Token یک شناسنده‌ی درخواستدهنده‌ی Token اختصاص داده خواهد شد.

موجودیت‌های دیگر در اکوسیستم Tokenisation همانند، دارنده‌ی کارت، صادرکننده، پذیرنده، بانک پذیرنده و شبکه‌ی پرداخت، نقش اصلی خود را حفظ خواهد کرد و تغییر بسزایی نخواهد داشت. در خیلی از موارد دارندگان کارت متوجه تبدیلات Token نخواهند شد و به روند عادی خود در استفاده از کارت بانکی خود ادامه می‌دهد.

در مواردی که پذیرنده‌ها به عنوان درخواست دهنده‌ی Token نیز می‌باشند باید منطبق بر رابط نرم‌افزاری TSP مورد نظر این درخواست صورت بگیرد و خود را منطبق سازد. [۵].

Secure Element ۵.۲

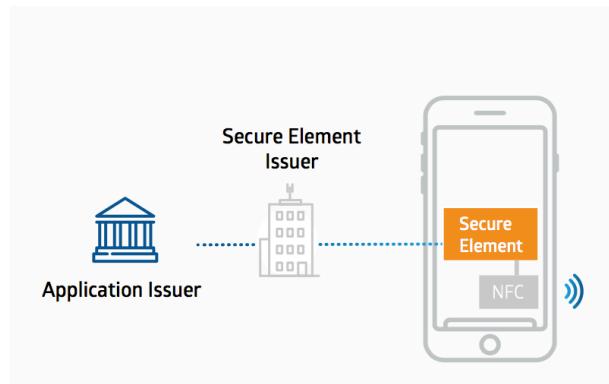
یک Secure Element (SE)، پلتفرمی سخت‌افزاری بسیار ایمن به منظور ذخیره‌ی نرم‌افزارها و داده‌های محروم‌انه و رمزگذاری شده است. بنابراین SE نقش بسیار مهمی در زمینه‌ی محاسبات موبایلی^{۱۵} بازی می‌کند.

برای مثال، در صنعت مالی از SE به منظور ذخیره‌سازی نرم‌افزارهای کارت و کلیدهای مخصوص رمزگاری که ملزومات تراکنش‌های مالی EMV بوسیله‌ی POS هستند، استفاده می‌شود. SE‌ها همچنین در بازار تعیین هویت مورد استفاده قرار می‌گیرد، که به منظور ذخیره‌سازی گواهینامه‌ها^{۱۶} و یا داده‌های بیومتریک^{۱۷} می‌باشد. در شکل ۳.۲ نمای کلی زیرساخت SE نشان داده شده است.

^{۱۵}Mobile Computing

^{۱۶}Certificate

^{۱۷}Biometric data



شکل ۳.۲

Host Card Emulation ۶.۲

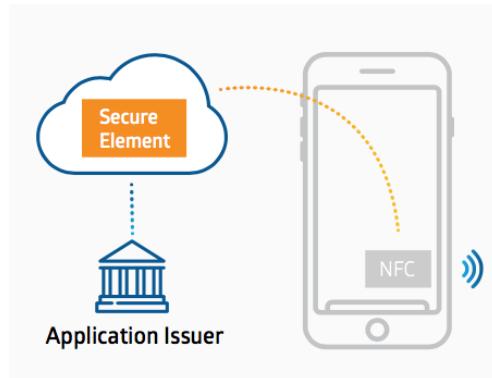
همانطور که در بخش قبل تشریح شد، به صورت مرسوم نرم‌افزارهای کیف‌پول الکترونیکی و NFC داده‌های مورد نیاز برای انجام یک تراکنش را بر روی یک تراشهٔ فیزیکی به نام SE درون یک سرویس موبایل، ذخیره می‌کنند. به همین علت مالکان SE همانند تولید کنندگان تلفن همراه و گردانندگان شبکه‌ی تلفن همراه MNO^{۱۸}، اجازه‌ی دسترسی و استفاده از این تراشه‌ها صادر می‌کنند و شارژهای مربوط به SE را انجام می‌دهند.

همچنانکه چندین و چند گرداننده‌ی SE در سراسر جهان وجود دارد، هر کدام دارای یک مدل تجاری و فی‌هستند. ارائه دهنده‌گان خدمات برای ارائه‌ی سرویس‌ها در قالب NFC ملزم به برقراری ارتباط با هریک از آن‌ها هستند. بسیاری از کارشناسان صنعت پرداخت مشکل یکی کردن سرویس‌ها و وجود چالش برای برقراری ارتباط با دارندگان SE را مشکلات اصلی در پیاده‌سازی کامل پرداخت NFC می‌دانند. با بوجود آمدن تکنولوژی (HCE) این صنعت دچار پیشرفت چشم‌گیری شد. این تکنولوژی به تراکنش‌های بر اساس NFC اجازه می‌دهد تا بدون نیاز به حضور SE و اتصال به آن، انجام شوند. بدین وسیله تکنولوژی HCE ارائه‌دهنده‌گان خدمات را قادر می‌سازد تا مشکلات مدل SE و چالش‌های یکی کردن^{۱۹} را حل و رفع کنند. در حقیقت در این روش اطلاعات ذخیره شده بر روی تراشه‌های SE بر روی فضای ابری قرار می‌گیرند و به مانند این است که SE‌ها از روی گوشی به سرورهای HCE منتقل شده‌اند. شکل

^{۱۸}Mobile Network Operator

^{۱۹}Integration

۴.۲ زیر ساخت HCE را نشان می‌دهد.



شکل ۴.۲ Host Card Emulation

۱.۶.۲ پیدایش تکنولوژی HCE

GSMA و Consult Hyperion در ماه ژانویه سال ۲۰۱۴ مقاله‌ای برای بانک‌ها و گردانندگان موبایل به منظور فهمیدن راههای HCE و SE منتشر کرد. در فوریه همان سال Visa و MasterCard پشتیبانی خود از تکنولوژی HCE را خبر دادند. به همراه آن مشخصات فنی برای این تکنولوژی نیز منتشر شد. از تاریخ آپریل ۲۰۱۴ فقط برای سرویس Google Wallet ممکن بود تا پرداخت NFC خود را از راه HCE بدون نیاز به SE انجام دهد. در سال ۲۰۱۵ شرکت مایکروسافت پشتیبانی از HCE را در سیستم‌عامل ویندوز ۱۰ خود گنجاند.

۲.۶.۲ فواید استفاده از تکنولوژی HCE

۱.۲.۶.۲ استقلال

با استقرار سرویس‌ها بر HCE هیچ نیازی به واسطه‌ها برای دسترسی به SE وجود ندارد. این امر باعث کم شدن خلاء میان ارائه‌دهنده‌ی نرم‌افزار و مشتری می‌شود.

۲.۲.۶.۲ یکی شدن آسان‌تر با شخص ثالث

در اختیار داشتن کنترل SE سهولت یکی شدن آسان‌تر با شخص ثالث را به همراه خواهد داشت.

۳.۲.۶.۲ هزینه‌های پایین‌تر

به دنبال استفاده از تکنولوژی HCE، هزینه‌های یکی کردن سرویس SE وجود نخواهند داشت.

۴.۲.۶.۲ استفاده از چندین کارت

از آنجایی که سیستم ذخیره‌سازی فیزیکی SE محدود است، با استفاده از HCE و استفاده از فضای ابری این محدودیت برای ذخیره‌سازی اطلاعات کارت‌ها رفع می‌شود و تعداد کارت بیشتری می‌تواند مورد پشتیبانی قرار بگیرد [۶].

فصل ۳

بررسی نحوهی کار سرویس‌های **Android Pay** و **Apple Pay**

در این فصل به نحوهی کار سرویس Android Pay و Apple Pay از دید فنی پرداخته خواهد شد. سپس، چگونگی استفاده‌ی هر دو سرویس از تکنولوژی Tokenisation بررسی خواهد شد. به عنوان مطرح‌ترین کیف پول الکترونیکی، سرویس Apple Pay بطور کامل مورد بررسی خواهد گرفت و در انتهای آن مقایسه‌ی دو سرویس و مزایا و معایب هر کدام پرداخته می‌شود.

Apple Pay ۱.۳

با استفاده از Apple Pay کاربران می‌توانند با دستگاه‌های iOS پشتیبانی شده و Apple Watch اقدام به پرداخت امن و راحت بکنند. انجام خرید توسط این سرویس بسیار ساده است و برای تأمین امنیت آن از قابلیت‌های نرم‌افزاری و سخت‌افزاری استفاده شده است.

همچنین برای محافظت از اطلاعات شخصی کاربران نیز طراحی شده است. اطلاعات تراکنش‌های انجام شده توسط Apple Pay ذخیره نمی‌شوند و این اطلاعات فقط بین کاربر پذیرنده و صادر کننده کارت رد و بدل خواهد شد.

Apple Pay ۱.۱.۳

Secure Element ۱.۱.۱.۳

Java Card یک استاندارد صنعتی و یک تراشه‌ی گواهی شده^۱ می‌باشد که تحت پلتفرم Secure Element است و منطبق با نیازمندی‌های صنعت مالی برای پرداخت‌های الکترونیکی می‌باشد.

NFC Controller ۲.۱.۱.۳

NFC Controller پروتکل‌های مربوط به NFC را مدیریت می‌کند و ارتباط دهنده بین پردازنده‌ی نرم‌افزار^۲ و Secure Element، و همچنین ارتباط دهنده بین POS و ترمینال^۳ می‌باشد.

Wallet ۳.۱.۱.۳

از wallet برای اضافه کردن و مدیریت کارت‌های اعتباری، بدھی و جایزه و همچنین انجام پرداخت‌ها از راه سرویس Apple Pay استفاده می‌شود. کاربران می‌توانند اطلاعات مربوط به کارت خود، صادر کننده‌ی کارت، تراکنش‌های انجام شده‌ی اخیر و سیاست‌های حریم‌شخصی^۴ بانک خود را مشاهده کنند.

Secure Enclave ۴.۱.۱.۳

Secure Enclave فرآیند احراز هویت کاربر را برعهده دارد، و قادر می‌سازد که یک تراکنش صورت پذیرد و داده‌ی اثرانگشت مورد استفاده برای تکنولوژی TouchID را فراهم می‌کند.

بر روی Apple Watch، دستگاه ممکن است نیاز به باز کردن قفل داشته باشد، و کاربر باید کنار انتهای آن را دوبار کلیک کند. این عمل دوبار کلیک کردن تشخیص داده می‌شود و به Secure Element مستقیماً ارسال می‌شود، بدون اینکه از پردازنده‌ی نرم‌افزار عبور کند.

^۱certified Chip

^۲Application Processor

^۳Point-of Sale terminal

^۴privacy policy

۵.۱.۱.۳ سرورهای Apple Pay

سرورهای Apple Pay وضعیت کارت‌های اعتباری و بدهی در Secure Device Account Number و Wallet در Elemetn را مدیریت می‌کنند. این سرورها هم با دستگاه‌های کاربران و هم با سرورهای شبکه‌ی پرداخت ارتباط برقرار می‌کنند. سرورهای Apple Pay همچنین مسئول رمزنگاری دوباره‌ی گواهی‌های پرداخت^۵ برای پرداخت درون نرم‌افزار هستند.

۲.۱.۳ چگونگی استفاده Apple Pay از Secure Element

در Secure Element ، applet های طراحی شده‌ای برای مدیریت Apple Pay ذخیره شده است و همچنین شامل applet های پرداخت که از طرف شبکه‌ی پرداخت گواهی شده است، می‌باشد. داده‌ی کارت اعتباری و یا بدهی که از شبکه‌ی پرداخت و یا صادر کننده‌های کارت ارسال می‌شوند به صورت رمزگذاری شده درون این applet ها ذخیره می‌شوند. رمزنگاری توسط کلیدهایی صورت می‌گیرد که توسط شبکه‌ی پرداخت و دامنه‌ی امنیتی applet ها شناخته شده هستند. این داده‌ها درون applet ها ذخیره می‌شوند و توسط قابلیت‌های امنیتی Secure Element محافظت می‌شوند. در هنگام انجام یک تراکنش، ترمیمال مستقیماً با Secure Element از راه NFC Controller بر روی یک گذرگاه اختصاصی ارتباط برقرار می‌کند.

۳.۱.۳ چگونگی استفاده Apple Pay از NFC Controller

به عنوان درگاهی به NFC Controller ، Secure Element از اینکه تمامی تراکنش‌های پرداخت بدون تماس به درستی به ترمیمال POS هدایت می‌شوند، اطمینان حاصل می‌کند. تنها درخواست‌هایی که از طرف میدان ترمیمال توسط NFC Controller می‌آیند به عنوان پرداخت بدون تماس علامت‌گذاری می‌شوند. بعد از اینکه اجازه‌ی پرداخت توسط کاربر با استفاده از Touch ID و یا کلمه‌ی عبور^۶ و یا در Watch با باز کردن قفل و سپس دوبار کلیک کردن گوشی انتهایی، داده شد، پاسخ‌های مربوط به پرداخت بدون تماس توسط applet های پرداخت که درون Secure Element قرار دارند فقط و فقط از طریق NFC controller به میدان NFC مسیردهی می‌شوند. بنابراین اطلاعات ردیبل شده در پرداخت بدون تماس از راه NFC فقط در میدان NFC محلی حاضر خواهد شد و این اطلاعات به هیچ عنوان از پردازنده‌ی

^۵Payment Credentials

^۶Passcode

نرمافزار عبور نخواهد کرد. در مقابل جزئیات اجازه‌ی پرداخت، در پرداخت به شکل درون نرمافزاری، بعد از رمزگاری توسط سرورهای Apple Pay به پردازنده‌ی نرمافزار مسیردهی می‌شود.

۴.۱.۳ نحوه اضافه شدن کارت‌های اعتباری و بدهی

زمانی که کاربر کارت اعتباری یا بدهی خود را به Apple Pay اضافه می‌کند، Apple اطلاعات کارت به همراه دیگر اطلاعات مثل اطلاعات حساب کاربر و دستگاه وی را به صادرکننده‌ی کارت به طور امن ارسال می‌کند. با استفاده از این اطلاعات صادرکننده‌ی کارت تعیین می‌کند که آیا این کارت می‌تواند به اضافه Apple Pay شود.

از سه فراخوانی سمت سرور برای ارتباطات ارسال و دریافت با صادرکننده و یا شبکه به منظور فرآیند اضافه کردن کارت استفاده می‌کند:

Required Fields –

Check Card –

Link and Provision –

الصادرکننده کارت و یا شبکه پرداخت از این فراخوانی‌ها به منظور اعتبارسنجی و تایید و اضافه کردن کارت‌ها به Apple Pay استفاده می‌کنند. تمامی ارتباطات Client-Server از طریق پروتکل SSL رمزگذاری می‌شوند.

شماره‌ی کارت‌ها هیچکدام به طور کامل نه بر روی سرورهای Apple و نه بر روی دستگاه‌ها ذخیره نمی‌شوند. در عوض یک Device Account Number ساخته می‌شود، رمزگاری می‌شود و بر روی Secure Element ذخیره می‌شود. Apple از اصطلاح Device Account Number به جای Token استفاده می‌کند. این شماره بطوری رمزگاری می‌شود که Apple به آن دسترسی نداشته باشد. این شماره منحصر به فرد است و با شماره‌های کارت اعتباری و بدهی متفاوت هستند. قرار Secure Element در Device Account Number برای iCloud و iOS و WatchOS کاملاً ایزوله شده اند و هیچ گاه بر روی سرورهای Apple و یا بر روی ذخیره نمی‌شوند.

برای استفاده کارت‌ها با Apple Watch از راه اضافه کردن کارت‌ها بواسطه‌ی نرمافزار Apple Watch بر روی iPhone انجام می‌پذیرد. اضافه کردن یک کارت برای Apple Watch نیازمند این است که ساعت در فاصله‌ی مجاز محدوده‌ی بلوتوث قرار داشته باشد. کارت‌ها به صورت خاص برای Apple Watch ثبت

می‌شوند و دارای Device Account Number درون خودشان هستند، که بر روی Secure Element ذخیره می‌شوند.

سه راه برای تعریف کارت‌های اعتباری و بدھی به Apple Pay وجود دارد:

- اضافه کردن کارت‌ها بصورت دستی

- اضافه کردن کارت‌ها از روی فایل از حساب iTunes

- اضافه کردن کارت‌ها از راه نرم‌افزار صادرکننده کارت

۱.۴.۱.۳ اضافه کردن کارت‌ها بصورت دستی

برای اضافه کردن کارت‌ها بصورت دستی ، اسم کارت، شماره کارت، تاریخ انقضاء و CVV2 برای سهولت فرآیند استفاده می‌شوند. کاربران از راه Settings، نرم‌افزار Wallet و یا Apple Watch می‌توانند این اطلاعات را تایپ کنند و یا از راه دوربین iSight اطلاعات را ارسال کنند. زمانی که دوربین اطلاعات کارت را دریافت می‌کند، Apple تلاش می‌کند اطلاعات لازم را از آن بدست آورد. زمانی که تمامی فیلدهای مربوطه توسط فرآیند Check Card بازرسی شدند و کامل شدند، اطلاعات به صورت رمزگاری شده به سرورهای Apple Pay ارسال می‌شوند.

اگر طی این فرآیند یک شناسنده مربوط به شرایط و ضوابط دریافت شد ، آن را دانلود می‌کند و شرایط و ضوابط صادرکننده مربوطه را به کاربر نشان می‌دهد. در صورت تأیید شرایط توسط کاربر Apple شناسنده مربوط به شرایط و ضوابط را تحت فرآیند Link and Provision به صادرکننده ارسال می‌کند. به علاوه در طی فرآیند Link and Provision، اطلاعاتی از دستگاه کاربر با صادرکننده کارت یا شبکه به اشتراک می‌گذارد، اطلاعاتی در مورد iTunes و فعالیت‌های App Store، برای مثال آیا کاربر تراکنش‌هایی بلند مدت از iTunes داشته است یا خیر، همچنین اطلاعاتی در مورد دستگاه کاربر برای مثال شماره‌ی تلفن، اسم و مدل دستگاهی که کاربر از آن استفاده می‌کند، بعلاوه مکانی که کاربر در حین انجام اضافه کردن کارت در آن لحظه قرار دارد. با استفاده از این اطلاعات بانک صادرکننده کارت تعیین می‌کند که آیا کارت اضافه شود و یا نه.

در نتیجه‌ی فرآیند Link and Provision دو اتفاق زیر رخ می‌دهد:

- دستگاه شروع به دانلود Wallet file می‌کند، که نمایان کننده کارت اعتباری و یا بدھی می‌باشد.

- دستگاه شروع به ذخیره‌ی کارت درون Secure Element می‌کند.

URL ها برای دانلود card art و اطلاعاتی مانند اطلاعات تماس و اطلاعات درباره‌ی صادرکننده می‌باشد. همچنین وضعیت فعلی را هم در اختیار دارد که شامل این است که آیا شخصی سازی Secure Element کامل شده است یا خیر و یا اینکه آیا اعتبارسنجی اضافه‌ی دیگری نیاز هست تا انجام پرداخت میسر شود.

۵.۱.۳ اجازه‌ی پرداخت

Tenha زمانی اجازه‌ی انجام یک پرداخت را می‌دهد که اجازه از Secure Eclave گرفته باشد، در واقع با تأیید کاربر که توسط Touch ID و یا کلمه‌ی عبور این اجازه صادر شده است. Touch ID روش پیش‌فرض در صورت در دسترس بودن است اما هر زمانی استفاده از کلمه‌ی عبور نیز ممکن است. بعد از سه تلاش ناموفق برای اثر انگشت کلمه‌ی عبور پیشنهاد داده می‌شود و بعد از پنج تلاش ناموفق کاربر فقط از طریق کلمه‌ی عبور می‌تواند پرداخت را انجام دهد.

ارتباطات بین Secure Element و Secure Eclave از راه یک واسط سریال صورت می‌گیرد، و اینکه Secure Controller به NFC و به مراتب به پردازنده نرم‌افزار متصل شده است. اگرچه Secure Element و Secure Eclave بصورت مستقیم بهم متصل نشده‌اند، این دو می‌توانند با استفاده از یک کلید به اشتراک گذاشته که در حین فرآیند تولید تهیه شده اند ارتباط امنی داشته باشند.

۶.۱.۳ کد امنیتی پویای مخصوص هر تراکنش

تمامی تراکنش‌ها که از applet ها نشأت می‌گیرند شامل کد امنیتی پویای مخصوص هر تراکنش^۷ همراه با یک Device Account Number می‌باشند. این کد یکبار مصرف با استفاده از یک شمارنده‌ای که در هر تراکنش به آن یک واحد اضافه می‌شود و با استفاده از یک کلیدی که در حین فرآیند اضافه شدن کارت درون applet قرار دارد، محاسبه می‌شود. این کلید توسط شبکه‌ی پرداخت و یا صادرکننده‌های کارت قابل تشخیص می‌باشد. بر اساس برنامه‌های مختلف پرداخت ممکن است داده‌های دیگری نیز در محاسبه‌ی این کد نقش داشته باشند.

⁷ Transaction-specific dynamic security code

این کدهای امنیتی به شبکه‌ی پرداخت و صادرکننده‌های کارت ارسال می‌شوند، تا اجازه‌ی انجام تراکنش صادر شود. طول این کد امنیتی بر اساس نوع تراکنشی که در حال انجام است ممکن است فرق داشته باشد.

۷.۱.۳ پرداخت از راه NFC

اگر iPhone روشی باشد و در نزدیکی میدان NFC قرار گیرد، به کاربر کارت اعتباری و یا بدهی مربوطه، یا کارت پیش‌فرض که از طریق تنظیمات مدیریت شده، نشان داده می‌شود. کاربر می‌تواند به داخل نرمافزار Wallet برود و کارت اعتباری و یا بدهی مورد نظر را انتخاب کند، و یا اگر تلفن همراه قفل است دو بار دکمه‌ی Home را فشار دهد.

در مرحله‌ی بعد کاربر باید با استفاده از اثر انگشت و یا کلمه‌ی عبور احراز هویت را انجام دهد. هیچ اطلاعات پرداختی بدون احراز هویت کاربر ارسال و جابجا نخواهد شد.

بعد از انجام احراز هویت، Device Account Number و یک کد امنیتی پویای مخصوص هر تراکنش برای پردازش پرداخت مورد استفاده قرار می‌گیرد. نه Apple و نه دستگاه کاربر، شماره‌ی کارت اعتباری و یا بدهی را به پذیرنده ارسال نخواهند کرد. Apple ممکن است اطلاعات بدون هویت تراکنش مانند زمان تقریبی تراکنش و مکان تراکنش را دریافت کند، که به بهتر شدن Apple Pay و دیگر سرویس‌های Apple کمک می‌کند.

۸.۱.۳ پرداخت درون نرمافزاری

از Apple Pay می‌توان برای پرداخت درون نرمافزار نیز استفاده کرد. زمانی که یک نرمافزار تراکنش پرداخت از راه Apple Pay را شروع می‌کند، سرورهای Apple Pay قبل از اینکه پذیرنده اطلاعات تراکنش را بگیرد، بصورت رمزگذاری شده تحويل می‌گیرند. سپس Apple اطلاعات تراکنش را دوباره با کلیدی که از پذیرنده در اختیار دارد رمزگذاری می‌کند. Apple Pay همچنین اطلاعات تراکنش همانند مقدار تقریبی خرید را نگهداری می‌کند. این اطلاعات به کاربر بازگردانده نخواهد شد و هرگز مشخص نمی‌کند که چه کالایی خریداری شده است.

زمانی که یک نرمافزار درخواست پرداخت را می‌دهد، یک API را فراخوانی که تعیین می‌کند دستگاه کاربر Apple Pay را پشتیبانی می‌کند یا خیر، اینکه آیا کاربر از قبل کارت‌های اعتباری و بدهی خود را اضافه کرده است. نرمافزار هر قسمت از اطلاعات که برای تکمیل تراکنش لازم است را درخواست می‌دهد. این اطلاعات ممکن است شامل صورت حساب و آدرس ارسال، و اطلاعات تماس باشد. سپس نرمافزار

درخواستی به iOS مبنی بر اطلاعات لازم برای نرمافزار و همچنین دیگر اطلاعات مورد نیاز مانند کارت را ارسال می‌کند.

در این زمان نرمافزار اطلاعاتی همچون کد ناحیه پستی و شهر و استان را به منظور محاسبه‌ی هزینه‌ی ارسال دریافت می‌کند. تا زمانی که کاربر اجازه‌ی پرداخت را توسط Touch ID و یا کلمه‌ی عبور صادر نکرده است، اطلاعات بصورت کامل به نرمافزار انتقال داده نخواهد شد. زمانی که اجازه‌ی پرداخت توسط کاربر صادر شود، اطلاعات به کاربر منتقل خواهد شد.

زمانی که کاربر اجازه‌ی پرداخت را می‌دهد، یک فرخوانی به سرورهای Apple Pay به منظور کسب کد رمزگذاری لحظه‌ای^۸ داده می‌شود. این کد به همراه دیگر اطلاعات تراکنش به Secure Element داده می‌شوند تا گواهی پرداخت^۹ تولید شود. سپس این گواهی توسط یک کلید Apple رمزگاری می‌شود، از Secure Element خارج می‌شوند و به سمت سرورهای Apple Pay ارسال می‌شوند. سپس اطلاعات گواهی رمزگشایی می‌شوند و کد لحظه‌ای درون گواهی با کد لحظه‌ای ارسالی از طرف Element به منظور بررسی صحت آن بررسی می‌شود و گواهی پرداخت با استفاده از کلید پذیرنده که توسط شناسنده‌ی پذیرنده قابل دسترسی است دوباره رمزگاری می‌شود و توسط API موجود به دستگاه کاربر بازگردانده می‌شود. سپس نرمافزار موجود، آن را به سیستم‌های پذیرنده به منظور پردازش عملیات پرداخت می‌فرستد. به منظور پردازش گواهی پرداخت، پذیرنده با استفاده از کلید خصوصی^{۱۰} خود می‌تواند آن را رمزگشایی کند. همراه با این، امضای دیجیتالی که از سمت سرورهای API به پذیرنده ارسال شده‌اند، این اجازه را به پذیرنده می‌دهد که از صحت گواهی پرداخت و اینکه پذیرنده درست است اطمینان حاصل کند [۷].

Android Pay ۲.۳

نحوه‌ی کار کلی سرویس Android Pay همانند سرویس Apple Pay با تفاوت‌هایی می‌باشد. سرویس Android Pay به جای استفاده از تراشه‌ای بر روی تلفن همراه، خواه تراشه‌ای بر روی سیم‌کارت و یا تراشه‌ای بر روی تلفن همراه، از تکنولوژی HCE استفاده می‌کند.

^۸Cryptographic nonce

^۹Payment Credential

^{۱۰}Private Key

مراحل کار سرویس Android Pay به صورت زیر می‌باشد:

۱. کاربر گزینه‌ی Buy With Android Pay را انتخاب می‌کند. با استفاده از رابط برنامه‌نویسی گوگل، پذیرنده درخواست شیء Masked Wallet را برای نمایش مشخصات کاربر و اطلاعات کارت، به کاربر را می‌دهد.
۲. کاربر پرداخت را تأیید می‌کند. پذیرنده با استفاده از رابط برنامه‌نویسی گوگل، درخواست شیء Full Wallet را می‌دهد. این درخواست حاوی کلید عمومی^{۱۱} پذیرنده نیز می‌باشد.
۳. گوگل درخواست Token شبکه‌ی پرداخت و Cryptogram را به شبکه‌ی پرداخت می‌دهد (معادل Transaction-specific dynamic security code در سرویس Apple Pay می‌باشد).
 - (آ) شبکه‌ی پرداخت درخواست Token پرداخت و Cryptogram را به Token Service Provider شبکه می‌دهد.
 - (ب) شبکه Token Service Provider درخواست Token پرداخت را به صادر کننده می‌دهد.
 - (ج) صادر کننده Token شبکه‌ی پرداخت و Cryptogram را به Token Service Provider شبکه می‌دهد.
 - (د) شبکه‌ی Token و Cryptogram را به شبکه‌ی پرداخت تحویل می‌دهد.
۴. شبکه‌ی پرداخت Token پرداخت و Cryptogram را به گوگل ارسال می‌کند.
۵. گوگل یک بسته اطلاعاتی که با استفاده از کلید عمومی ارسال شده در درخواست Full Wallet و آنرا در جواب رابط نرمافزاری گوگل می‌گنجاند.
۶. نرمافزار Android Pay با استفاده از فراخوانی تابع onActivityResult() داده‌ی پرداخت رمزگاری شده را به سرور پذیرنده ارسال می‌کند.
۷. پذیرنده با استفاده از زیر ساخت خود، و یا با استفاده از SDK یک پردازنده‌ی پرداخت، شروع به پردازش تراکنش می‌کند [۸].

^{۱۱}Public Key

فصل ۴

طراحی و پیاده‌سازی نرم‌افزار پرداخت سمت سرور

در این فصل پیاده‌سازی نرم‌افزار سمت سرور مورد بررسی قرار خواهد گرفت و توضیح کلی در مورد تکنولوژی‌های مورد استفاده در پیاده‌سازی همچون RESTful Services، servlet ها و Hibernate صحبت خواهد شد.

۱.۴ ساختار کلی

زبان برنامه‌نویسی به کار رفته در سمت سرور زبان Java می‌باشد. نحوه کار نرم‌افزار سمت سرور بدین شکل می‌باشد که، در سمت سرور وب‌سرویس‌های پیاده‌سازی شده آماده‌سازی سرویس دهی به درخواست‌های کلاینت‌ها می‌باشند. وب‌سرویس‌ها توسط قالب کاری servlet پیاده‌سازی شده‌اند. پس از اینکه وب‌سرویس‌ها درخواست را از کلاینت‌ها دریافت کردند، محتوای مورد نیاز را از پایگاه داده که MySQL می‌باشد با استفاده از تکنولوژی Hibernate واکاوی می‌کنند و در قالب JSON^۱، محتوای مناسب را به کلاینت‌ها تحویل می‌دهند. در بخش‌های به توضیح تکنولوژی‌های ذکر شده و نحوه پیاده‌سازی آن‌ها پرداخته خواهد شد.

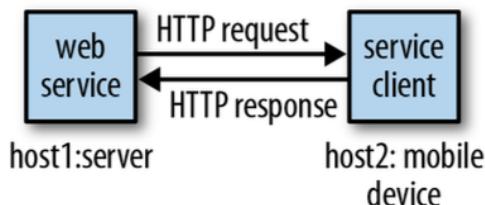
کلاس‌هایی در قالب Controller برای دریافت و پردازش درخواست‌های کلاینت‌ها تعبیه شده، و کلاس‌هایی در قالب Model به منظور ارتباط با پایگاه داده و عملیات جستجو در آن را انجام می‌دهند.

^۱JavaScript Object Notation

Web Service ۲.۴

اگرچه کلمه Web Service معانی مختلفی را شامل می‌شود، اما می‌توان بدین شکل تعریف کرد. یک Web Service شامل یک سرویس و یک کلاینت که همچنین به عنوان درخواست دهنده یا مصرف‌کننده نیز شناخته می‌شود. همانطور که اسم Web Service بیان می‌کند، یک Web Service نوعی نرمافزار بصورت وبی درآمده است و بطور معمول از پروتکل HTTP استفاده می‌کند.

Web ها می‌توانند توسط زبان‌های برنامه‌نویسی متفاوتی برنامه‌نویسی شوند. راه آشکار برای پیاده‌سازی یک Web Service توسط یک Web Server می‌باشد. یک کلاینت Web Service نیاز دارد بر روی یک ماشین که دسترسی شبکه را دارد، بر روی پروتکل HTTP اجرا شود. از دیدگاه فنی، یک Web Service یک سیستم نرمافزاری توزیع شده است که می‌تواند اجزایش بر روی دستگاه‌های از لحاظ فیزیکی مجزا مستقر شود. مثالی را در نظر بگیرید که متشکل از یک Web Server که یک Web Service را می‌بازانی می‌کند، و یک دستگاه تلفن همراه که یک نرمافزاری را می‌بازانی می‌کند که درخواست‌ها را به این Web Service ارسال می‌کند. در شکل ۱.۴ ارتباط بین یک کلاینت و سرور نشان داده شده است.



شکل ۱.۴: یک Web Service و کلاینت

یک Web Service ممکن است از لحاظ معماری بسیار پیچیده‌تر باشد، بطوری که فرض کنید، کلاینت‌ها بطور همزمان درخواست‌ها را ارسال می‌کنند، و خود Web Service امکان دارد متشکل از چند دیگر باشد.

Web Service ها در دو نوع متفاوت می‌باشند:

RESTful –

SOAP –

Web Service های بر پایه‌ی SOAP^۲ از XML به عنوان ساختار محتوای تبادلی استفاده می‌کنند. همچنین RESTful Web Service بیشتر برای طراحی‌های مستقل از پروتکل انتقال استفاده می‌شوند. Web Service های بر پایه‌ی REST^۳ برای از میان بردن پیچیدگی‌های Web Service بوجود آمده‌اند. REST توسط دکتر Roy Fielding که موضوع رساله‌ی دکتری وی بود مطرح شد. REST و SOAP کاملاً از هم متفاوت هستند. SOAP یک پروتکل ارسال پیام است و در واقع نحوه‌ی ارسال پیام‌ها را بیان می‌کند که پیام‌ها متون XML هستند، اما REST یک شیوه‌ای از معماری نرم‌افزار برای سیستم‌های مافوق رسانه‌ای توزیعی^۴، سیستم‌هایی که در آن‌ها متون، تصاویر گرافیکی، صدا و دیگر رسانه‌ها بر روی شبکه ذخیره می‌شوند، که از راه hyperlink قابل دسترسی هستند. World Wide Web مثالی روشن از یک چنین سیستمی است. در این وب، HTTP هم پروتکل انتقال است و هم سیستم پیام‌دهی، به این علت که درخواست‌ها^۵ و پاسخ‌های^۶ پیام‌های مدنظر هستند.

مزیت بسیار حائز اهمیت Web Service ها مستقل از زبان برنامه‌نویسی^۷ هستند، بدین معنی که برای مثال در سمت سرور Web Service ها به زبان Java نوشته شده‌اند اما می‌توان از طریق یک کلاینت که با استفاده از زبان Perl برنامه‌نویسی شده با آن ارتباط برقرار کرد. رابطه‌ای برنامه‌نویسی متفاوتی برای پیاده‌سازی RESTful Web Service ها در زبان Java وجود دارد که عبارتند از :

HttpServlet –

JAX-RS – که دارای پیاده‌سازی‌های مختلف است.

Restlet – که مشابه با JAX-RS می‌باشد.

JAX-WS – که به ندرت مورد استفاده قرار می‌گیرد [۹].

در این پژوهه از قالب‌کاری Servlet برای پیاده‌سازی Web Service ها و از Apache Tomcat به عنوان

^۱Simple Object Access Protocol

^۲REpresentational State Transfer

^۳distributed hypermedia system

^۴Request

^۵Response

^۶Language Neutral

서버 نرم افزار^۸ استفاده شده است.

۳.۴ تشریح نحوه پیاده سازی

۱.۳.۴ بسته Controllers

در این بسته^۹، Servlet های اصلی نرم افزار پیاده سازی شده اند. برای مثال کلاس هایی برای عملیات ورود کاربر، ثبت کاربر جدید، پیدا کردن محل حال حاضر کاربر بر اساس اطلاعات Beacon و تأیید شماره هی کاربر توسط وب سرویس SMS، که از کلاس HttpServlet ارث می برند، پیاده سازی شده است.

۱.۱.۳.۴ کلاس WhereIsBeacon

این servlet به منظور پیدا کردن مکان فعلی کاربر پیاده سازی شده است. نحوه استفاده از این کلاس بدین شکل است: زمانی که کاربر در یک فروشگاه حضور پیدا می کند، با استفاده از تکنولوژی iBeacon حضور کاربر کشف می شود. سپس سه مقدار UUID و Major و Minor از Beacon گرفته می شود و برای تعیین محل فعلی کاربر به این وب سرویس درخواستی ارسال می شود. در فصل ۵ به توضیح این تکنولوژی و مقادیر مربوطه پرداخته شده است.

کد مربوط به این کلاس به شکل زیر می باشد:

```
@WebServlet("/getstore")
public class WhereIsBeacon extends HttpServlet {
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
//set MIME type to JSON
response.setContentType("application/json");
ManageBeacon manageBeacon = new ManageBeacon((SessionFactory)
getServletContext().getAttribute("sessionfactoryobj"));
```

^۸Application Server

^۹Package

```

String uuid = request.getParameter("uuid");
int major = Integer.parseInt(request.getParameter("major"));
int minor = Integer.parseInt(request.getParameter("minor"));

StoreEntity store = manageBeacon.getStore(uuid,major,minor);
//respond to client
if (store != null)
{
    HttpSession session = request.getSession();
    session.setAttribute("currentstore",store);
    StoreResponse rsp = new StoreResponse(500,"store is
detected",store.getIdstore(),store.getName());
    Gson gson = new Gson();
    String jsonString = gson.toJson(rsp,StoreResponse.class);
    PrintWriter printWriter = response.getWriter();
    printWriter.println(jsonString);
}
else {
    StoreResponse storeResponse = new
    StoreResponse(510,"The store could not be detected",0,null);
    Gson gson = new Gson();
    String jsonString = gson.toJson(storeResponse,StoreResponse.class);
    PrintWriter printWriter = response.getWriter();
    printWriter.println(jsonString);
}
}

```

مرحله‌ی بعد از ارثبری کلاس HttpServlet پیاده‌سازی تابع doPost می‌باشد. درخواستی که از سمت کلاینت به سرور ارسال می‌شود، از متده POS HTTP استفاده می‌کند. بنابراین برای در servlet مورد نظر باید تابع doPost پیاده‌سازی شود.

با استفاده از قطعه کد زیر، مشخص می‌شود که نوع محتوایی که این وب سرویس بر می‌گرداند از چه نوعی است:

```
response.setContentType("application/json");
```

تمامی پاسخ‌ها در این نرم‌افزار به فرمت JSON می‌باشند.
در قطعه کد زیر مقادیر ارسالی توسط کاربر که در به عنوان پارامترهای درخواست هستند گرفته می‌شوند:

```
String uuid = request.getParameter("uuid");  
int major = Integer.parseInt(request.getParameter("major"));  
int minor = Integer.parseInt(request.getParameter("minor"));
```

توسط قطعه کد زیر این مکان فعلی کاربر جستجو می‌شود :

```
StoreEntity store = manageBeacon.getStore(uuid,major,minor);
```

توسط قطعه کد زیر مکان فعلی کاربر در جلسه‌ی ایجاد شده برای کاربر ذخیره می‌شود:

```
HttpSession session = request.getSession();  
session.setAttribute("currentstore",store);
```

برای تبدیل کردن مقادیر ارسالی به معادل فرمت JSON آن‌ها، کلاسی معادل فیلدات ارسالی به شکل Beans باید ساخته شود. کلاس استفاده شده برای جواب به شکل زیر می‌باشد:

```
public class StoreResponse extends Response {  
    int storeid;  
    String storename;  
  
    public StoreResponse(int resultcode, String metadata,
```

```

    int storeid, String storename) {
super(resultcode, metadata);
this.storeid = storeid;
this.storename = storename;
}

public int getStoreid() {
return storeid;
}

public String getStorename() {
return storename;
}

public void setStoreid(int storeid) {
this.storeid = storeid;
}

public void setStorename(String storename) {
this.storename = storename;
}

```

شیءای همانند کد زیر برای تبدیل آن به معادل JSON ساخته می‌شود:

```

StoreResponse rsp = new StoreResponse(500,"store is
detected",store.getIdstore(),store.getName());

```

برای تبدیل شیء ساخته شده به معادل JSON آن به شکل زیر عمل می‌شود:

```
Gson gson = new Gson();  
String jsonString = gson.toJson(rsp,StoreResponse.class);
```

برای ارسال رشته‌ی JSON بدستآمده به کلاینت به شکل زیر عمل می‌شود:

```
PrintWriter printWriter = response.getWriter();  
printWriter.println(jsonString);
```

قطعه کد زیر مربوط به زمانی است که مکان کاربر یافت نشده است:

```
StoreResponse storeResponse = new  
StoreResponse(510,"The store could not be detected",0,null);  
Gson gson = new Gson();  
String jsonString = gson.toJson(storeResponse,StoreResponse.class);  
PrintWriter printWriter = response.getWriter();  
printWriter.println(jsonString);
```

Hibernate ۴.۴

یک راه حل Object-Relational Mapping (ORM) برای پلتفرم Java است، که بصورت یک قالب کاری متن باز^{۱۰} توسط Gavin King در سال ۲۰۰۱ بوجود آمد. Hibernate یک سرویس پرس‌وجوی قدرتمند و با عملکرد بالا برای نرم‌افزارهای Java می‌باشد [۱۰].

Hibernate کلاس‌های Java را به جداول SQL نگاشت می‌کند و بین اشیاء مرسوم Java و سرور پایگاهداده O/R قرار دارد و وظیفه‌ی ذخیره‌ی اشیاء را بر روی مکانیزم‌ها و الگوهای O/R دارد. در شکل ۲.۴ ساختار R نشان داده شده است.

^{۱۰}Open Source



شکل ۲.۴ ORM/Hibernate

۱.۴.۴ اجزاری برنامه‌نویسی Hibernate

۱.۱.۴.۴ Configuration شیء

اولین شیءی که در هر نرمافزار بر اساس Hibernate و تنها یک با هم ساخته می‌شود، شیء Configuration می‌باشد. این شیء نشان دهنده‌ی تنظیمات و یا فایل‌های خصوصیات می‌باشد. شیء Configuration دو جزء اصلی را فراهم می‌کند:

: توسط یک یا چند فایل تنظیمات پشتیبانی شده توسط Hibernate انجام می‌شود. این فایل‌ها hibernate.cfg.xml و hibernate.properties هستند.

: این جزء اتصال بین کلاس‌های Java و جداول پایگاه داده را ایجاد می‌کند.

۲.۱.۴.۴ SessionFactory شیء

از شیء Configuration برای ساخت شیء SessionFactory استفاده می‌شود. شیء به اصطلاح یک شیء سبک وزن می‌باشد و در زمان شروع برنامه به منظور استفاده‌ی مجدد از آن، ساخته می‌شود. برای هر پایگاه داده نیازمند یک SessionFactory جدا خواهیم بود.

۳.۱.۴.۴ Session شیء

یک Session برای ایجاد یک ارتباط فیزیکی با پایگاه داده مورد استفاده قرار می‌گیرد. شیء Session به اصطلاح شیء سبک می‌باشد که در هر تعامل با پایگاه داده باید ساخته شود.

۴.۱.۴.۴ Transaction شیء

یک Transaction نشان‌دهنده‌ی یک واحد کاری با پایگاه‌داده می‌باشد و بیشتر RDBMS‌ها از قابلیت Transaction پشتیبانی می‌کنند.

Query شیء ۵.۱.۴.۴

شیء Query از رشته‌ی SQL و یا HQL برای نسبت دادن داده‌ها و ساختن آن‌ها استفاده می‌کنند.

Criteria شیء ۶.۱.۴.۴

شیء Criteria برای ساخت و اجرای پرس‌وجوها در محیط شیء‌گرایی و واکاوی اشیاء به کار می‌روند.

hibernate.cfg.xml ۲.۴.۴

Hibernate نیازمند آن است قبل از اینکه از آن استفاده شود، اطلاعات نگاشت به منظور تعریف چگونگی ربط کلاس‌ها به جداول پایگاه داده را بداند. همچنین Hibernate نیازمند است اطلاعات اتصال به پایگاه داده و پارامترهای مربوطه به آن داده شود. این اطلاعات در فایلی به نام hibernate.properties و یا hibernate.cfg.xml در فرمت XML آورده می‌شوند. همچنین در این فایل اسم کاربری و کلمه‌ی عبور jdbc:m مربوط به پایگاه داده نیز وارد می‌شود. نمونه‌ای از فایل تنظیمات برای پایگاه داده‌ای به نشانیysql://localhost/test به شکل زیر می‌باشد:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
"http://www.hibernate.org/
dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration> <session-factory>
<property name="hibernate.dialect">
org.hibernate.dialect.MySQLDialect
</property>
<property name="hibernate.connection.driver_class">
com.mysql.jdbc.Driver
</property>
<!-- Assume test is the database name -->
<property name="hibernate.connection.url">
jdbc:mysql://localhost/test
```

```

</property>
<property name="hibernate.connection.username">
root
</property>
<property name="hibernate.connection.password">
root123
</property>
<!-- List of XML mapping files -->
<mapping resource="BeaconEntity.hbm.xml"/>
</session-factory>
</hibernate-configuration>

```

۳.۴.۴ کلاس POJO

یک کلاس POJO^{۱۱} یک کلاس Java است که از هیچ کلاس دیگری به ارث نمی‌برد و یا رابط^{۱۲} دیگری را پیاده سازی نکرده است. همه‌ی اشیاء عادی Java به صورت POJO هستند. این کلاس‌ها در اصل منطبق با کلاس‌های JavaBeans و باید به فرمت آن‌ها باشند.

به عنوان مثال کلاس Zیر به شکل BeaconEntity.java و JavaBeans می‌باشد:

```

public class BeaconEntity {
private int idbeacon;
private String name;
private String uuid;
private Integer major;
private Integer minor;
private Integer idstore;

```

^{۱۱}Plain Old Java Object

^{۱۲}Interface

```
public BeaconEntity(int idbeacon, String name, String uuid, Integer major, Integer minor) {
    this.idbeacon = idbeacon;
    this.name = name;
    this.uuid = uuid;
    this.major = major;
    this.minor = minor;
    this.idstore = idstore;
}

public int getIdbeacon() {return idbeacon;}
public void setIdbeacon(int idbeacon) {this.idbeacon = idbeacon;}
public String getName() {return name;}
public void setName(String name) {this.name = name;}
public String getUuid() {return uuid;}
public void setUuid(String uuid) {this.uuid = uuid;}
public Integer getMajor() {return major;}
public void setMajor(Integer major) {this.major = major;}
public Integer getMinor() {return minor;}
public void setMinor(Integer minor) {this.minor = minor;}
public Integer getIdstore() {return idstore;}
public void setIdstore(Integer idstore) {this.idstore = idstore;}
}
```

۴.۴.۴ فایل Mapping configuration

با استفاده از فایل Mapping configuration کلاس‌های تعریف شده به جداول پایگاه داده نگاشت می‌شوند.
نمونه‌ای از یک فایل نگاشت:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

<class name="mypaydbmap.BeaconEntity" table="beacon" schema="mypay">
<id name="idbeacon">
<column name="idbeacon" sql-type="int(11)"/>
</id>
<property name="name">
<column name="name" sql-type="varchar(45)" length="45"/>
</property>
<property name="uuid">
<column name="uuid" sql-type="varchar(200)" length="200"/>
</property>
<property name="major">
<column name="major" sql-type="int(11)" not-null="true"/>
</property>
<property name="minor">
<column name="minor" sql-type="int(11)" not-null="true"/>
</property>
<property name="idstore">
```

```

<column name="idstore" sql-type="int(11)" not-null="true"/>
</property>
</class>
</hibernate-mapping>

```

۵.۴.۴ نحوه استفاده

پس از ساخت فایل تنظیمات ، کلاس‌های POJO، فایل نگاشت و آماده کردن جداول پایگاه داده برای اتصال به آن‌ها، آماده برای انجام فعالیت‌های مربوط به پایگاه‌داده می‌باشد. به عنوان مثال برای اضافه کردن یک حساب کاربری برای کاربر به شکل زیر توسط تابع `addUser()` انجام می‌شود:

```

public Integer addUser(String username, String password, String type, String phone, String email, Integer age) {
    try{
        SessionFactory sessionFactory = new
        Configuration().configure().buildSessionFactory();
    }catch (Throwable ex) {
        System.err.println("Failed to create sessionFactory object."
            + ex); throw new
        ExceptionInInitializerError(ex);
    }

    Session session = sessionFactory.openSession();
    Transaction tx;
    Integer id = null;
    try {
        tx = session.beginTransaction();
        UserEntity userEntity = new
        UserEntity(username, password, type, age, phone, email, (byte)1);
    }
}

```

```
id = (Integer)session.save(userEntity);
tx.commit();
}catch (Exception e){
System.err.println(e);
}
finally {
session.close();
}
return id;
}
```

فصل ۵

طراحی و پیاده‌سازی نرم‌افزار پرداخت سمت کلاینت

در این فصل ابتدا به توضیح تکنولوژی‌های استفاده شده و سپس به نحوهی عملکرد، پیاده‌سازی و چگونگی استفاده نرم‌افزار از تکنولوژی‌ها بررسی خواهد شد.

Beacon ۱.۵

یک دستگاه ساعت کننده‌ی سیگنال رادیویی Beacon است. به بیان ساده Beacon عملکردی شبیه به فانوس دریایی دارد: بطور پشت‌سرهم سیگنال‌هایی یکسان را ارسال می‌کند که از طریق دستگاه‌های دیگر قابل مشاهده است. اگرچه به جای نور مرئی، سیگنال‌های رادیویی که از ترکیبی از اعداد و حروف تشکیل شده‌اند که در فواصل زمانی معین تقریباً یک دهم ثانیه، سیگنال‌ها را مخابره می‌کند. دستگاهی که دارای تکنولوژی BLE^۱ یا Bluetooth v4.0 باشد، مانند تلفن‌های همراه هوشمند، می‌تواند سیگنال‌های Beacon را دریافت کند، همانند دریانوردی که چرا غ فانوس دریایی را می‌بیند و از مکان خود مطلع می‌شود.

۱.۱.۵ ساختار داخلی Beacon

Beacon‌ها دستگاه‌های بسیار کوچک و ساده‌ای به لحاظ الکترونیکی هستند. آن‌ها متشکل از یک پردازنده، رادیو، و باتری‌ها می‌باشند. Beacon‌ها اغلب از باتری‌های کوچک لیتیومی^۲ که مقدار انرژی بیشتری دارند

^۱Bluetooth Low Energy

^۲lithium

و کوچتر از باتری‌های نوع AA هستند و یا از راه درگاه‌های USB انرژی مورد نیاز خود را بدست می‌آورند.

در شکل ۱.۵ نمونه‌ای از دستگاه Beacon تولیدشده‌ی شرکت Estimote نشان داده شده است.



شکل ۱.۵ Estimote Beacon

۲.۱.۵ موارد استفاده

از Beacon می‌توان در، خردهفروشی^۳، آموزش، فرهنگ، فرودگاهها، رویدادها، بیمارستان‌ها و اداره‌ها استفاده کرد. از موضوعات بروز و جدید، به مسائل IoT^۴ و بازاریابی مجاورتی^۵ می‌توان اشاره کرد. در این پایان‌نامه موضوعات کاربردهای تجاری و موردهای تجاری Beacon مورد بحث نخواهد بود.

۳.۱.۵ پروتکل‌ها

پروتکل‌هایی به منظور ارتباط با Beacon‌ها وجود دارد که معروف‌ترین آن‌ها Eddystone و iBeacon می‌باشند. در ادامه به توضیح این دو پروتکل پرداخته خواهد شد. تمرکز بیشتر بر روی پروتکل iBeacon خواهد بود.

iBeacon ۱.۳.۱.۵

پروتکل iBeacon اولین و پر استفاده‌ترین پروتکل ارتباطی حال حاضر است. این پروتکل توسط Apple توسعه یافته است و توسط iOS نیز پشتیبانی می‌شود. اگر چه این پروتکل تحت دیگر پلتفرم‌ها

^۳Retail

^۴Internet of Things

^۵Proximity Marketing

سیستم عامل‌ها کار می‌کند، اما بهترین عملکرد را با دستگاه‌های iPhone و iPad دارد. در شکل ۲.۵ آیکون این تکنولوژی نشان داده شده است.



شکل ۲.۵: iBeacon icon

Eddystone ۲.۳.۱.۵

پروتکل Eddystone دارای فرمتی جدید و پروتکل ارتباطی باز است که توسط Google توسعه یافته است. بیشتر عملکرد Eddystone مشابه iBeacon می‌باشد اما دارای قابلیت‌های توسعه یافته‌ی بیشتری می‌باشد. شکل ۳.۵ آیکون این تکنولوژی را نشان می‌دهد. Eddystone دارای سه نوع مختلف است:



شکل ۳.۵: Eddystone icon

Eddystone-UID –

Eddystone-URL –

Eddystone-TLM –

Eddystone-UID کارایی همانند iBeacon دارد: انتشار قطعه کدی به فواصل زمانی یکسان. Eddystone-TLM یک URL که توسط هر کسی با یک گوشی هوشمند (خواه نرم‌افزار شمارا نصب کرده باشند یا نه) قابل مشاهده است. Eddystone-TLM داده‌های مربوط به فاصله سنجی را انتشار می‌دهند [۱۱].

iBeacon ۴.۱.۵ تکنولوژی

این تکنولوژی با عرضه‌ی iOS 7 معرفی شد و امکانات جدیدی در زمینه‌ی مکان‌یابی و آگاهی از مکان iBeacon یک دستگاه دارای تکنولوژی Bluetooth Low Energy (BLE) است. با استفاده از تکنولوژی iBeacon ارائه کرد.

می‌تواند ناحیه‌ای اطراف یک شیء را مقرر کند تا زمانی که دستگاه iOS از آن خارج شد و یا به آن وارد شد و فاصله‌ی تقریبی آن را تعیین کند.

۵.۱.۵ دستگاه‌های دارای تکنولوژی iBeacon

دستگاه‌هایی با تکنولوژی iBeacon می‌توانند برای تأمین انرژی از باتری‌های سلولی برای ماهها و یا بیشتر استفاده کنند. دستگاه‌های iOS نیز می‌توانند اعلان‌های iBeacon را انتشار دهند، اگر چه این قابلیت برای ناحیه‌ی محدودی می‌تواند باشد.

۶.۱.۵ سیگنال‌های iBeacon

اطلاعاتی که در اعلان‌های سینگال iBeacon می‌باشد به شرح زیر هستند:

UUID –

Major –

Minor –

این سه مقدار اطلاعات شناسایی را برای iBeacon فراهم می‌کنند. توسعه دهنده‌گان نرمافزار از UUID، به منظور تعریف یک شناسنده برای نرمافزار خود یا مورد کاربردی خود استفاده می‌کنند. از دو مقادیر Major و Minor هم برای تقسیم بندی‌های بیشتر استفاده می‌شود. برای مثال نرمافزار خردۀ فروشی با شبکه موجود در سراسر کشور را در نظر بگیرید که از این تکنولوژی استفاده کرده باشد. مقدار UUID یک مقدار ثابتی برای کل نرمافزار خواهد بود، و توسط تمامی مکان‌ها به اشتراک گذاشته می‌شود. این مقدار به دستگاه iOS امکان این را می‌دهد که با یک شناساگر، مناطق مربوط به این فروشگاه را تشخیص بدهد. از مقدار Major برای تقسیم بندی شهرها استفاده می‌شود، به این شکل که هر شهر مختلف دارای یک شماره‌ی Major خاص خود است. برای مثال شهر تهران مقدار ۱، شهر اصفهان ۲ و به همین ترتیب برای شهرهای دیگر. از مقدار Minor به منظور بخش‌بندی درون فروشگاه استفاده می‌شود، برای مثال بخش حسابداری مقدار ۱۰، بخش لوازم خانگی مقدار ۲۰ و بخش پوشاسک مقدار ۳۰. بدین ترتیب دستگاه iOS با استفاده از این مقادیر می‌تواند از مکان حال حاضر خود مطلع شود. هیچ یک ازین مقادیر توسط Apple ثبت نخواهد شد.

iBeacon ممکن است تکنولوژی BLE می‌باشد، از این رو نیازمند مدل‌های بالاتر از iPhone 4S، iPod نسل ۵ و iPad نسل ۳ به بالا یا iPad mini می‌توانند از این تکنولوژی استفاده کنند.

۷.۱.۵ iBeacon رابطه‌ای نرم‌افزاری

ماقبل iOS 7 از واسط کاربری Core Location برای تعریف مناطق جغرافیایی^۶ استفاده می‌شد (استفاده از طول و عرض جغرافیایی و شعاع).^۷ iBeacon قابلیتی اضافه‌تر با تعریف شناسنده برای منطقه بیان کرد.

۱.۷.۱.۵ حریم خصوصی

به این علت که iBeacon بخشی از Core Location می‌باشد، اجازه‌ی کاربر یکسانی برای استفاده از این سرویس مورد نیاز خواهد بود. زمانی که نرم‌افزاری تلاش برای استفاده از iBeacon می‌کند، کاربر هشدار اجازه‌ی یکسانی با Core Location خواهد دید.

۲.۷.۱.۵ iBeacon دقیق

برای اطمینان حاصل کردن از یک تجربه‌ی خوب برای کاربر، بررسی چگونگی تشخیص سیگنال‌ها توسط iBeacon‌ها و دقیق آن‌ها مورد اهمیت است. زمانی که یک دستگاه iOS یک سیگنال Beacon را دریافت می‌کند، دستگاه از قوت سیگنال برای تعیین دقیق و اندازه‌گیری مجاورت با آن استفاده می‌کند. هرچقدر که سیگنال قوی‌تر باشد iOS می‌تواند با اطمینان بیشتری مقدار مجاورت را تشخیص دهد. دو سرویس اصلی که iBeacon می‌دهد و در پیاده‌سازی مورد استفاده قرار گرفته است، Monitoring و Ranging می‌باشد که در ادامه به توضیح این دو سرویس پرداخته می‌شود.

Monitoring ۸.۱.۵

همانند Monitoring منطقه و یا نظارت کردن منطقه در تکنولوژی موجود جغرافیایی، یک نرم‌افزار می‌تواند ورود و خروج از یک منطقه را به کاربر اطلاع دهد. زمانی که یک نرم‌افزار درخواست برای نظارت یک منطقه را می‌دهد، حتماً باید UUID منطقه‌ی مورد نظر را ارائه دهد. در حالی که نظارت مناطق برای یک دستگاه محدود به ۲۰ UUID مختلف می‌باشد، دستگاه می‌تواند چندین مکان فیزیکی را تحت نظارت قرار دهد. برای مثال یک دستگاه می‌تواند با استفاده از تنها یک UUID چندین مکان مختلف را تحت نظارت قرار دهد. روش تحت نظارت قرار دادن با iBeacon نسبت به روش جغرافیایی فرق دارد. بدین شکل که در روش iBeacon تنها با یک خط کد می‌توان چندین مکان و یا شیء را تحت نظارت قرار داد.

^۶geofence

^۷Proximity Marketing

همانطور که قبلا هم گفته شد از مقادیر Major و Minor نیز می‌توان برای تقسیم بندی‌های بیشتر استفاده کرد. برای شروع زیر نظر قرار دادن یک منطقه، این دو مقدار ضروری نخواهند بود. همانند نظارت با استفاده از مکان جغرافیایی، زمانی که کاربر از منطقه خارج و یا وارد می‌شود، نرم‌افزار مطلع خواهد شد. اگر نرم‌افزار بسته شده باشد، و در حال کار نباشد، آگهی‌ها^۹ به صورت پشت‌پرده^{۱۰} به نرم‌افزار تحویل داده می‌شوند. یکی از ملاحظات مهم در iOS 7 در مورد این مسئله به این شکل است که اگر کاربر به صراحت اجازه به فعالیت سرویس Background App Refresh را ندهد، نرم‌افزار دیگر اعلان‌ها را دریافت نخواهد کرد. با این وجود استفاده از سرویس Ranging را می‌تواند ادامه دهد.

بر اساس BLE محدوده‌ی تحت پوشش سیگنال‌ها به ده‌ها متر می‌رسد، که از نظارت با استفاده از جغرافیا دقیق‌تر می‌باشد. در حقیقت موارد استفاده‌ی مکان‌یابی جغرافیایی، در مکان‌های سرباز می‌باشد، در صورتی که مورد استفاده‌ی iBeacon برای مکان‌های سربسته می‌باشد. با این وجود دقت در سیگنال‌های iBeacon می‌تواند بر اساس موضع فیزیکی متغیر باشد.

Ranging ۹.۱.۵

iOS 7 مجموعه‌ای جدید برای تعیین مقدار تقریبی مجاورت نسبت به یک دستگاه با استفاده از تکنولوژی iBeacon معرفی کرد، که اسم این فرآیند Ranging و یا محدوده‌یابی می‌باشد. بر اساس سناریوهای متداول، iOS فیلترهایی برای حدس میزان مجاورت نیز معرفی کرده است. این حدسهای مجاورت به چهار وضعیت زیر تعریف می‌شوند:

- Immediate: بالاترین حالت اطمینان است که نشان می‌دهد دستگاه بسیار به Beacon نزدیک می‌باشد.

- Near: دستگاه با یک دیدی واضح نسبت به Beacon قرار دارد، فاصله‌ی مجاورت حدوداً ۱ تا ۳ متر است.

- Far: این وضعیت نشان می‌دهد که یک دستگاه Beacon قابل تشخیص است، اما اطمینان در دقت برای تعیین نزدیکی آن خیلی کم است. نکته‌ی مهم در این جا این است که این وضعیت نشان

^۹Notification

^{۱۰}Background

نمی‌دهد که دستگاه از لحاظ فیزیکی به iBeacon نزدیک نمی‌باشد.

- Unknown: مجاورت نسبت به Beacon قابل تشخیص نیست. این وضعیت ممکن است نشان دهنده این امر باشد که محدوده‌یابی شروع شده است، اما معیارها برای تشخیص مقدار مجاورت کافی نیستند.

۱۰.۱.۵ محدودیت‌های فیزیکی

همانطور که قبل اشاره شد دستگاه‌های iBeacon برای انتشار سیگنال‌ها از تکنولوژی BLE استفاده می‌کنند. BLE از فرکانس 2.4 GHz استفاده می‌کند، بهمین علت توسط موانع فیزیکی همچون دیوار، در یا دیگر ساختارهای فیزیکی تحت تأثیر اثر میرایی یا تضعیف قرار می‌گیرد. فرکانس 2.4 GHz همچنان می‌تواند در معرض آب قرار بگیرد، بنابراین بدن انسان نیز بر ضعف سیگنال‌ها تأثیر خواهد گذاشت. اطلاع از موانع موجود برای سیگنال‌ها بدین دلیل مهم است که، این موانع بر روی قدرت سیگنال تأثیر مستقیم می‌گذارند، بنابراین دستگاه iOS سیگنال‌های ضعیفتری را برای تشخیص مجاورت دریافت می‌کنند [۱۲].

۲.۵ برنامه‌نویسی در iOS

برای برنامه‌نویسی در محیط iOS، می‌توان از زبان‌های برنامه‌نویسی Objective-C و Swift استفاده کرد. در این پژوهه از زبان Swift برای برنامه‌نویسی در محیط iOS استفاده شده است.

۳.۵ اجزای اصلی

AppDelegate.swift ۱.۳.۵

این کلاس دارای دو وظیفه‌ی اصلی می‌باشد:

- کلاس AppDelegate یک پنجره برای نرم‌افزار ایجاد می‌کند که در آن محتویات نرم‌افزار رسم می‌شود و انتقال بین صفحه‌های مختلف صورت می‌گیرد.

- کلاس AppDelegate یک نقطه‌ی شروع^{۱۰} و یک حلقه‌ی اجرا^{۱۱} که رویدادهای ورودی^{۱۲} به آن

^{۱۰} Entry Point

^{۱۱} Run Loop

^{۱۲} Input Event

تحویل داده می‌شوند، را می‌سازد. این کار توسط صفت ^{۱۳} @UIApplicationMain انجام می‌شود.

@UIApplicationMain ۲.۳.۵

استفاده از صفت @UIApplicationMain معادل با فراخوانی تابع UIApplicationMain به عنوان تعیین اسم کلاس AppDelegate به عنوان اسم کلاس نماینده ^{۱۴} می‌باشد. در جواب سیستم یک شیء نرمافزار ^{۱۵} تشکیل می‌دهد. شیء نرمافزار مسئول مدیریت چرخهٔ حیات ^{۱۶} نرمافزار می‌باشد. سیستم همچنین موردنی از کلاس AppDelegate می‌سازد و آن را به شیء نرمافزار مقرر می‌کند. در نهایت سیستم نرمافزار را اجرا می‌کند.

کلاس AppDelegate همچنین حاوی پیاده‌سازی توابع نماینده نیز می‌باشد:

```
func application(_ application: UIApplication,  
didFinishLaunchingWithOptions launchOptions:  
[UIApplicationLaunchOptionsKey: Any]?) -> Bool  
func applicationWillResignActive(_ application: UIApplication)  
func applicationDidEnterBackground(_ application: UIApplication)  
func applicationWillEnterForeground(_ application: UIApplication)  
func applicationDidBecomeActive(_ application: UIApplication)  
func applicationWillTerminate(_ application: UIApplication)
```

این توابع اجازهٔ برقراری ارتباط با نمایندهٔ نرمافزار را فراهم می‌کنند. برای مثال در حین فرآیند انتقال از یک صفحه به صفحهٔ دیگر و یا انتقال نرمافزار از Background به Foreground و یا بستن نرمافزار، شیء نرمافزار تابع نمایندهٔ مربوطه را فراخوانی می‌کند، و پاسخی که از قبل تعیین شده برای آن را اجرا می‌کند. فراخوانی صحیح این توابع به عهدهٔ سیستم می‌باشد و برنامه‌نویس نیاز به انجام کاری نمی‌باشد. هریک از توابع نماینده یک رفتار پیش‌فرض را دارد هستند. اگر پیاده‌سازی برای این توابع در نظر گرفته

^{۱۳}Attribute

^{۱۴}Delegate Class

^{۱۵}Application Object

^{۱۶}Life Cycle

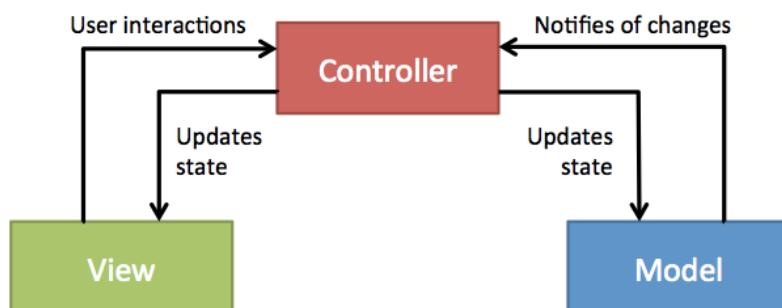
نشود، رفتار پیش‌فرض به اجرا در خواهد آمد.

Storyboard ۴.۳.۵

یک نمایش بصری از رابط کاربری نرم‌افزار می‌باشد، که صفحات مختلف، محتوی و انتقال بین صفحات را نمایش می‌دهد. برنامه‌نویس می‌تواند از این راه تغییرات بر رابط کاربری نرم‌افزار را ایجاد کند و نتایج را همان لحظه مشاهده کند.

۴.۵ الگوی طراحی MVC

همانند بسیاری از پلتفرم‌های نرم‌افزاری که از الگوی طراحی^{۱۷} MVC استفاده می‌کنند، در برنامه‌نویسی iOS نیز از این الگوی طراحی استفاده شده است. الگوی MVC شامل، View، Model و Controller است. در این سبک از طراحی Controller وظیفه‌ی ارتباط بین View‌ها و مدل داده^{۱۸} را بر عهده دارد. در این طراحی Model‌ها وظیفه‌ی نگهداری داده‌های نرم‌افزار را بر عهده دارند و View‌ها را رابط کاربری را نشان می‌دهند و محتوای نمایشی نرم‌افزار را می‌سازند. با پاسخ به اقدامات کاربران و شکل دادن محتوای View‌ها از مدل داده، Controller‌ها بعنوان درگاهی میان View و Model عمل می‌کنند. شکل ۴.۵ ارتباط بین اجزای مختلف مدل MVC را نشان می‌دهد.



شکل ۴.۵: الگوی طراحی MVC

^{۱۷}Design Pattern

^{۱۸}Data Model

UIView ۱.۴.۵

عناصری که در رابط کاربری نمایش داده می‌شوند View نام دارند. View ها محتوا را به کاربر نمایش می‌دهند. View ها رفتارهای مختلفی دارند از جمله‌ی نمایش واسطه‌های کاربری و نشان‌دادن عکس‌العمل به کاربر. در iOS تمام اشیاء View توسط نوع UIView و یا زیر‌کلاس‌های آن ساخته می‌شوند. از View های پرکاربرد می‌توان به UITextField اشاره کرد که اجازه‌ی تایپ یک متن در یک خط را به کاربر می‌دهد.

UIViewController ۲.۴.۵

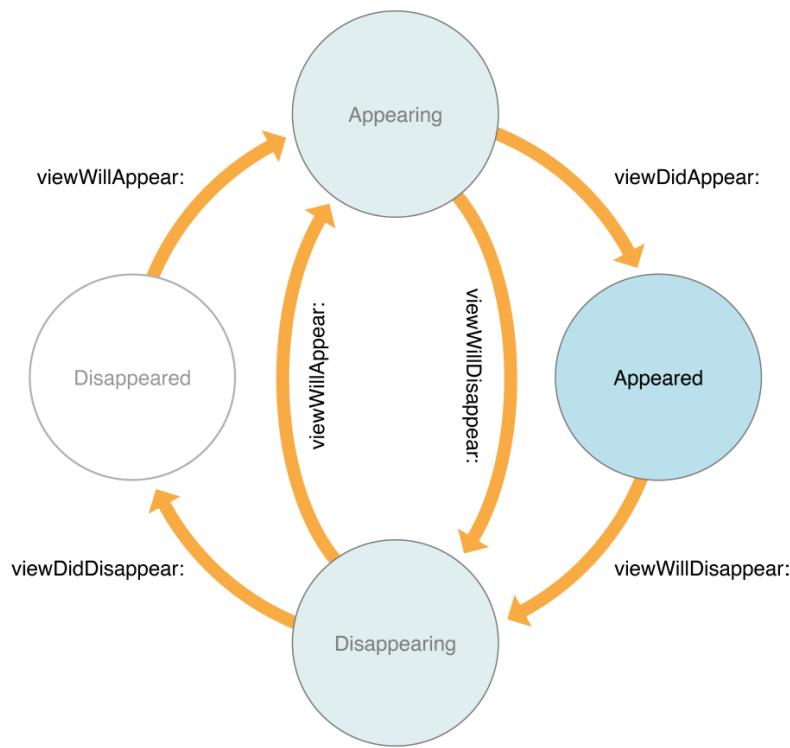
در iOS نقش ViewController را ViewController ها بر عهده دارند. همه‌ی اشیاء ViewController در iOS از نوع UIViewController و زیر‌کلاس‌های آن هستند. برنامه‌نویس با پیاده‌سازی و شخصی‌سازی توابع می‌تواند رفتار ViewController را تغییر دهد. یک شیء‌ای از کلاس ViewController به همراه یک سری از توابع به منظور مدیریت سلسله مراتب View آن می‌باشد. iOS بصورت خودکار و در زمان مناسب که ViewController در وضعیت‌های متفاوت قرار می‌گیرد، این توابع را فراخوانی می‌کند. این مسئله که چه زمانی این توابع فراخوانی می‌شوند برای تشخیص رفتار ViewController حائز اهمیت هستند. شکل ۵.۵ چرخه‌ی حیات یک ViewController و فراخوانی توابع مربوطه را نشان می‌دهد. iOS توابع را به شکل زیر فراخوانی می‌کند:

Storyboard – زمانی که محتویات یک View توسط UIViewController ساخته می‌شود و از viewDidLoad() بارگذاری می‌شود، این تابع فراخوانی می‌شود.

بطورمعمول این تابع فقط یکبار توسط iOS زمانی که محتوای View ساخته می‌شود، فراخوانی می‌شود. با این وجود محتوای View الزاماً، بار اولی که ViewController مقداردهی اولیه می‌شود، ساخته نمی‌شود.

قبل از اینکه محتوای View مربوط به سلسله مراتب ViewController اضافه شود، این تابع فراخوانی می‌شود. از این تابع برای انجام مواردی که لازم است قبل از نشان داده شدن View انجام شود، استفاده می‌شود. این تابع به سادگی نشان می‌دهد که View مورد نظر در حال اضافه شدن به سلسله مراتب View نرم‌افزار می‌باشد.

درست بعد از اینکه View مربوط به ViewController، به سلسله مراتب ViewDidApper() نرم‌افزار اضافه شود، این تابع فراخوانی می‌شود. از این تابع به منظور انجام اعمالی استفاده می‌شود



شکل ۵.۵: وضعیت‌های View Controller

که، می‌خواهیم به محض نشان دادن View انجام شوند. برای مثال واکاوی داده و یا نشان دادن یک Animation . این تابع به سادگی نشان می‌دهد که View مورد نظر به سلسله مراتب View نرم‌افزار اضافه شده است.

۵.۵ استفاده از رابط برنامه‌نویسی نرم‌افزار Core Location

برای استفاده از تکنولوژی iBeacon ، از رابط برنامه‌نویسی Core Location استفاده می‌شود. برای پیاده سازی دو قابلیت Monitoring و Ranging از کلاس AppDelegate موجود، بدلیل اینکه امکان استفاده در حالت Background وجود داشته باشد، استفاده می‌شود.

در ابتدا رابط نرم‌افزاری Core Location را به کلاس با قطعه کد زیر وارد می‌کنیم:

```
import CoreLocation
```

پس از وارد کردن رابط نرم‌افزاری، کلاس AppDelegate می‌باید برای پیاده‌سازی توابع کلاس نماینده،

از پروتکل CLLocationManagerDelegate پیروی کند. سپس نیاز به یک شیء برای تعریف نماینده و یک شیء برای مشخص کردن منطقه برای iBeacon به منظور استفاده از این تکنولوژی می‌باشد.

```
let locationManager = CLLocationManager()  
let region = CLBeaconRegion(proximityUUID:  
    NSUUID(uuidString: "163EB541-B100-4BA5-8652-EB0C513FB0F4")  
    ! as UUID, identifier: "mypay")
```

قبل از استفاده از هریک از قابلیت‌ها، باید نماینده مشخص شود و اجازه‌ی دسترسی به از کاربر باید صادر شود. بدین منظور قطعه کد زیر یک صفحه‌ی درخواست به کاربر نشان می‌دهد:

```
locationManager.requestAlwaysAuthorization()  
locationManager.delegate = self
```

این قطعه کد را در تابع application استفاده می‌کنیم.

Monitoring ۱.۵.۵

از قابلیت Monitoring تکنولوژی iBeacon به منظور اطلاع از ورود و خروج کاربر از ناحیه‌ی Beacon استفاده می‌شود.

برای شروع کار Monitoring در تابع application قطعه کد زیر را وارد می‌کنیم:

```
locationManager.startMonitoring(for: region)
```

زمانی که کاربر به محدوده‌ی Beacon وارد شود تابع زیر فراخوانی می‌شود:

```
func locationManager(CLLocationManager, didEnterRegion: CLRegion)
```

و زمانی که کاربر از محدوده خارج می‌شود، تابع زیر فراخوانی می‌شود:

```
func locationManager(CLLocationManager, didExitRegion: CLRegion)
```

پیاده‌سازی این دو تابع به شکل زیر می‌باشد:

```
func locationManager(_ manager: CLLocationManager,  
didEnterRegion region: CLRegion) {  
locationManager.startRangingBeacons(in: self.region)  
let content = UNMutableNotificationContent()  
content.title = "Pay here with MyPay"  
content.body = "you are close to a store"  
content.badge = 12  
content.sound = UNNotificationSound.default()  
// let trigger = UNTimeIntervalNotificationTrigger  
(timeInterval: 5,repeats: false)  
let request = UNNotificationRequest(identifier:  
"braconnotifi", content: content, trigger: nil)  
let center = UNUserNotificationCenter.current()  
center.add(request, completionHandler: nil)  
}  
  
func locationManager(_ manager: CLLocationManager,  
didExitRegion region: CLRegion) {  
defaults.set(false, forKey: "storestate")  
defaults.set(nil, forKey: "storename")  
defaults.set(nil, forKey: "storeid")  
defaults.synchronize()  
let content = UNMutableNotificationContent()  
content.title = "heey!"  
content.body = "you left the Beacon area"  
content.badge = 12
```

```

content.sound = UNNotificationSound.default()
//           let trigger = UNTimeIntervalNotificationTrigger
(timeInterval: 5,repeats: false)
let request = UNNotificationRequest(identifier:
"beaconnotifi", content: content, trigger: nil)
let center = UNUserNotificationCenter.current()
center.add(request, completionHandler: nil)
}

```

زمانی که کاربر وارد محدوده‌ی Beacon می‌شود، اطلاعی به کاربر به معنی ورود وی به محدوده‌ی Beacon داده می‌شود. به همین شکل زمانی که کاربر از محدوده‌ی Beacon خارج می‌شود اعلان دیگری ارسال می‌شود.

Ranging ۲.۵.۵

برای اطلاع از ورود و خروج کاربران از Monitoring استفاده شد. اما تابع مربوطه اطلاعات کافی مربوط به کشف شده را در اختیار مان قرار نمی‌دهد و به منظور بدست آوردن این مقادیر از UUID و Major و Minor استفاده می‌شود. تابع زیر زمانی فراخوانی می‌شود که تغییری در مقدار مجاورت ایجاد شود:

```

func locationManager(CLLocationManager, didRangeBeacons:
[CLBeacon], in:CLBeaconRegion)

```

پیاده‌سازی این تابع بشکل زیر می‌باشد:

```

if beacons.count > 0 {
    let knownBeacon = beacons[0]
    notifyEntryToServer(uuid: knownBeacon.proximityUUID.uuidString,
    major: knownBeacon.major.intValue,
    minor: knownBeacon.minor.intValue)
    locationManager.stopRangingBeacons(in: self.region)
}

```

```
}
```

تابع notifyEntryToServer مقادیر بدست آمده از Beacon را به منظور یافتن مکان فعلی کاربر به سرور ارسال می‌کند.

پیاده‌سازی این تابع در قطعه کد زیر آمده است:

```
let defaultConfiguration = URLSessionConfiguration.default
let delegate = self
let operationQueue = OperationQueue.main
let defaultSession = URLSession(configuration:
    defaultConfiguration, delegate:
    delegate, delegateQueue: operationQueue)

if let srvURL = URL(string: MyServer.Method.getStoreMethod) {
    var srvUrlRequest = URLRequest(url: srvURL)
    srvUrlRequest.httpMethod = "POST"

    let body = BodyMaker()
    body.appendKeyValue(key: "uuid", value: uuid)
    body.appendKeyValue(key: "major", value: String(major))
    body.appendKeyValue(key: "minor", value: String(minor))

    let bodyString = body.getBody()
    srvUrlRequest.httpBody =
        bodyString?.data(using: String.Encoding.utf8)
    let dataTask = defaultSession.dataTask(with: srvUrlRequest)
    dataTask.resume()
}
```

در این این تابع یک درخواست POST به سرور به منظور پیدا کردن مکان فرستاده می‌شود، سه مقدار UUID و Major و Minor به عنوان پارامترها ارسال می‌شوند.

برای ارسال بسته‌های HTTP در iOS از کلاس URLSession و از دو راه می‌توان این کار را انجام داد:

- استفاده از Application Handler

- استفاده از توابع نماینده

در این نرمافزار برای ارسال بسته‌ها از روش استفاده از توابع نماینده استفاده شده است. برای استفاده از این روش ابتدا کلاس AppDelegate از پروتکل URLSessionDelegate و URLSessionDataDelegate تبعیت کرده، و سپس به عنوان نماینده تعریف می‌شود. تابعی که برای دریافت پیام ارسالی از طرف سرور باید پیاده‌سازی شود، به صورت زیر می‌باشد:

```
func urlSession(_ session: URLSession, dataTask:  
URLSessionDataTask, didReceive data: Data) {  
  
var serverResultCode: Int?  
  
var serverMetaData: String?  
  
var serverStoreID: Int?  
  
var serverStoreName: String?  
  
let responseData = data  
  
// parse the result as JSON, since that's what the API provides  
do {  
  
guard let receivedData = try JSONSerialization.jsonObject(with:  
responseData,options: []) as? [String: Any] else {  
// print("Could not get JSON from responseData as dictionary")  
return  
}  
  
guard let resultCode = receivedData["resultcode"] as? Int else {  
// print("Could not get resultcode as int from JSON")  
}
```

```
return
}

serverResultCode = resultCode
guard let metaData = receivedData["metadata"] as? String else {
// print("Could not get resultcode as int from JSON")
return
}

serverMetaData = metaData
guard let storeID = receivedData["storeid"] as? Int else {
// print("Could not get resultcode as int from JSON")
return
}

serverStoreID = storeID
guard let storeName = receivedData["storename"] as? String else {
// print("Could not get resultcode as int from JSON")
return
}

serverStoreName = storeName

} catch  {
// print("error parsing response from POST on /getstore")
return
}

if serverResultCode == 500
{
defaults.set(true, forKey: "storestate")
defaults.set(serverStoreName, forKey: "storename")
```

```

defaults.set(serverStoreID, forKey: "storeid")
defaults.synchronize()
print(serverMetaData!)
}
}

```

این تابع مقدار ارسالی توسط سرور را دریافت می‌کند، آن‌ها را با استفاده از تجزیه گر JSON تجزیه نموده و در پایان اطلاعات را در UserDefaults ذخیره می‌کند.

UserDefaults ۶.۵

کلاس UserDefaults یک رابط برنامه‌نویسی برای تعامل با سیستم پیش‌فرض‌ها^{۱۹} فراهم می‌کند. سیستم پیش‌فرض‌ها به یک نرم‌افزار اجازه می‌دهد تا خودش را مطابق با تنظیمات کاربر شخصی‌سازی کند. نرم‌افزار چنین تنظیماتی را با مقرر کردن مقادیری به یک مجموعه‌ای از پارامترها در پایگاهداده‌ی پیش‌فرض‌ها ذخیره می‌کند. این پارامترها به مقادیر پیش‌فرض منتب می‌شوند به این دلیل که برای تعیین کردن وضعیت اولیه‌ی نرم‌افزار در زمان شروع استفاده می‌شوند.

در پروژه از مقادیر پیش‌فرض به منظور ذخیره‌ی مقادیر احراز هویت و وضعیت نرم‌افزار مورد استفاده قرار گرفته است.

نحوه‌ی استفاده از مقادیر پیش‌فرض بدین شکل است که ابتدا یک مورد از مقادیر پیش‌فرض گرفته می‌شود:

```
let defaults = UserDefaults.standard
```

سپس به شکل زیر پارامترها و مقادیر آن‌ها به همدیگر منتب می‌شوند:

```

defaults.set(true, forKey: "storestate")
defaults.set(serverStoreName, forKey: "storename")
defaults.set(serverStoreID, forKey: "storeid")

```

^{۱۹}system defaults

برای واکاوی مقادیر به شکل زیر عمل می‌شود:

```
defaults.bool(forKey: "storestate")
```

Model ۷.۵

همانظور که در بخش ۴.۵ ذکر شد، یکی از مؤلفه‌های الگوی طراحی MVC، Model‌ها می‌باشند که وظیفه‌ی ذخیره‌ی اطلاعات کاربر و نگهداری وضعیت نرم‌افزار را بر عهده دارند.

در پروژه برای ذخیره‌سازی کارت‌های بانکی نیازمند Model‌ها می‌باشیم که از قابلیت NSCoding استفاده شده‌است. بدین منظور کلاس Card.swift به عنوان Model و برای ذخیره‌سازی کارت‌های بانکی به شکل زیر پیاده‌سازی شده است:

```
class Card: NSObject, NSCoding {
    //MARK: Properties
    var cardNumber: Int? = nil
    var cardName: String? = nil
    var expirationDate: String? = nil
    var cvv2: Int? = nil
    var bankName: String? = nil

    struct Propertykey {
        static let cardNumberKey = "cardNumber"
        static let cardNameKey = "cardName"
        static let expirationDateKey = "expirationDate"
        static let cvv2Key = "cvv2"
        static let bankNameKey = "bankName"
    }
}

//Initialization
```

```

init(num: Int?, crdName: String?, exprdate: String?, cvv2: Int?
, bankName: String?)
{
    self.cardNumber = num
    self.cardName = crdName
    self.expirationDate = exprdate
    self.cvv2 = cvv2
    // if let theBankName = bankName{
    //     self.bankName = theBankName
    //}
}

//MARK: Archiving Paths
static let DocumentsDirectory = FileManager().urls(for:
.documentDirectory, in: .userDomainMask).first!
static let ArchiveURL =
DocumentsDirectory.appendingPathComponent("mypaycards")

//MARK: NSCoding
func encode(with aCoder: NSCoder) {
    aCoder.encode(cardNumber, forKey: Propertykey.cardNumberKey)
    aCoder.encode(cardName, forKey: Propertykey.cardNameKey)
    aCoder.encode(expirationDate, forKey:
        Propertykey.expirationDateKey)
    aCoder.encode(cvv2, forKey: Propertykey.cvv2Key)
    // aCoder.encode(bankName, forKey:
    Propertykey.bankNameKey)
}

```

```

required convenience init?(coder aDecoder: NSCoder) {
    let num = aDecoder.decodeObject(forKey:
        PropertyKey.cardNumberKey) as? Int
    let crdName = aDecoder.decodeObject(forKey:
        PropertyKey.cardNameKey) as? String
    let exprdate = aDecoder.decodeObject(forKey:
        PropertyKey.expirationDateKey) as? String
    let cvv2 = aDecoder.decodeObject(forKey:
        PropertyKey.cvv2Key) as? Int
    let bankname = aDecoder.decodeObject(forKey:
        PropertyKey.bankNameKey) as? String
    self.init(num: num, crdName: crdName, exprdate:
        exprdate, cvv2: cvv2, bankName: bankname)
}
}

```

با استفاده از روش NSCoding داده به شکل مقرر کردن مقدار به خصوصیت‌هایشان ^{۲۰} به یک کلید خاص ذخیره می‌شوند. یک کلید یک رشته‌ی ساده است. برای سهولت استفاده این رشته‌ها در قالب یک Struct درست می‌شوند. برای ذخیره‌سازی با استفاده از این روش، تابع encode(with: NSCoder) به منظور کدگذاری داده، پیاده‌سازی می‌شود و به منظور واکاوی تابع مقداردهی اولیه‌ی ^{۲۱} مناسب به منظور کدگشایی داده‌ها پیاده‌سازی می‌شود.

^{۲۰} Property

^{۲۱} init method

مراجع

- [1] *Tokenization Overview White paper.* BellID Inc, 2015.
- [2] [https://www.bellid.com/blog/how-will-we-pay-in 2020/.](https://www.bellid.com/blog/how-will-we-pay-in-2020/)
- [3] [https://www.amazon.com/.](https://www.amazon.com/) 2016.
- [4] Central Bank of Islamic Republic of Iran article.
- [5] EMVco, *Tokenisaion Specification-Technical Framework v1.0.*
March 2014.
- [6] BellID, *Host Card Emulation HCE E-book.*
- [7] *iOS Security White paper.* Apple Inc, May 2016.
- [8] CyberSource, *How Android Pay works.* October 2016.
- [9] O'Reilly, *Java Web Services 2nd Edition.*
- [10] Tutorialspoint, *Hibernate Guide.*
- [11] [https://www.kontakt.io/.](https://www.kontakt.io/)
- [12] *iBeacon Technology Overview White paper.* Apple Inc, June 2
2014.

Abstract

With security problems in electronic payment field, using digital wallet is significantly considerable. Because current payment methods expose essential card information, the most primary goals of digital wallets are as follows: preserving payment information, making transactions more secure, as well as payment facility.

In early chapters, tokenisation technology will be reviewed which has primary role in securing digital wallets. In remaining chapters, different parts of the implemented project will be investigated.

Finally, in this thesis, we have implemented an iBeacon based payment system.

keywords: Tokenisation, iBeacon, Electronic Payment, Digital Wallet



Faculty of Electrica & Computer Engineering

**B.Sc. Thesis in Computer Software
Engineering**

**Modeling, Design and Implementation of
Mobile Payments and Digital Wallets:
A Case Study**

By:

Mohammad Hesam Modaberi

Supervisor:

Dr. Morteza Dorrigiv

February 2017