

**T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

BSM 498 BİTİRME ÇALIŞMASI

GÜVENLİ ÖDEME SİSTEMİ

G191210068 – ÖMER FARUK GÜZEL

**Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Danışmanı : Doç. Dr. Ünal ÇAVUŞOĞLU**

2022-2023 Bahar Dönemi

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

GÜVENLİ ÖDEME SİSTEMİ

BSM 498 - BİTİRME ÇALIŞMASI

ÖMER FARUK GÜZEL

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Bu tez .. / .. / ... tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

.....
Jüri Başkanı

.....
Üye

.....
Üye

ÖNSÖZ

Ödeme işlemleri gerçekleştiren şirketlerin kullanıcıları, ödeme yapacakları zaman güvencikleri ve bizzat kendilerinin de kaydolduğu bir sistemden ödeme yapmak isterler. Ödeme alan şirketlerin kullanıcıları, kartlarını bir sisteme bağlamak ve bağladıkları sistemden hızlıca ve güvenli bir şekilde ödeme işlemini gerçekleştirmeyi amaçlarlar. Çalışması yapılan bu ödeme sisteminin avantajları ise kullanıcıların arzuladığı işlemleri kullanıcılarına sunar. Sunulan hizmetler karşısında kullanıcılara güven duygusunu ve hızlı ödeme işlemi desteğini sunar.

İÇİNDEKİLER

ÖNSÖZ	iii
SİMGELER VE KISALTMALAR LİSTESİ.....	vi
ŞEKİLLER LİSTESİ	vii
ÖZET	ix
BÖLÜM 1. GİRİŞ	10
1.1. 3-D Secure	10
1.1.1. 3-D Secure sisteminin çalışma prensibi.....	10
1.1.2. 3-D Secure sisteminin çalışma aşamaları	11
1.1.3. 3-D Secure sisteminin avantajları	11
BÖLÜM 2. Ödeme Sisteminde Kullanılacak Teknolojiler	12
2.1 Spring Boot	12
2.2 Android/Kotlin	13
2.3 Apache Kafka	13
2.4 PostgreSQL	14
2.5 Docker	15
BÖLÜM 3. Güvenli Ödeme Sistemi	16
3.1 SauPay ile Güvenli Ödeme İşlemi	16
3.2 Güvenli Ödeme İşleminde İmzalama ve Şifreleme	17
3.2.1 Android Tarafında Güvenli Ödeme İşleminde İmzalama ve Şifreleme	18
3.2.2 Backend Tarafında Güvenli Ödeme İşleminde İmzalama ve Şifreleme.....	19
3.2.3 SHA-256 Algoritması	20
3.2.4 Güvenli Ödemede Verilerin SHA-256 Algoritması ile İmzalanması.....	20
3.2.5 AES Algoritması.....	21
3.2.6 Güvenli Ödemede Verilerin AES Algoritması ile Şifrlenmesi	22
3.3 Ödeme Sistemi ile Bağlantılı Uygulamalar Arasındaki Akış	23
3.3.1. Akış Diyagramı İşlem Açıklamaları	25
3.3.2. Kullanıcı İşlemlerinin Keycloak ile Gerçekleştirilmesi.....	27
3.3.2.1 Register İşleminin Keycloak ile Gerçekleştirilmesi	27
3.3.2.2 Backend Tarafında Register İşleminin Keycloak ile Gerçekleştirilmesi	29
3.3.2.3 Login İşleminin Keycloak ile Gerçekleştirilmesi	29
3.3.2.4 Request İşleminin Keycloak ile Gerçekleştirilmesi	30
3.3.2.5 Keycloak ile ApiGateway Konfigurasyonu	31
3.3.2.6 Keycloak ile User Service konfigürasyonu	32

3.3.3. Saupay Ödeme Sistemi Veritabanı Diyagramı	33
3.3.4. Saupay Backend MicroService Mimari Kodlaması	34
3.4.1 SauPay ödeme uygulaması mobil arayüz tasarımı.....	34
3.4.1.1. SauPay ödeme uygulaması giriş sayfası mobil arayüz tasarımı	34
3.4.1.2. SauPay ödeme uygulaması kaydol sayfası mobil arayüz tasarımı	36
3.4.1.3. SauPay ödeme uygulaması anasayfa sayfası mobil arayüz tasarımı	37
3.4.1.4. SauPay ödeme uygulaması ödeme tamamlama sayfası mobil arayüz tasarımı ...	38
3.4.1.5. SauPay ödeme uygulaması ödeme işlemi kart seçimi sayfası mobil arayüz tasarımı	39
3.4.1.6 SauPay ödeme uygulaması 3D secure/ödeme sayfası mobil arayüz tasarımı	40
3.4.1.7 SauPay ödeme uygulaması ödeme onayı sayfası mobil arayüz tasarımı	41
3.4.2 SauGetir Alışveriş Uygulaması Mobil Arayüz Tasarımı	42
3.4.2.1 SauGetir alışveriş uygulaması sipariş tamamlama sayfası mobil arayüz tasarımı .	42
3.4.3 SauBank Banka Uygulaması Mobil Arayüz Tasarımı	43
3.4.3.1 SauBank banka uygulaması anasayfa sayfası mobil arayüz tasarımı	43
BÖLÜM 4. Ödeme Sistemi Testi İçin Tasarlanan Uygulamalar	45
4.1 Gerçekleştirilen Uygulamalar	45
4.1.2. SauPay uygulaması.....	45
4.1.3. SauGetir uygulaması	45
4.1.4. SauBank uygulaması.....	46
4.2 Gerçekleştirilen Uygulamalar Arası İletişim	46
4.2.1. Önyüz Haberleşmesinde Kullanılan DeepLink Yöntemi	46
4.2.2. Arka Yüz Haberleşmesinde Kullanılan Rest Api Yöntemi.....	46
BÖLÜM 5. Güvenli Ödeme Uygulaması Kullanımı ve Sonuçları	47
5.1 SauPay Ödeme Sistemi Kullanımının Amaçlanması	47
5.2 SauPay Ödeme Sistemi Kullanımının Avantajları	47
5.3 İşletmelerin SauPay Ödeme Sistemine Entegre Olması	48

SİMGELER VE KISALTMALAR LİSTESİ

SQL	: Structured Query Language
XML	: Extensible Markup Language
API	: Application Programming Interface
JPA	: Java Persistence Api
LXC	: Linux Containers
URI	: Uniform Resource Identifier
JSON	: JavaScript Object Notation

ŞEKİLLER LİSTESİ

Şekil 2.1.	Spring Boot Logo.....	12
Şekil 2.2.	Kotlin Logo.....	13
Şekil 2.3.	Apache Kafka Logo.....	13
Şekil 2.4.	PostgreSQL Logo.....	14
Şekil 2.5.	Docker Logo.....	15
Şekil 3.2.	Verilerin Şifrelenmesi, İmzalanması ve Doğrulanması.....	17
Şekil 3.2.1	Android Tarafında Verilerin Şifrelenmesi, İmzalanması ve Doğrulanması.....	18
Şekil 3.2.2	Backend Tarafında Verilerin Şifrelenmesi, İmzalanması ve Doğrulanması.....	19
Şekil 3.2.4	Verilerin SHA-256 Algoritması ile İmzalanması.....	21
Şekil 3.2.6	Verilerin AES Algoritması ile Şifrelenmesi.....	23
Şekil 3.3	Uygulamalar arası bağlantılı akış diyagramı.....	24
Şekil 3.3.2.1	Register İşleminin Keycloak ile Gerçekleştirilmesi.....	28
Şekil 3.3.2.2	Backend Tarafında Register İşleminin Keycloak ile Gerçekleştirilmesi.....	29
Şekil 3.3.2.3	Login İşleminin Keycloak ile Gerçekleştirilmesi.....	30
Şekil 3.3.2.4	Request İşleminin Keycloak ile Gerçekleştirilmesi.....	31
Şekil 3.3.2.5	Keycloak Api Gateway Konfigurasyonu.....	32
Şekil 3.3.2.6	Keycloak User Service Konfigurasyonu	32
Şekil 3.3.3	Saupay Ödeme Sistemi Veritabanı Diyagramı.....	33
Şekil 3.3.4	Saupay Backend MicroServices Mimari Kodlaması.....	34
Şekil 3.4.1.1	Ödeme Uygulaması Giriş Ekranı Tasarımı.....	35

Şekil 3.4.1.2	Ödeme Uygulaması Kaydol Ekranı Tasarımı.....	36
Şekil 3.4.1.3	Ödeme Uygulaması Anasayfa Ekranı Tasarımı.....	37
Şekil 3.4.1.4	Ödeme Uygulaması Ödeme Tamamlama Ekranı Tasarımı	38
Şekil 3.4.1.5	Ödeme Uygulaması Kart Seçim Ekranı Tasarımı.....	39
Şekil 3.4.1.6	Ödeme Uygulaması 3-D/Secure Ödeme Ekranı Tasarımı...	40
Şekil 3.4.1.7	Ödeme Uygulaması Ödeme Doğrulama Ekranı Tasarımı...	41
Şekil 3.4.2.1	E Ticaret Uygulaması Sipariş Tamamlama Ekranı Tasarımı	43
Şekil 3.4.2.1	Banka Uygulaması Anasayfa Ekranı Tasarımı.....	44

ÖZET

Anahtar kelimeler: 3-D Secure, Güvenli Ödeme, Doğrulama, Şifreleme

Tasarlanan ödeme sistemi, kullanıcıların sisteme entegre olmuş işletmeler üzerinden kart bilgilerinin saklanması ve veri akışının güvenli iletilmesiyle birlikte sorunsuz, güvenli ve ödeme sahtekarlığının önüne geçilen bir ödeme işlemi hizmeti sunmayı amaçlamaktadır. Ödeme sisteminin tasarımında ödeme akışı çift doğrulama üzerine kurulmuştur. Entegre olan sistemler arasındaki iletişim akışı ise şifreleme algoritmasına dayanarak şifrelenmiş, gizli bir veri iletişimini esas alır. Bu ödeme sisteminin gerçekleştirilmesinde farklı teknolojilerin kullanımı ve deneyimi amaçlanmaktadır.

Sistem tasarlanırken 3 farklı alanda çalışan mobil uygulama önyüzü ve bu ön yüzlerin arka işlerini gerçekleştiren arka yüzleri ile tasarlanmış olup içerisinde güvenliği sağlayacak algoritmalar, kullanıcı ve gerçek veri doğrulaması ile desteklenen şirketlerin kolaylıkla entegre olabileceği, bankaların rahatlıkla güvenebileceği birlikte çalışan bir ödeme sistemi hayata geçirilmesi amaçlanmıştır.

Buna bağlı olarak amaçlanan ve gerçekleştirilen ödeme sisteminin kullanımında ve geliştirilmesinde güvenli ödeme alanında doğru, gizli ve güvenli ödeme işlemlerinin gerçekleştirilmesine olanak sağlar. Ödeme sahtekarlığının ve güvensiz ödeme sistemlerinin önüne geçerek şirketlere ve kullanıcılara rahatlıkla güvenebilecekleri şifrelenen veri aktarımını gerçekleştiren bir sistemi vadetmektedir.

BÖLÜM 1. GİRİŞ

Güvenli ödeme, internet üzerinden kredi kartı ile yapılan ödemelerde kart sahibi, işletme ve banka arasında güvenli bir bilgi akışı sağlamak, kart sahibi tarafından yapılan doğrulama sürecinden geçerek kart sahibinin bilgisi dahilinde bir ödeme işlemi gerçekleştirmek ve kart bilgileri ile kullanıcı verilerini şifrelenmiş bir şekilde aktarıp güvenli bir şekilde depolamayı, saklamayı amaçlamaktadır.

1.1. 3-D Secure

3-D Secure, 3 boyutlu XML tabanlı güvenlik protokolüdür. İnternet üzerinden banka kartı ile yapılan alışverişlerde banka, kart sahibi ve alışveriş yapılan işletme arasında aktarılan bilgilerin, sadece entegre olan sistemin çözeceği bir şekilde şifrelenmesini, güvenli bir ortamda saklanılmasını ve çift taraflı doğrulanmasını amaçlamaktadır. Çift taraflı doğrulama, veriyi gönderilen sistemde depolayıp alınan sistemden karşılaştırma ile depolanan veriyi doğrulayarak güvenliğini sağlar. İki aşamalı doğrulamadaki güvenlik şifresi sadece kart sahibine ve genellikle cep telefonuna SMS olarak gönderilir.

Böylece internet alışverişi söz konusu olduğunda karşılaşılan çalıntı kart kullanımı, banka kartı ya da kredi kartı bilgilerinin kopyalanması gibi durumlar önemli ölçüde önlenmiş olur.

1.1.1. 3-D Secure sisteminin çalışma prensibi

3D Secure sistemine entegre olmuş işletmeden alışveriş yapan kullanıcı alışveriş işleminde ödeme aşamasına geldiğinde 3-D Secure akışı başlar. 3-D Secure/Güvenli Ödeme sayfasına yönlendirildikten sonra işletme, kart sahibinin bankası ile güvenli ve şifreli bir şekilde veri alışverişinde bulunarak OTP (tek kullanımlık şifre) akışını başlatır. OTP akışı ile kart sahibinin telefonuna süre ve doğrulama korumalı tek kullanımlık şifre gönderilir. Kullanıcı, gönderilen bu şifreyi 3-D Secure – Güvenli Ödeme sayfasında onay kodu bölümüne yazarak doğrulamalar sonucunda ödemeyi güvenli bir şekilde tamamlamış olur. Kullanıcı 3 denemeden fazla yanlış doğrulama kodu girerse veya kod girmesi için gereken süre aşılsa OTP akışı sonlanır ve işleme

devam edilemez. Ödeme işleminin tekrar yapılması gerekir. Buna bağlı olarak herhangi bir çalıntı veya kart sahibinden izinsiz olarak yapılacak ödeme durumunun önüne geçilir.

1.1.2. 3-D Secure sisteminin çalışma aşamaları

- Kart sahibinin kredi/banka kart bilgileri girmesi
- Bilgileri alınan kartın 3-D Secure sistemine entegre olup olmadığının kontrol edilmesi
- Kullanıcının 3-D Secure /Güvenli Ödeme sayfasına yönlendirilmesi
- Sisteme kayıtlı olan kullanıcının telefonuna tek kullanımlık doğrulama kodunun sms ile gönderilmesi
- Gönderilen onay kodunun doğru bir şekilde girilip doğrulanması
- Doğrulan onay kodu ile birlikte ödeme işleminin başarılı bir şekilde onaylanması

1.1.3. 3-D Secure sisteminin avantajları

- 3-D Secure sistemi ile kullanıcıların kart bilgileri şifrelenmiş bir şekilde aktarılır.
- 3-D Secure sistemi ile mevcut kart bilgileri güvenli ve korumalı bir şekilde saklanır.
- Doğrulama kodu kullanılarak kart sahibinin bilgisi dahilinde bir ödeme işlemi gerçekleşir.
- Doğrudan ödeme almadan önce yapılan doğrulama işlemi ile ödeme sahtekarlığının önüne geçilir.
- 3-D Secure sistemine işletmeler başvuru yaparak kolaylıkla entegre olabilir.

BÖLÜM 2. Ödeme Sisteminde Kullanılacak Teknolojiler

2.1 Spring Boot



Görsel 2.1. Spring Boot Logo

Spring Framework, nesnelerin kendi bağımlılıklarını tanımlamasına izin veren ve Spring konteynerinin daha sonra onlara enjekte ettiği bir *bağımlılık enjeksiyon özelliği sunar*. Bu, geliştiricilerin mikro hizmetler ve dağıtılmış ağ uygulamaları için ideal olan gevşek bağlı bileşenlerden oluşan modüler uygulamalar oluşturmasını sağlar. Spring Framework ayrıca veri bağlama, tür dönüştürme, doğrulama, istisna işleme, kaynak ve olay yönetimi ve daha fazlası gibi bir uygulamanın gerçekleştirmesi gereken tipik görevler için yerleşik destek sunar. RMI (Uzaktan Yöntem Çağırma), AMQP (Gelişmiş Mesaj Sıralama Protokolü), Java Web Hizmetleri ve diğerleri gibi çeşitli Java EE teknolojileri ile bütünleşir. Özetle, Spring Framework, geliştiricilere, herhangi bir ortamda çalışan, gevşek bağlı, platformlar arası Java EE uygulamaları oluşturma ihtiyacı olan tüm araçları ve özellikleri sağlar.[1]

Uygulamamızda Spring Boot ile ödeme sistemine entegre olan uygulamaların ve ödeme sistemi uygulamasının arka yüz servislerinin yazılması ve uygulama ön yüzler ile haberleşmesi için Rest API mimarisi kullanımı amaçlanmaktadır. Ödeme uygulamasının Spring Boot ile yazılan servislerinin güvenliği ise OAuth2 Spring Boot Security yaklaşımı kullanılarak tasarlanacaktır. Bu uygulama monolitik yapıda olmayıp aksine Spring Boot üzerinde mikro servis yaklaşımı kullanılarak bir yapı kurulacak ve mikro servislerin orkestrasyonu ise Eureka tarafından yapılacaktır. Servisler Eureka 'ya kaydolacak ve birbirleri arası iletişimlerini Eureka 'nın sağlamış olduğu hizmet ile gerçekleştirilecektir. Uygulama üzerinde ilişkisel tabloları eşleştirmek için JPA standartı ve veri tabanı işlemlerini yönetmek için bir ORM aracı olan Hibernate kullanılacaktır.

2.2 Android/Kotlin



Görsel 2.2. Kotlin Logo

Kotlin, statik olarak Apache 2.0 lisansı altında geliştirilmiş ücretsiz, açık kaynak koda sahip, nesne yönelimli, Java ‘ya nazaran daha kısa ve kendine has kodlama hizmeti veren, Java ile birlikte uyumlu çalışabilen, güvenli bir programlama dilidir. Ödeme sistemi ve bu sisteme entegre olacak sistemlerin mobil uygulamaları Android işletim sistemi üzerinde Kotlin dili ile Android Studio platformunda yazılması tasarlanmaktadır. Bu mobil uygulamalar ön yüz olarak geçmekte ve sistemlerin arka yüzleri arasında Rest API mimarisi ile veri alışverişinde bulunacaktır.

2.3 Apache Kafka



Görsel 2.3. Apache Kafka Logo

Kafka dağıtık bir ‘Message Broker’ sistemidir. Yani bir mesajı alır bir yere depolar ve onu okuyacak olanlarda bu mesajları okur. LinkedIn tarafından oluşturulmuştur. Scala ve Java ile yazılmıştır. İçerisindeki mesajlar INDEX yapısında göre yazılır ve okunur. Bundan dolayı oldukça hızlı bir sistemdir. Sistemde Kafka, asenkron işlerin gerçekleştiriminde kullanılması planlanmaktadır.

Bu işlere örnek olarak kullanıcının ödeme yapma aşamasında telefonuna gelen doğrulama kodunun atılması, kullanıcının ödeme yaptığında kullanıcıya e-posta yoluyla bilgilendirme mesajı gönderilmesi veya şifremi unuttum aşamasında şifre güncelleme işi için e-posta yoluyla şifre güncelleme linki gönderimi gibi asenkron iş örnekleri verilebilir.

2.4 PostgreSQL



Görsel 2.4. PostgreSQL Logo

PostgreSQL, gelişmiş, kurumsal sınıf ve açık kaynaklı bir ilişkisel veritabanı sistemidir. PostgreSQL, hem SQL (ilişkisel) hem de JSON (ilişkisel olmayan) sorgulamayı destekler. PostgreSQL, açık kaynak topluluğu tarafından 20 yılı aşkın bir süredir geliştirilerek desteklenen oldukça kararlı bir veri tabanıdır. PostgreSQL, birçok web uygulamasının yanı sıra mobil ve analitik uygulamaları için birincil veri tabanı olarak kullanılır. PostgreSQL projesi 1986 yılında Kaliforniya Üniversitesi Berkeley Bilgisayar Bilimleri Bölümü'nde başlamıştır. Proje orijinal olarak, yine Berkeley'de geliştirilen eski Ingres veri tabanına atıfta bulunularak POSTGRES olarak adlandırıldı. POSTGRES projesinin amacı, çoklu veri türlerini desteklemek için gereken minimum özellikleri eklemektir. 1996'da POSTGRES projesi, SQL desteğini açıkça göstermek için PostgreSQL olarak yeniden adlandırıldı. Bugün, PostgreSQL genellikle Postgres olarak kısaltılır. O zamandan beri, özel bir katkıda bulunanlar topluluğu olan PostgreSQL Küresel Geliştirme Grubu, açık kaynaklı ve ücretsiz veri tabanı projesinin sürümlerini çıkarmaya devam ediyor. Başlangıçta PostgreSQL, UNIX benzeri platformlarda çalışacak şekilde tasarlanmıştır. Ardından PostgreSQL, Windows, macOS ve Solaris gibi çeşitli platformlarda çalışacak şekilde geliştirildi.[2]

Sistemimizde PostgreSQL, verilerin kalıcı olarak depolanması, istenilen anda bu bilgilere ulaşılabilmesi ve verilerin gizlenmiş bir şekilde saklanması için kullanımı amaçlanmaktadır.

2.5 Docker



Görsel 2.5 Docker Logo

Docker, açık kaynaklı bir 'container' teknolojisidir. Docker, aynı işletim sistemi üzerinde birden fazla, bağımsız ve birbirinden izole konteynırlar sayesinde sanallaştırma sağlayan bir teknolojidir. Uygulamaların kolayca kurulumunu, yatay ölçeklenmiş bir şekilde testini, çalışmasını ve dağıtımını sağlar. Ayrıca Docker sunucu maliyetlerini önemli ölçüde azaltır. Docker, sistemimizde uygulamaların yatay ölçeklenmiş bir şekilde test edilmesini sağlamak, donanım maliyetinden kaçmak ve uygulamalarımızı dockerize edebilmek için kullanımı amaçlanmaktadır.

BÖLÜM 3. Güvenli Ödeme Sistemi

SauPay kullanıcıların hesap oluşturabilecekleri, hesaplarına giriş ve çıkış yapabilecekleri, kredi kartlarını hesaplarına entegre edebilecekleri, entegre olan kredi kartları ile ödeme yapabilecekleri bir çevrim içi ödeme sistemi hizmeti sunan ödeme sistemidir. SauPay uygulamasına kullanıcılar farklı bankalara ait kartlarını entegre edebilir ve bu kartlar ile ödeme yapabilirler. SauPay uygulaması ile kullanıcılar, bu sisteme entegre olan işletmelerden ödeme yapacakları zaman “SauPay ile öde” seçeneği seçilir ve sistem, arka tarafta 3D Secure kapsamı altında işletme, banka ve kullanıcı kartı arasında güvenli bir bilgi akışıyla birlikte ödeme işlemini başarıyla gerçekleştirir.

3.1 SauPay ile Güvenli Ödeme İşlemi

SauPay ile güvenli ödeme işlemi 3-D Secure prensiplerine uyarak ödeme işlemlerini gerçekleştirmektedir. Güvenliği sağlayan en temel iki unsur verilerin şifrelenmesi ve doğrulama yapılmasıdır. SauPay ödeme sisteminde işletme, SauPay sistemine entegre olmak için başvuruda bulunur. Başvurusu onaylandığı takdirde SauPay, işletmeye ait işletme id si verir ve bu id ödeme veri tabanında işletme bölümünde kaydedilir. İşletme artık her “SauPay ile Öde” isteği attığında bu id ile istek atar. Bunun yanı sıra işletme ödeme isteğini atacağı zaman verilerini string bir ifade de toplar ve SHA algoritması kullanarak imzalar. Ödeme bu imzalanan veriyi ödeme arka yüz servisinde doğrulayarak işlemlerine devam eder. Eğer imza doğrulama sağlanmazsa veri bütünlüğü bozulmuş olur ve ödeme işlemi bu akışta kesilir. Onaylanan imza sürecinin ardından işletmeden gelen ödeme verilerine göre şifrelenmiş bir ödeme token’ı oluşturulur ve oluşturulan token veri tabanında işletme bölümünün işlemleri bölümüne kaydedilir. Token bu sefer ödeme tarafından imzalanarak işletmeye gönderilir ve işletme, imzalı token’ı doğrular. Doğrulama başarılı olursa token, ödeme mobil uygulamasına gönderilir. Kullanıcı ödeme uygulamasına kayıtlı ise giriş yapar değilse kaydolar ve akabinde giriş yapar. Başarılı giriş işleminin ardından ödeme uygulamasının almış olduğu token, ödeme servisi tarafından böyle bir token olup olmadığı veri tabanından kontrol edilir ve doğrulanır. Doğrulama işleminin sonucunda ödeme uygulamasına doğrulama başarılı bilgisi dönülür ve kullanıcının kaydetmiş olduğu kartlardan biri kullanıcı tarafından seçilir ve

ödeme işlemine geçileceği zaman ödeme uygulaması, servisine ödeme yapılacak olan kartın bankası ile iletişime geçerek ödeme işlemini gerçekleştirmesi gerektiğini söyler. Başarılı bir iletişim sürecinden geçen ödeme akışı sonucunda bankadan ödeme alınır ve işletme aracılığıyla kullanıcıya işlemin başarılı bir şekilde gerçekleştirildiği bilgisi verilir. Böylece çevrim içi güvenli ödeme uygulamasının ödeme akışı tamamlanmış olur.

3.2 Güvenli Ödeme İşleminde İmzalama ve Şifreleme

Veri gönderecek olan client ödeme isteğini atacağı zaman verilerini string bir ifade de toplar ve SHA-256 algoritması kullanarak imzalar. Gönderilecek olan veri ise body’de şifrelenerek gönderilir. İsteği karşılayan uygulama öncelikle verinin şifresini çözer ve daha sonra imzalanan



veriyi doğrulayarak işlemlerine devam eder. Eğer imza doğrulama sağlanmazsa veri bütünlüğü bozulmuş olur ve ödeme işlemi bu akışta kesilir.

Görsel 3.2 Verilerin Şifrenmesi, İmzalanması ve Doğrulanması

3.2.1 Android Tarafında Güvenli Ödeme İşleminde İmzalama ve Şifreleme

```
@Throws(Exception::class)
fun generateSignature(randomKey: String, body: String): String {
    val concatenatedString = SECRET_KEY_BACKEND + randomKey + body
    Log.d( tag: "body", body)
    val digest = MessageDigest.getInstance( algorithm: "SHA256")
    val encodedHash = digest.digest(concatenatedString.toByteArray());
    return encodeToString(encodedHash, DEFAULT).trim().uppercase(Locale.ENGLISH);
}

@Throws(Exception::class)
fun isSignatureValid(randomKey: String, body: String, signature: String): Boolean {
    val generatedSignature = generateSignature(randomKey, body)
    return generatedSignature == signature
}

@Throws(Exception::class)
fun encrypt(plainText: String): String {
    val secretKey = SecretKeySpec(SECRET_KEY_BACKEND.toByteArray(), algorithm: "AES")
    val cipher = Cipher.getInstance( transformation: "AES")
    cipher.init(Cipher.ENCRYPT_MODE, secretKey)
    val encryptedBytes = cipher.doFinal(plainText.toByteArray(charset( charsetName: "UTF-8")))
    return encodeToString(encryptedBytes, NO_WRAP)
}

@Throws(Exception::class)
fun decrypt(encryptedText: String): String {
    val secretKey = SecretKeySpec(SECRET_KEY_BACKEND.toByteArray(), algorithm: "AES")
    val cipher = Cipher.getInstance( transformation: "AES")
    cipher.init(Cipher.DECRYPT_MODE, secretKey)
    val decryptedBytes = cipher.doFinal(decode(encryptedText, NO_WRAP))
    return String(decryptedBytes)
}
```

Görsel 3.2.1 Android Tarafında Verilerin Şifrenmesi, İmzalanması ve Doğrulanması

3.2.2 Backend Tarafında Güvenli Ödeme İşleminde İmzalama ve Şifreleme

```

@Component
public class EncryptionUtil {

    1 usage  ▲ OMER FARUK GUZEL
    public String encrypt(String data,String secret_Key){
        try {
            SecretKeySpec secretKey = new SecretKeySpec(secret_Key.getBytes(), "AES");
            Cipher cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            byte[] encryptedBytes = cipher.doFinal(data.getBytes());
            return Base64Utils.encodeToString(encryptedBytes);
        } catch (Exception e) {
            e.printStackTrace();
            return "Decrypting Error";
        }
    }

    2 usages  ▲ OMER FARUK GUZEL
    public String decrypt(String encryptedData,String secret_Key){
        try {
            SecretKeySpec secretKey = new SecretKeySpec(secret_Key.getBytes(), "AES");
            Cipher cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            byte[] encryptedBytes = Base64Utils.decodeFromString(encryptedData);
            byte[] decryptedBytes = cipher.doFinal(encryptedBytes);
            return new String(decryptedBytes);
        } catch (Exception e) {
            e.printStackTrace();
            return "Decrypting Error";
        }
    }

    ▲ OMER FARUK GUZEL
    public String signature( String randomKey, String encryptedData,String secretKey) {
        try {
            String concatenatedString = secretKey + randomKey + encryptedData;
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] encodedHash = digest.digest(concatenatedString.getBytes(StandardCharsets.UTF_8));
            String controlSignature= Base64Utils.encodeToString(encodedHash).trim().toUpperCase(Locale.ENGLISH);
            return controlSignature;
        } catch (Exception e) {
            e.printStackTrace();
            return "Signature Error";
        }
    }

    2 usages  ▲ OMER FARUK GUZEL
    public Boolean checkSignature(String signature,String randomKey,String encryptedPaymentRequest,String secretKey){

        System.out.println("hepsi: " + signature + " " + randomKey );
        try {

            String concatenatedString = secretKey + randomKey + encryptedPaymentRequest;
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] encodedHash = digest.digest(concatenatedString.getBytes(StandardCharsets.UTF_8));
            String controlSignature= Base64Utils.encodeToString(encodedHash).trim().toUpperCase(Locale.ENGLISH);
            return Objects.equals(signature, controlSignature);

        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
}

```

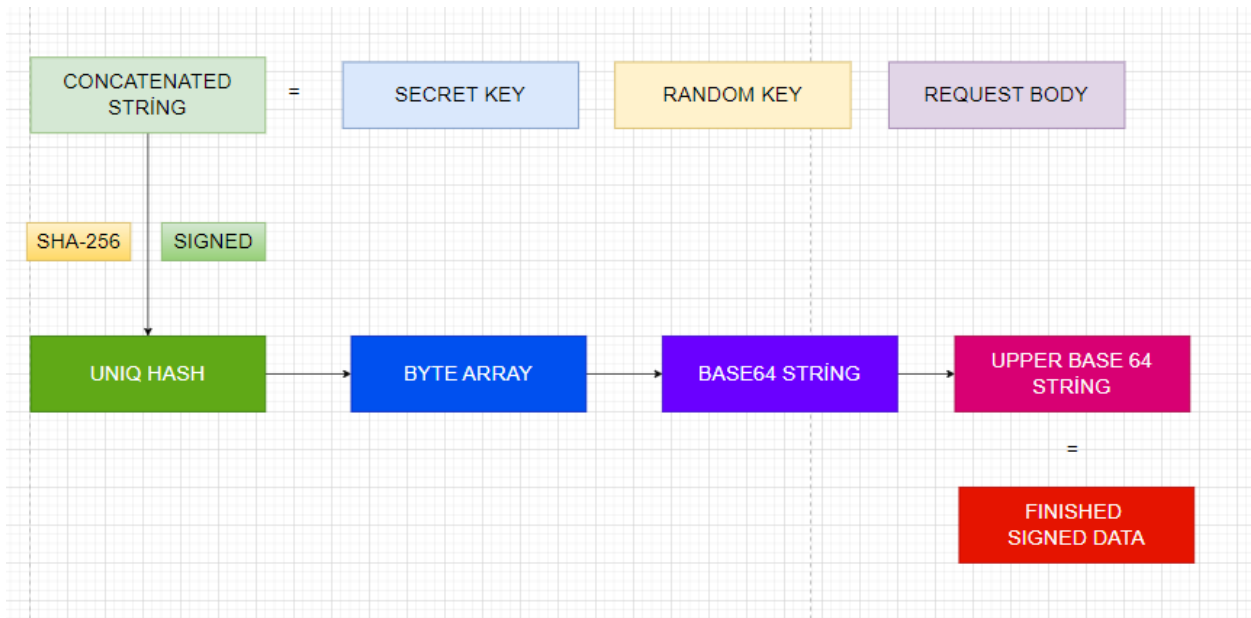
3.2.3 SHA-256 Algoritması

SHA-256, Güvenli Karma Algoritması 256 bit sürümüdür. SHA, verileri karma işlemiyle işleyerek sabit uzunlukta bir özet üreten bir kriptografik karma işlemidir. SHA-256, özellikle dijital imzalama, kimlik doğrulama ve veri bütünlüğü sağlama gibi güvenlik uygulamalarında yaygın olarak kullanılan bir algoritmadır. SHA-256'nın imzalama kullanımı, bir belgenin veya veri kümesinin bütünlüğünü doğrulamak veya belgenin hangi kişi veya kuruluş tarafından oluşturulduğunu doğrulamak için kullanılır. İşlem genellikle aşağıdaki adımları içerir: Belgeyi hazırlama: İmzalanacak belge veya veri kümesi oluşturulur. Hash değerinin hesaplanması: SHA-256 algoritması kullanılarak belgenin bir özeti veya karma değeri hesaplanır. Bu, belgenin benzersiz bir sayısal özeti olacaktır. Özeti imzalanması: İmzalama sürecinde, özet, bir özel anahtar kullanılarak dijital olarak imzalanır. Bu özel anahtar, belgeyi imzalayan kişiye özgüdür ve genellikle bir kriptografik anahtar çiftin parçası olan özel bir anahtardır. Bu işlem, belgeyi dijital olarak imzalar ve kimlik doğrulama için kullanılan bir sayısal imza oluşturur. İmzanın doğrulanması: İmzalanan belgeye erişen kişi veya sistem, aynı SHA-256 algoritmasını kullanarak belgenin özetini hesaplar. Daha sonra, imza doğrulama süreciyle bu özet doğrular ve belgenin bütünlüğünü ve kimin tarafından imzalandığını doğrular. İmza, ilgili genel anahtar kullanılarak doğrulanır. SHA-256, kriptografik olarak güçlü ve matematiksel olarak dayanıklı bir algoritmadır. Bilgi güvenliği açısından önemli bir rol oynar ve dijital imzalama gibi güvenlik uygulamalarında yaygın olarak kullanılır.

3.2.4 Güvenli Ödemede Verilerin SHA-256 Algoritması ile İmzalanması

İlk olarak, üç parametre alınır: random key, encrypted data ve secret key. Bu parametreler işlemin yapılacağı verileri temsil eder. Concatenated string adında bir dize oluşturulur. Bu dize, secret key, random key ve encrypted data değerlerini birleştirir. Message digest sınıfı kullanılarak SHA-256 algoritması belirtilen ve bir digest örneği oluşturulur. Digest yöntemi, concatenated string değerini SHA-256 algoritmasıyla işleyerek bir özet (hash) üretir. Sonuç, bir byte dizisinde depolanır. Base64Utils.encodeToString() yöntemi kullanılarak encoded hash değeri Base64 formatına

kodlanır. Bu, özetin bir dizeye dönüştürülmesini sağlar. Elde edilen kontrol imzası dizesi büyük harflere dönüştürülür ve gereksiz boşluklardan arındırılır. Bu kontrol imzası, veri bütünlüğünü doğrulamak veya verinin kaynağını doğrulamak için kullanılabilir. İşletmeler ve Ödeme uygulaması bu verilen string değerlerinin belirli şekilde bir araya gelmesi kuralı ile bir hash üretir ve bu hash ile gönderilecek veriler imzalanır. Uygulama backendi ise gelen imzalı veriyi bu sıralam kuralını ve secret key değerini bildiği için imzalamayı çözer ve gelen veri imzası başarılı bir şekilde çözülürse verinin bozulmadığı anlaşılır. Ardından işlem akışı devam eder.



Görsel 3.2.4 Verilerin SHA-256 Algoritması ile İmzalanması

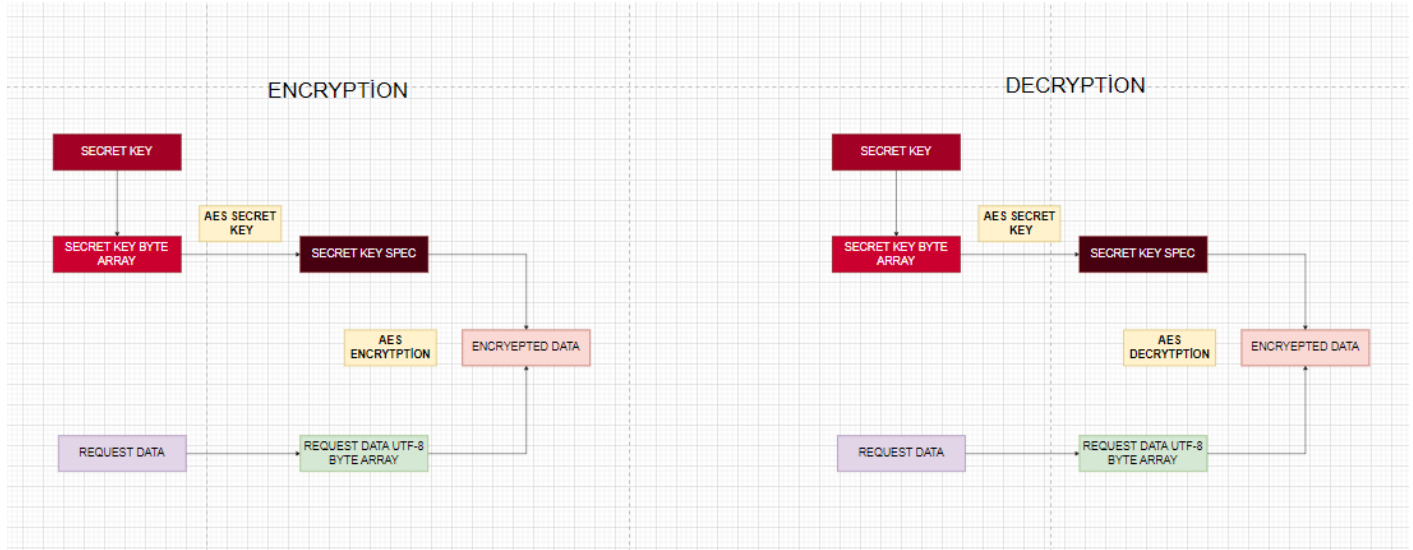
3.2.5 AES Algoritması

AES (Advanced Encryption Standard), güçlü ve yaygın olarak kullanılan bir blok şifreleme algoritmasıdır. AES, 128 bit, 192 bit veya 256 bit anahtar boyutlarıyla çalışabilen simetrik bir şifreleme algoritmasıdır. Simetrik bir algoritma olduğu için aynı anahtar, veriyi şifrelemek ve şifreyi çözmek için kullanılır. AES'in çalışma prensibi şu adımları içerir: Anahtar Oluşturma: AES için kullanılacak anahtar, belirlenen boyuta göre (128 bit, 192 bit veya 256 bit) rastgele bir şekilde oluşturulur. Anahtarın güvenliği, şifreleme işleminin güvenliği için kritik öneme sahiptir. Blok Şifreleme: AES, veriyi belirli boyutlarda bloklar halinde işler. Standart olarak, bir AES bloğu 128 bit (16 byte) boyutundadır. Şifrelenecek veri, bloklar halinde ayarlanır ve her bir blok ayrı ayrı

işlenir. Yöntem Seçimi: AES, veriyi şifrelemek için farklı şifreleme modları sunar. Bunlar arasında ECB (Electronic Codebook), CBC (Cipher Block Chaining), CTR (Counter), OFB (Output Feedback) ve daha fazlası bulunur. Her modun kendine özgü özellikleri ve kullanım senaryoları vardır. Yer Değiştirme İşlemi (Substitution): Şifreleme işlemi sırasında, bloklar içindeki her byte, belirli bir S-box tablosu kullanılarak yer değiştirme işlemine tabi tutulur. Bu işlem, şifrelemenin gücünü artıran bir karmaşıklık katmanıdır. Karıştırma İşlemi (Permutation): Yer değiştirme işleminden sonra, bloklar içindeki byte'lar karıştırılır. Bu adım, şifrelemenin difüzyon özelliğini sağlar ve verinin istatistiksel olarak düzensiz hale gelmesini sağlar. Anahtar Ekleme İşlemi (Key Addition): Her blok şifrelemesi öncesi, anahtar ile blok arasında bir XOR işlemi gerçekleştirilir. Bu işlem, güvenlik ve anahtar bağımlılığını sağlar. Şifreleme Modları ve Şifre Çözme: Seçilen şifreleme moduna göre, blok şifreleme işlemi tekrarlanır ve ardışık bloklar arasında farklı bağımlılıklar sağlanır. Şifre çözme işlemi ise şifreleme işleminin tersini uygular ve verinin orijinal hâline geri dönmesini sağlar.

3.2.6 Güvenli Ödemede Verilerin AES Algoritması ile Şifrenmesi

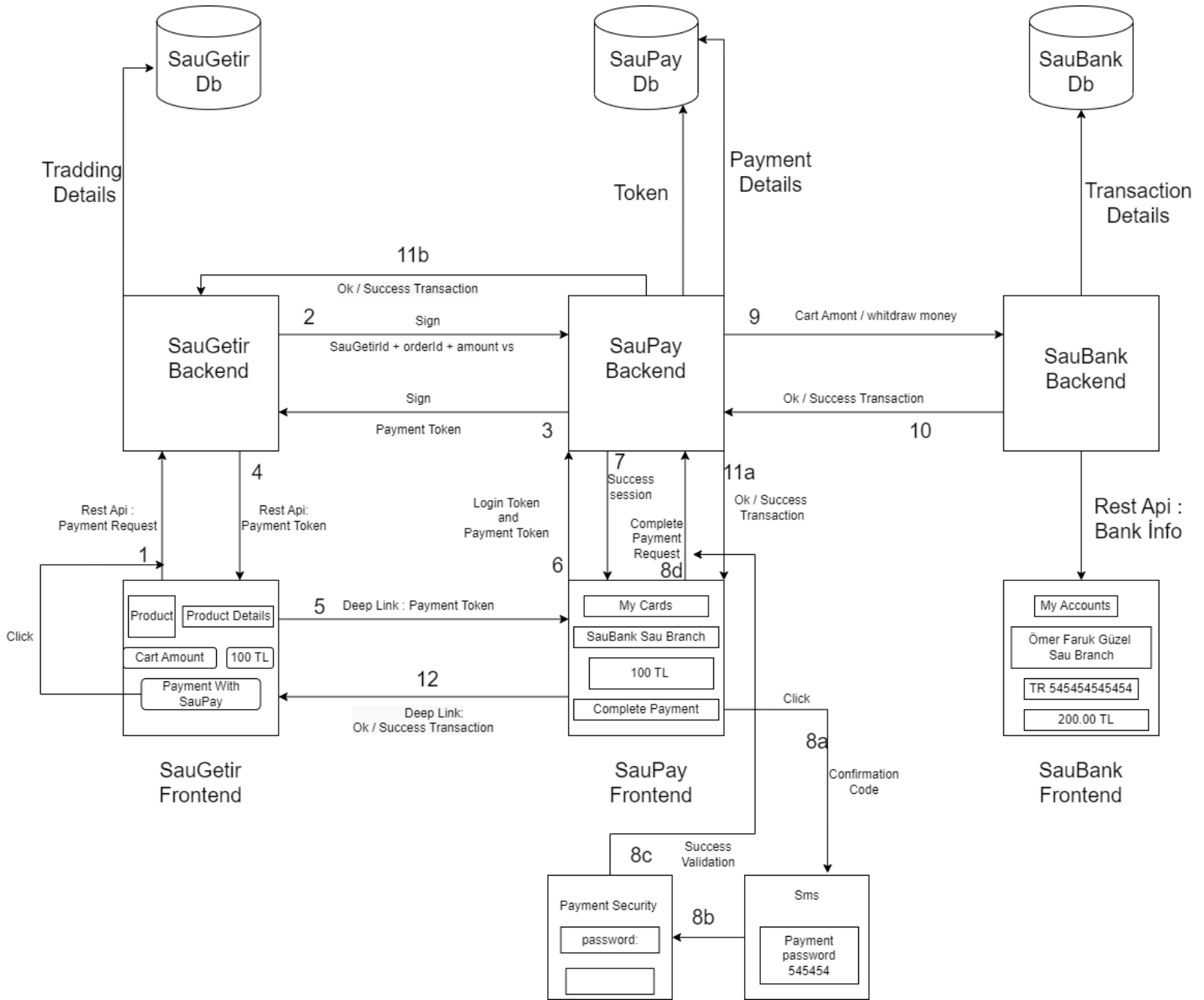
Backend veya Client arasında gönderilecek olan veriler şifrenir. Api isteğinin body kısmında (Request Body) gönderilecek olan veriler iki tarafta da bulunan secret key ile birlikte AES algoritması kullanılarak şifrenmekte ve istekler herhangi bir saldırıya karşı güvenli hale getirilmektedir. Buna bağlı olarak şifreli olan veriler çözülmeden imzada çözülemeyecektir. Çünkü imzalanan veride şifrenmeyen body verisi bulunmaktadır. Bundan dolayı imzanın çözülmesi için öncelikle şifrelenen veri çözülmelidir. Akabinde çözülen veri imza doğrulanmasında kullanılmaktadır.



Görsel 3.2.6. Verilerin AES ile Şifrelenmesi

3.3 Ödeme Sistemi ile Bağlantılı Uygulamalar Arasındaki Akış

Ödeme sisteminin test edilebilmesi için 3 farklı işleyişte olan SauPay, SauGetir ve SauBank mobil uygulamaları tasarlanmıştır. Bu uygulamalar arasındaki bağlantılar, işlem sırasına göre ödeme işleminin gerçekleşebilmesi için birbirleri arasında veri alışverişinde bulunabildikleri akış diyagramı tasarlanmıştır. Akış diyagramında belirtilen numaralar işlem sırasını, ok yönleri ise o işlemten sonra hangi işleme gideceğini belirtir. Akış içerisinde SauGetir uygulaması, sisteme kayıtlı olduğu varsayılır ve bu işletme uygulamasında alışveriş işlemi sepet aşamasını tamamlamış, ödeme aşamasına geldiği zaman SauPay ile öde seçeneğinin seçildiği andan itibaren başlayan ve başarılı ödeme işleminin gerçekleştiği bir akış şematize edilmiştir. Başarılı ödeme sonucunda ise işletme uygulamasında kullanıcıya başarılı ödeme işlemi bilgisi dönülür.



Görsel 3.3 Uygulamalar Arasındaki Bağlantılı Akış Diyagramı

3.3.1. Akış Diyagramı İşlem Açıklamaları

- İşlem 1: Kullanıcı SauGetir işletmesinden alışveriş yapacağı malzemeleri sepetine ekler ve ödeme aşamasında “SauPay ile öde” seçeneğini seçer. Daha sonra bu ödeme isteği doğrultusunda işletmeden yapılan alışveriş sepetinin bilgileri ve işletme bilgileri rest api ile işletme servisine iletilir (Önyüz–Arka yüz haberleşmesi).
- İşlem 2: İşletme uygulamasından gelen ödeme detayı ve işletme bilgileri string bir ifadede toplanarak işletmeye ait gizli anahtar kullanılarak dijital imza ile imzalanır ve rest api ile SauPay Ödeme arka yüz servisine iletilir (Arka yüz – Arka yüz haberleşmesi).
- İşlem 3: Ödeme arka yüz servisi imzalanmış verileri işyerinden almış olduğu genel ahatar ile doğrular ve token oluşturma akışına devam eder. Daha sorna oluşturmuş olduğu bu token’ ı kendi gizli anahtarı ile imzalar ve ve bu ödeme token’ını işyeri arka yüz servisine iletir. (Arka yüz – Arka yüz haberleşmesi)
- İşlem 4: İşyeri arka yüz servisi almış olduğu imzalı token’ı kendisinin de bildiği ödeme genel anahtarını kullanarak doğrular ve işyeri arka yüz servisi token ’ı işletme önyüzüne iletir. (Arka yüz – Önyüz haberleşmesi)
- İşlem 5: İşyeri önyüzü, ödeme uygulamasının oluşturduğu o anki ödeme işlemine ait ödeme token’ ını Deep Link ile ödeme önyüzüne iletir. (Önyüz – Önyüz haberleşmesi)
- İşlem 6: Ödeme önyüzü ne gelen bu işlemden sonra kullanıcının oturum açık değilse sisteme giriş yapması istenir. Kullanıcı sistemde kayıtlı değilse sisteme kaydolar ve öyle giriş yapar. Kullanıcı giriş istekleri için OAuth2 güvenliği ile oluşturulan token’ ı istek atarken isteğin “header” bölümüne ekler ve istek üzerinde işyerinden Deep Link ile alınan token gönderilir (Önyüz – Arka yüz haberleşmesi).
- İşlem 7: Başarılı giriş token’ı ve ödeme token’ı onaylama işleminin ardından ödeme arka yüzü, başarılı doğrulama işlemini döner ve oturum açılır (Arka yüz – Önyüz haberleşmesi).

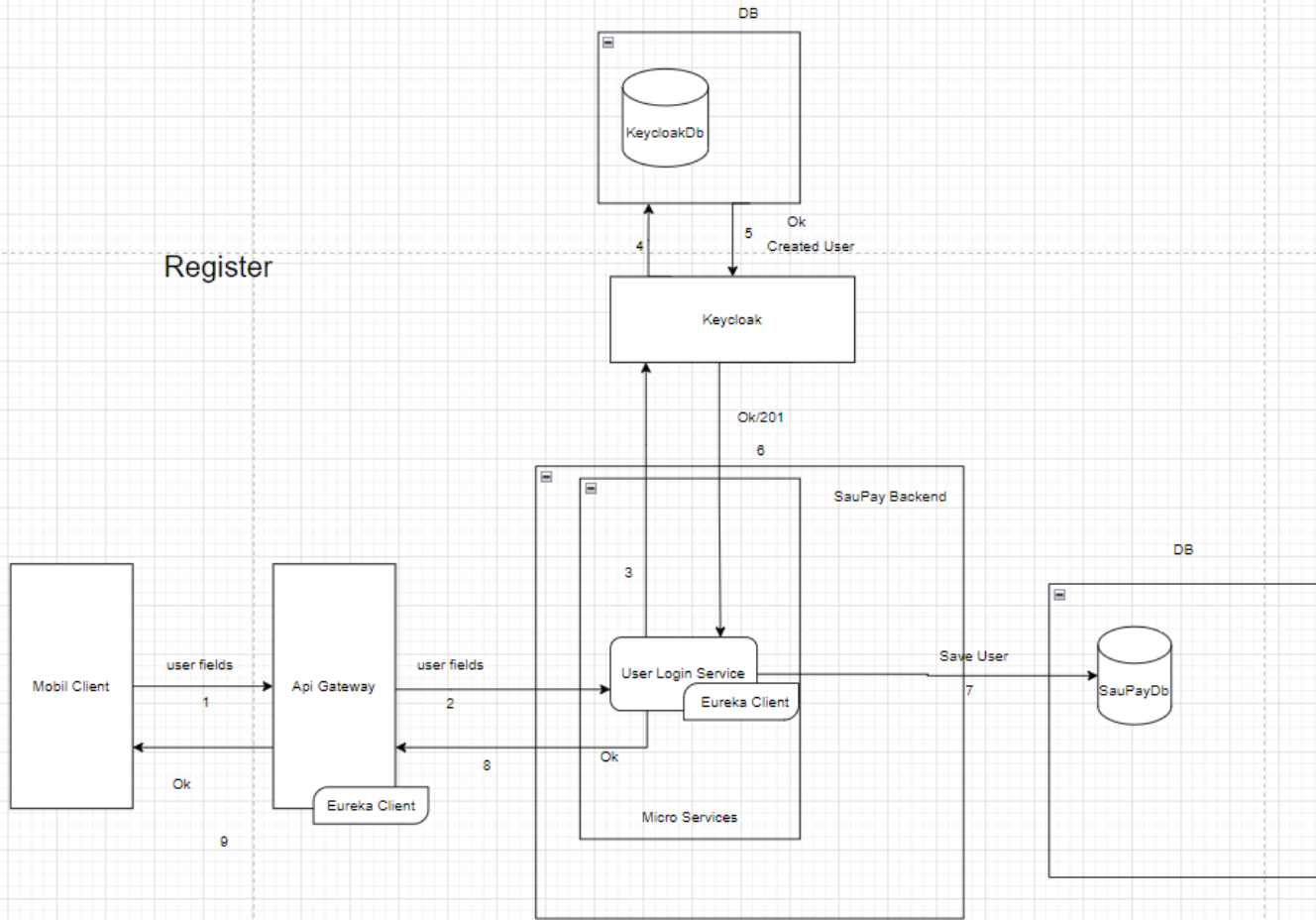
- İşlem 8a: Açılan oturumla birlikte kullanıcı ödeme işlemini gerçekleştirecek banka kartını seçer (Banka kartı önceden kayıtlı varsayılarak grafik çizilmiştir). Ardından ödeme isteği attığı vakit banka kartı bilgisi, hesap tutarı vs. bilgileri ile birlikte OTP (Tek kullanımlık şifre) akışı başlar. Kullanıcının telefonuna tek kullanımlık süre güvenli doğrulama kodu gönderilir.
- İşlem 8b: Sms ile gönderilen onay kodu, Güvenli Ödeme / 3D Secure sayfasında onay kodu bölümüne yazılır.
- İşlem 8c: Doğrulama işlemi doğrulanırsa başarılı doğrulama bilgisi 8. Adıma döner.
- İşlem 8d: Tek kullanımlık şifre ile kullanıcıdan doğrulama bilgisi alınan adımdan gelen başarılı doğrulama ile ödeme detayları banka ile haberleşmek için ödeme arka yüzüne aktarılır (Önyüz – Arkay yüz haberleşmesi).
- İşlem 9: Ödeme arka servisi bu istek üzerine banka arka servisi ile haberleşerek ödeme önyüzünden aldığı ödeme bilgileriyle birlikte bankaya ödeme hesap tutarı kadar bankadan kullanıcı hesabından para çekilmesini talep eder.
- İşlem 10: Kullanıcının banka hesabından ödeme tutarı kadar para çekilme işleminin sonucunda başarılı işlem bilgisi ödeme arka servisine iletilir.
- İşlem 11a: Ödeme arka yüzünden başarılı ödeme işlemi bilgisi alan ödeme önyüzü bu

3.3.2. Kullanıcı İşlemlerinin Keycloak ile Gerçekleştirilmesi

Keycloak, uygulamanızın güvenliğini daha kolay hale getirecek açık kaynaklı bir araçtır. Kullanıcı kimlik doğrulama ve erişim yönetimi uygulaması gibi düşünebiliriz. Bu uygulama tipik olarak, ortamınıza erişim talep eden bir kullanıcının veya sistemin kimliğini doğrulamayı amaçlar ve o kullanıcının veya sistemin hangi rollere ve varlıklara erişimi olduğunu söyleyen bir dizi kuralı değerlendirir [3].

3.3.2.1 Register İşleminin Keycloak ile Gerçekleştirilmesi

Keycloak kendi veri tabanı olan kimlik doğrulama, yetkilendirme, oturum yönetme ve güvenlik işlemlerini gerçekleştirebilen açık kaynaklı bir uygulamadır. Saupay uygulamasında kullanıcı, sisteme kaydolma isteği gerçekleştirir ve gerekli bilgileri bu istekle birlikte gönderir. Daha sonra kullanıcı, daha önce kaydolmadıysa keycloak sistemine ve uygulama veri tabanına kaydedilir. Böylece senkron bir kullanıcı kaydı tutulur.



Görsel 3.3.2.1 Regsiter İşleminin Kecloak ile Gerçekleştirilmesi

3.3.2.2 Backend Tarafında Register İşleminin Keycloak ile Gerçekleştirilmesi

1 usage ⇅ OMER FARUK GUZEL

```
public RegisterResponse registerUser(UserRegisterRequest userRequest) {

    List<UserRepresentation> userRepresentations = keycloakUserService.readUserByEmail(userRequest.getUserEmail());
    if (userRepresentations.size() > 0) {
        throw new GeneralException("404","This email already registered as a user. Please check and retry.");
    }
    userRepository.findByUserEmail(userRequest.getUserEmail()).ifPresent(user -> {
        throw new GeneralException("404","This email already registered as a user. Please check and retry.");
    });

    //user attributes set
    UserRepresentation userRepresentation = new UserRepresentation();
    userRepresentation.setEmail(userRequest.getUserEmail());
    userRepresentation.setEmailVerified(false);
    userRepresentation.setEnabled(true);
    userRepresentation.setUsername(userRequest.getUserName());

    //password set
    CredentialRepresentation credentialRepresentation = new CredentialRepresentation();
    credentialRepresentation.setValue(userRequest.getUserPassword());
    credentialRepresentation.setTemporary(false);
    userRepresentation.setCredentials(Collections.singletonList(credentialRepresentation));

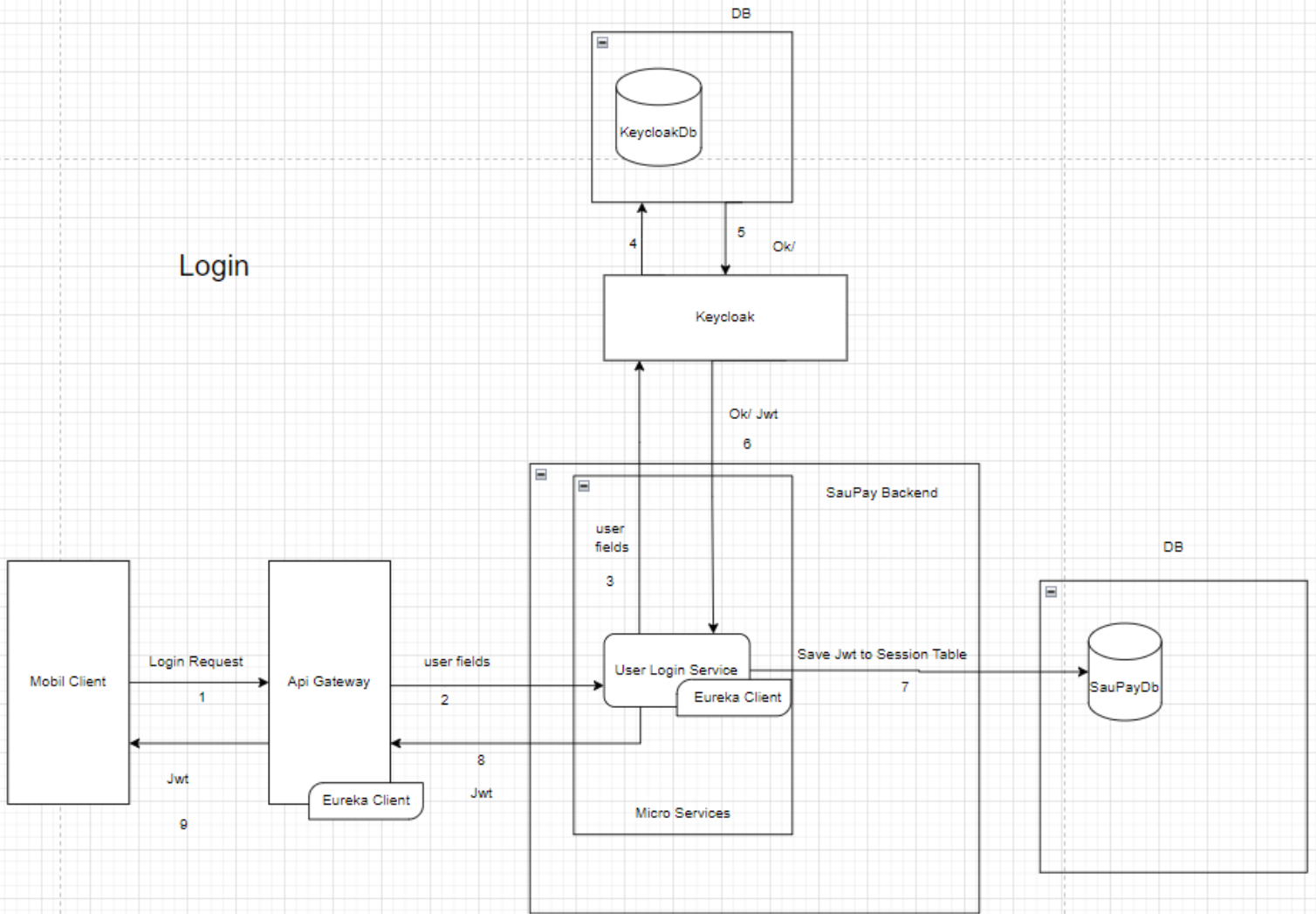
    //user creation Response Code Number
    Integer userCreationResponse = keycloakUserService.createUser(userRepresentation);
    // ekrana logla
    System.out.println("Return Http Code: " +userCreationResponse);
    if (userCreationResponse == 201)//201 means user created successfully
    {
        log.info("User created under given username {}", userRequest.getUserEmail());

        User user = new User( id: "",userRequest.getUserName(),userRequest.getUserSurname(),userRequest.getUserEmail(),userRequest.getUserPassword(),userRequest.getUserPhone(),userRequest.getUserTC(), LocalDateTir
        UserDto userDto = userDtoConverter.convert(userRepository.save(user));
        System.out.println("User created successfully" + " " + userDto.getCustomerName() + " " + userDto.getCustomerSurname() + " " + userDto.getCustomerEmail() + " " + userDto.getCustomerPhone() + " " + userDto
        return new RegisterResponse( userRegistered: true);
    }
    else {
        throw new GeneralException("404","We couldn't find user under given identification. Please check and retry");
    }
}
```

Görsel 3.3.2.2 Backend Tarafında Register İşleminin Keycloak ile Gerçekleştirilmesi

3.3.2.3 Login İşleminin Keycloak ile Gerçekleştirilmesi

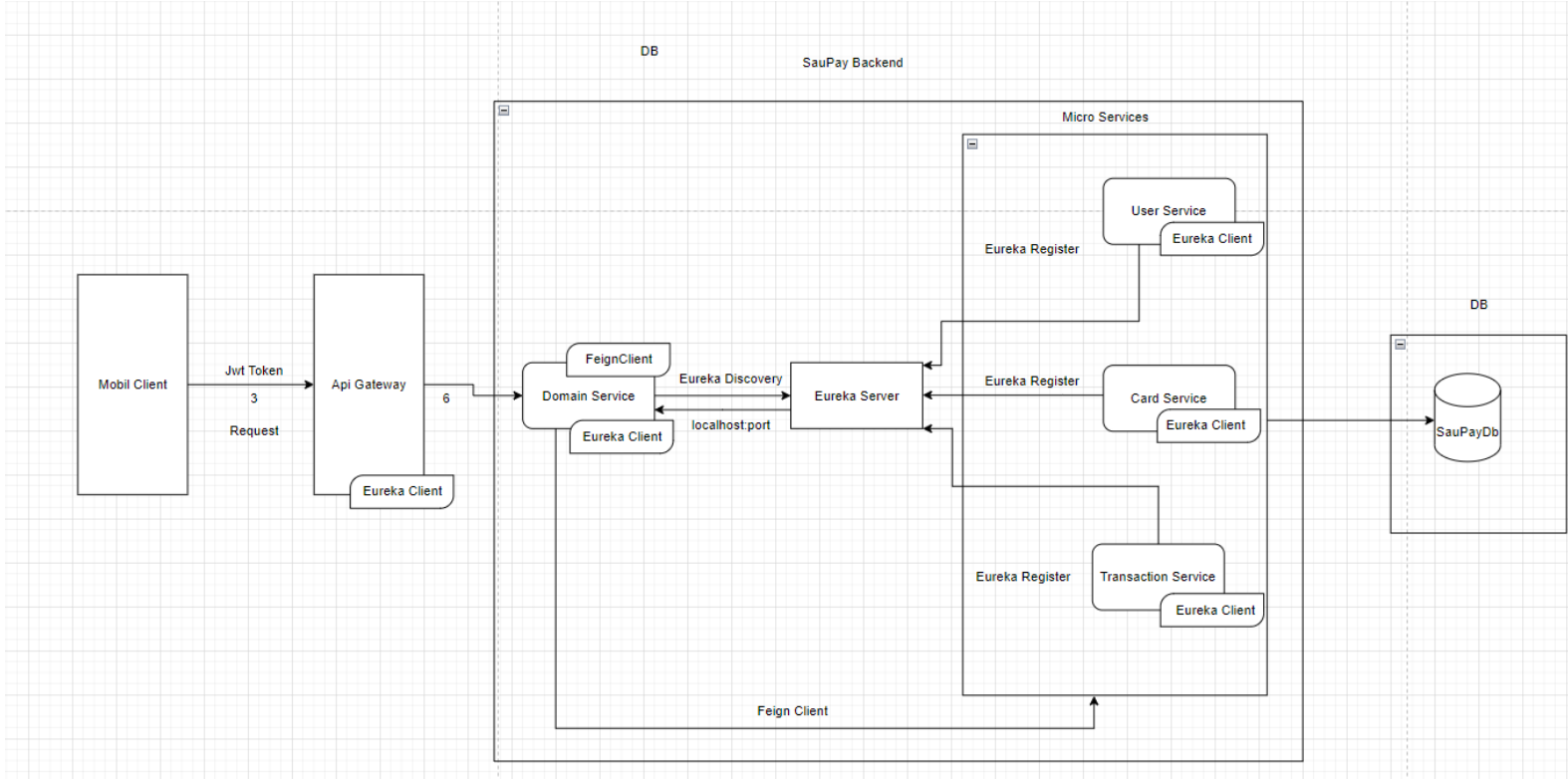
Saupay uygulamasında kullanıcı, sisteme giriş isteği gönderir. Bu istekle beraber gerekli bilgiler alındıktan sonra Saupay, Keycloak ile haberleşip kullanıcı bu sistemine kayıtlı ise Keycloak tarafından bir Jwt token üretilir ve Saupay'e bu token dönülür. Alınan Jwt token, uygulama veri tabanına kaydedilir ve client, 'a dönülür. Client, isteklerinde bu token'ı kullanır ve token geçerli ise başarılı cevaplar alır.



Görsel 3.3.2.3 Login İşleminin Keycloak ile Gerçekleştirilmesi

3.3.2.4 Request İşleminin Keycloak ile Gerçekleştirilmesi

Client'a dönülen Jwt token ile gerekli istekler yapılır ve Keycloak ile haberleşip token geçerliliği kontrol edilir ve isteklere karşı başarılı cevaplar verilir.



Görsel 3.3.2.4 Request İşleminin Kecalok ile Gerçekleştirilmesi

3.3.2.5 Keycloak ile ApiGateway Konfigurasyonu

Saupay uygulamasının backend'inde oluşturulan apigateway katmanında user authentication ve authorization işlemleri yapılmaktadır. Bu işlemler keycloak aracılığı ile gerçekleştirilmektedir.

Keycloak sisteminin bütünleşerek çalışabilmesi için apigateway katmanında konfigürasyonları yapılması gerekir. Keycloak sisteminde saupay uygulamasına özgü alanlar oluşturulur ve saupay uygulamasına özel bağlantılar ile application.yml dosyasında ayarlamalar yapılır.

```

security:
  oauth2:
    client:
      provider:
        keycloak:
          token-uri: ${app.config.keycloak.url}/realms/${app.config.keycloak.realm}/protocol/openid-connect/token
          authorization-uri: ${app.config.keycloak.url}/realms/${app.config.keycloak.realm}/protocol/openid-connect/auth
          user-name-attribute: preferred_username
          user-info-uri: ${app.config.keycloak.url}/realms/${app.config.keycloak.realm}/protocol/openid-connect/userinfo
          jwk-set-uri: ${app.config.keycloak.url}/realms/${app.config.keycloak.realm}/protocol/openid-connect/certs
          user-info-authentication-method: header
# eğer yoksa
# registration:
#   internet-banking-core-client:
#     provider: keycloak
#     client-id: saupay
#     client-secret: hrxlVD8J9Rsyu7pWVhCgrxIKT8vKXNF3
#     authorization-grant-type: authorization_code
#     redirect-uri: http://localhost:8888/login/oauth2/code/keycloak
#     scope: openid
resourceserver:
  jwt:
    jwk-set-uri: ${app.config.keycloak.url}/realms/${app.config.keycloak.realm}/protocol/openid-connect/certs

```

Görsel 3.3.2.5 Keycloak ile ApiGateway konfigürasyonu

3.3.2.6 Keycloak ile User Service konfigürasyonu

Uygulama akışında kullanıcı kayıt işleminde, kullanıcı önce keycloak’a eklenir daha sonra başarılı ekleme işlemi ardından uygulama veri tabanına eklenmektedir. Bundan dolayı keycloak’da ekleme yapabilme yeteneğini kazanmak için secret id’yi konfigürasyona eklemeliyiz.

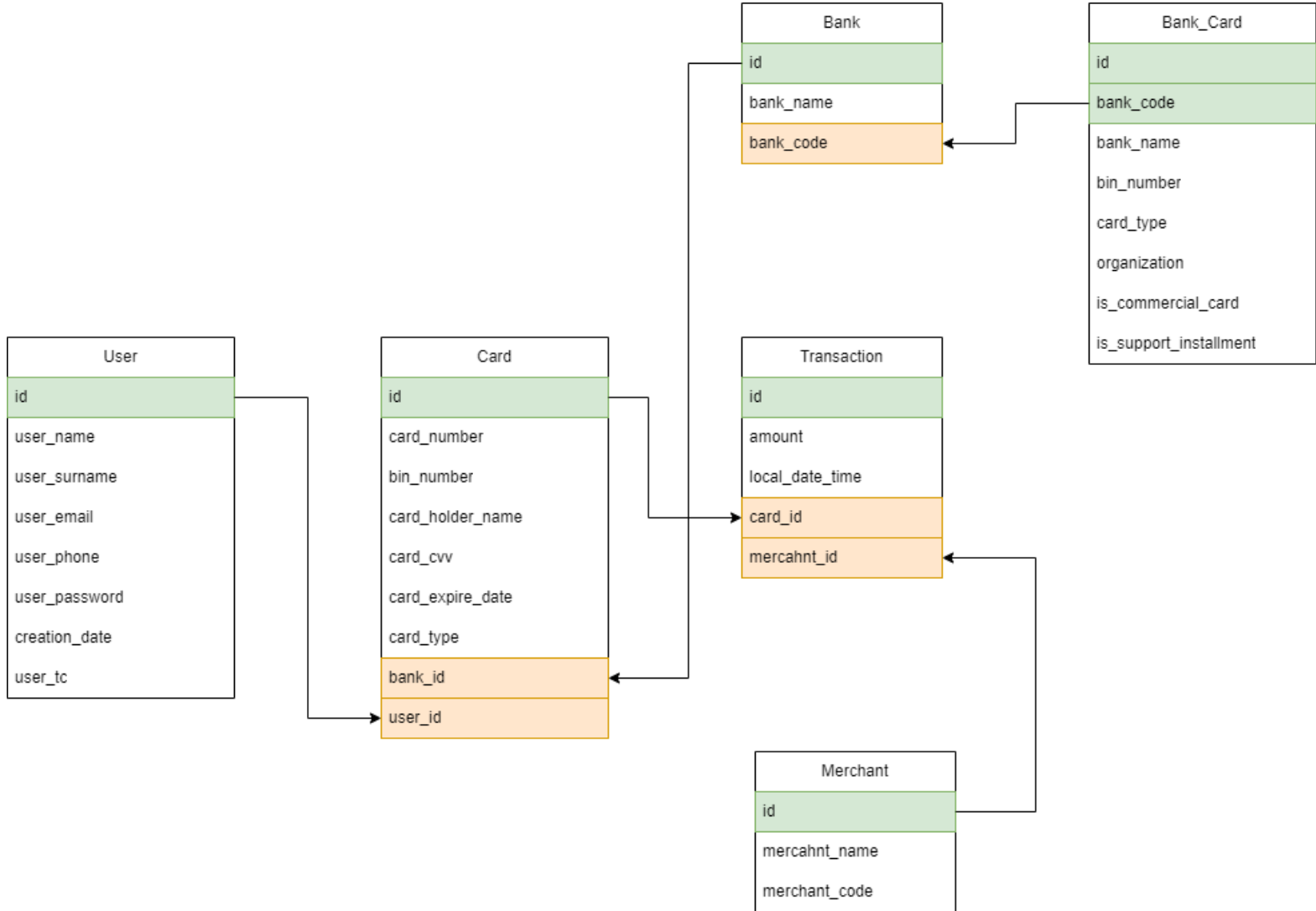
```

#Keycloak Admin Client ==> Keycloak Admin Client ile Keycloak Server'a bağlanılır ve Keycloak Server'da kullanıcı oluşturulur.
app.config.keycloak.server-url: http://localhost:8080
app.config.keycloak.realm: saupayment
app.config.keycloak.clientId: saupay
app.config.keycloak.client-secret: F2x5l2HVKJ8q9wsnYugksLtbh1DWCvN6

```

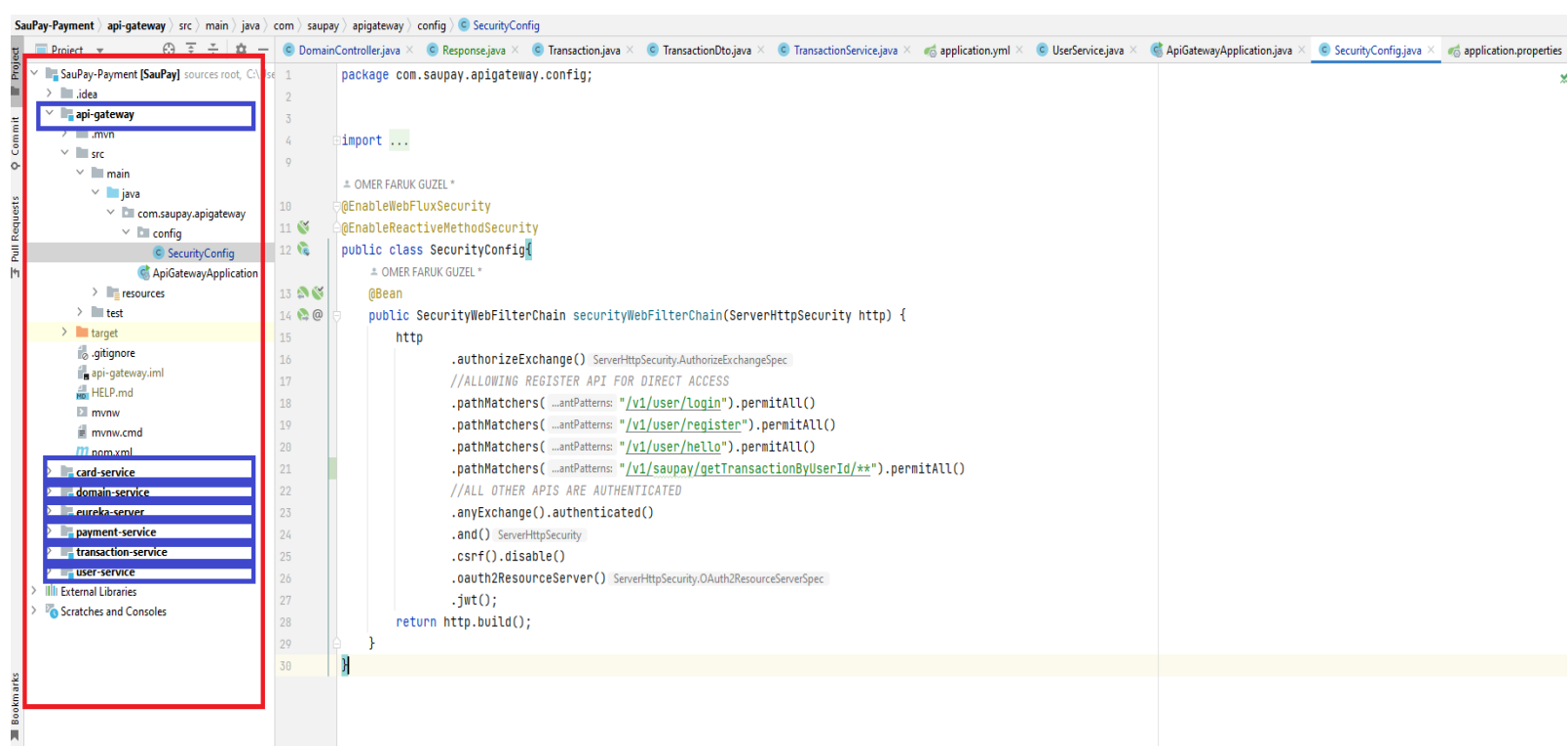
Görsel 3.3.2.6 Keycloak ile User Service konfigürasyonu

3.3.3. Saupay Ödeme Sistemi Veritabanı Diyagramı



Görsel 3.3.3 Saupay Ödeme Sistemi Veritabanı Diyagramı

3.3.4. Saupay Backend MicroService Mimari Kodlaması



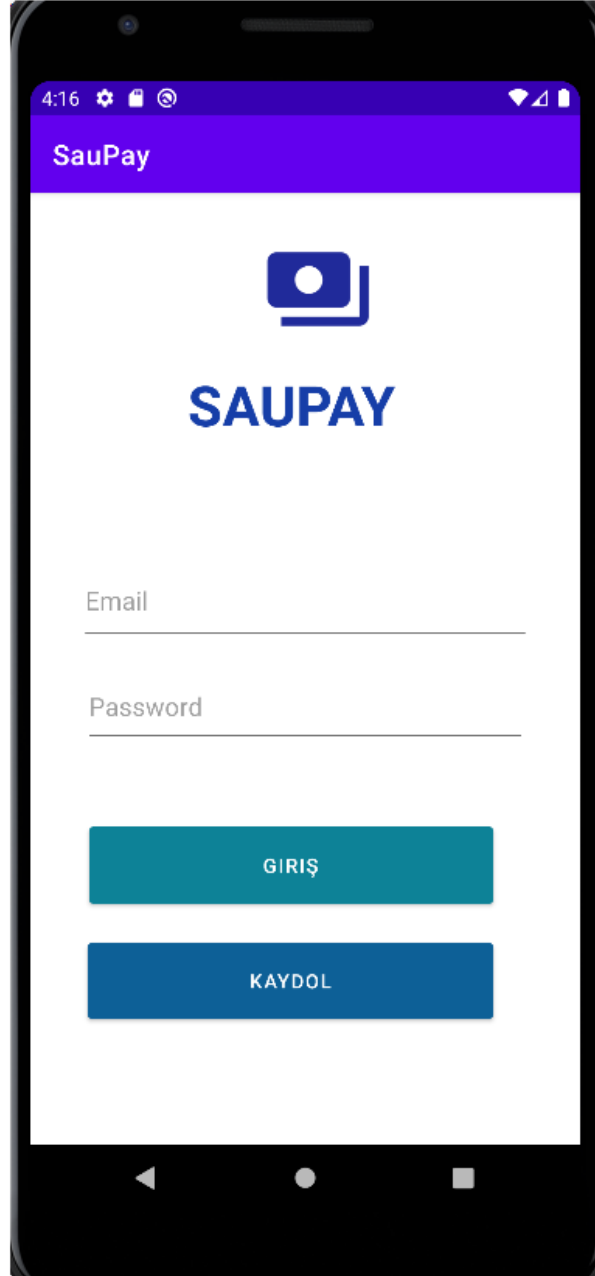
Görsel 3.3.4 Saupay Backend MicroService Mimari Kodlaması

3.4.1 SauPay ödeme uygulaması mobil arayüz tasarımı

SauPay ödeme uygulaması, ödeme sisteminin kullanıcıyla etkileşimle bulunduğu, gerekli verilerin kullanıcıya aktarıldığı, gerekli verilerin ise kullanıcıdan alındığı ara yüzü tasarlanan mobil uygulamadır. Bu uygulamanın Android platformu için tasarlanmış şekli tasarlanmış ve sunulmuştur.

3.4.1.1. SauPay ödeme uygulaması giriş sayfası mobil arayüz tasarımı

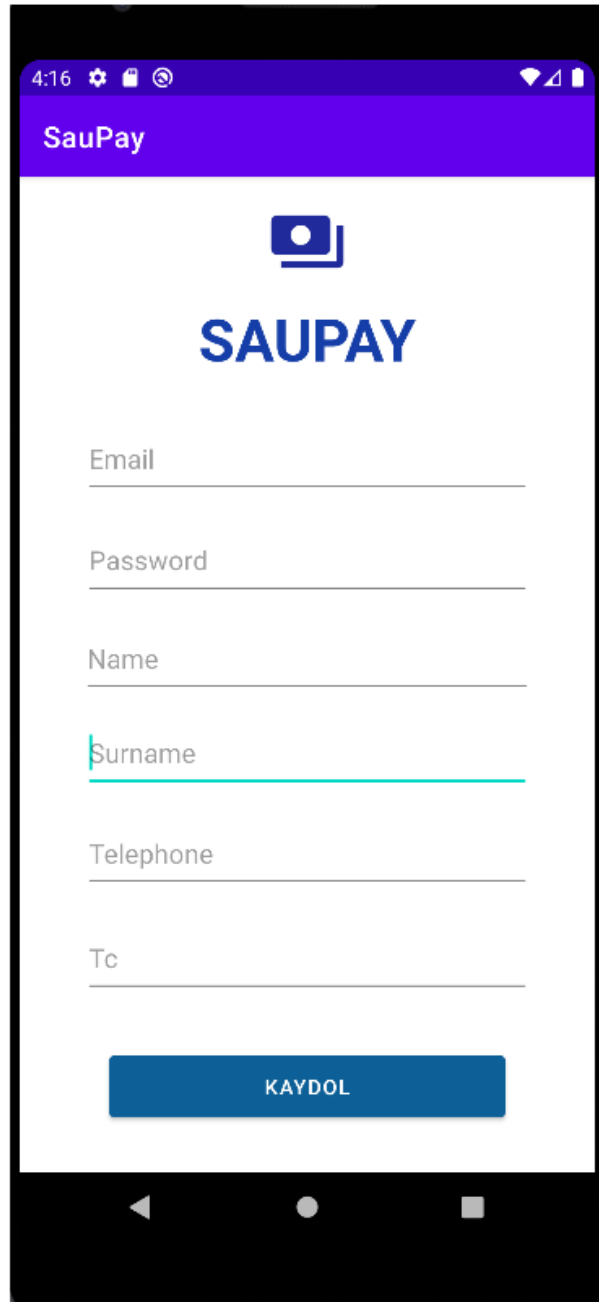
Tasarımı yapılan SauPay ödeme uygulaması giriş sayfası, kayıtlı kullanıcıların sistemden gelen uygulama üzerindeki verilerine erişebilmesi için etkileşimde bulunduğu güvenlik sayfasıdır.



Görsel 3.4.1.1 Ödeme Uygulaması Giriş Ekranı Tasarımı

3.4.1.2. SauPay ödeme uygulaması kaydol sayfası mobil arayüz tasarımı

Tasarımı yapılan SauPay ödeme uygulaması kaydol sayfası, kayıtlı olmayan kullanıcıların sistemden gelen uygulama üzerindeki verilerine erişebilmesi için sisteme kaydolacağı sayfadır.

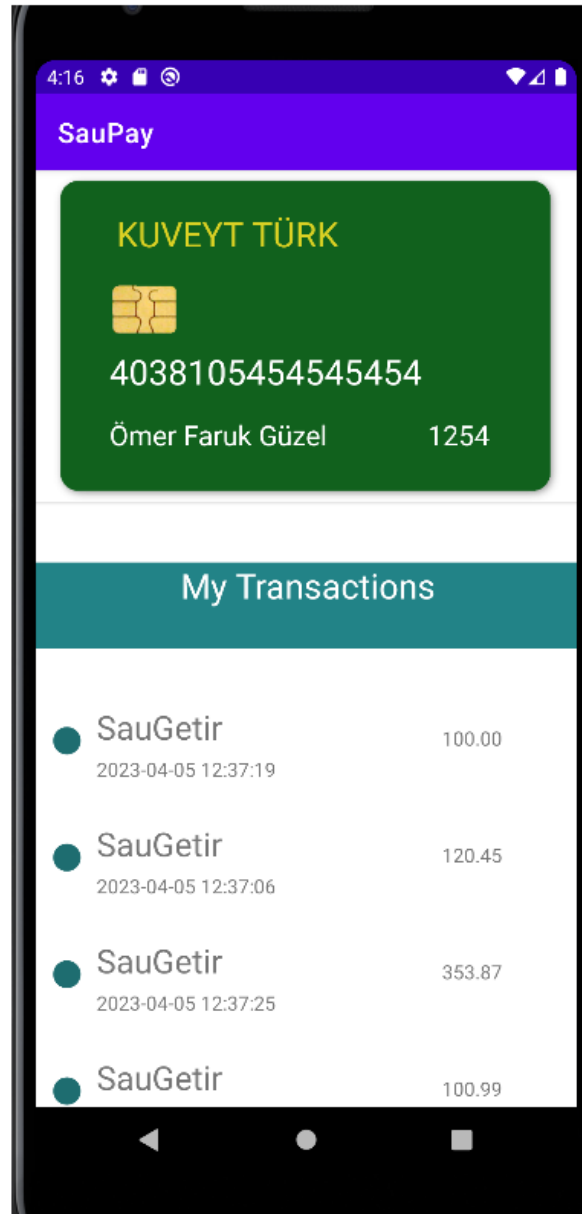


The image shows a mobile app registration screen for SauPay. The screen has a white background with a blue header bar at the top containing the text "SauPay". Below the header is a blue logo consisting of a square with a circle inside, followed by the word "SAUPAY" in blue capital letters. The registration form consists of several input fields with labels: "Email", "Password", "Name", "Surname", "Telephone", and "Tc". Each field has a horizontal line for text entry. The "Surname" field has a red underline. At the bottom of the form is a blue button with the text "KAYDOL" in white capital letters. The screen is framed by a black border, and the bottom navigation bar of the phone is visible at the very bottom.

Görsel 3.4.1.2 Ödeme Uygulaması Kaydol Ekranı Tasarım

3.4.1.3. SauPay ödeme uygulaması anasayfa sayfası mobil arayüz tasarımı

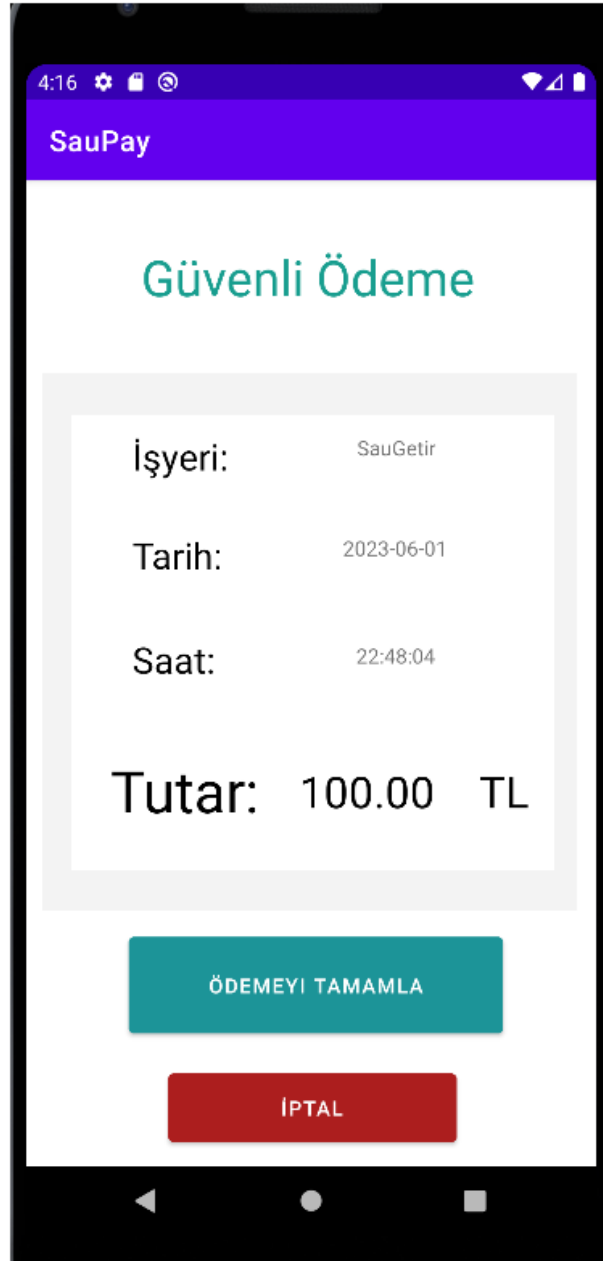
Tasarımı yapılan SauPay ödeme uygulaması giriş sayfası, kayıtlı kullanıcıların sisteme giriş yaptıktan sonra kart bilgilerine ve daha önce yapmış olduğu ödeme işlem verilerine erişebildiği, kullanıcının cüzdanını temsil ettiği anasayfa/cüzdanım sayfasıdır.



Görsel 3.4.1.3 Ödeme Uygulaması Anasayfa Ekranı Tasarımı

3.4.1.4. SauPay ödeme uygulaması ödeme tamamlama sayfası mobil arayüz tasarımı

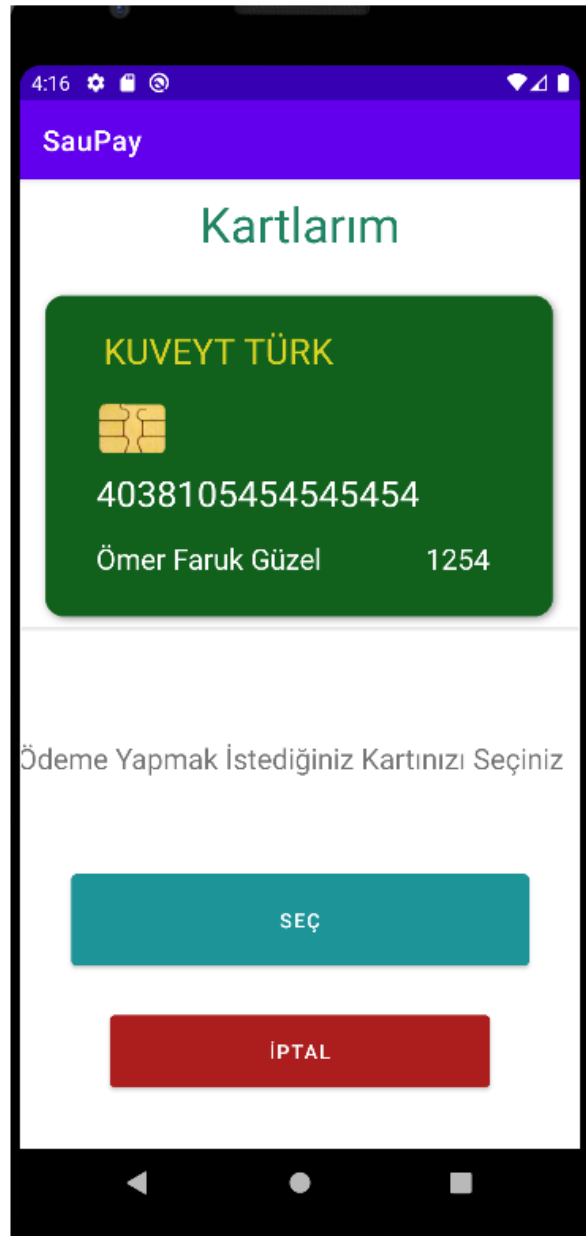
Tasarımı yapılan SauPay ödeme uygulaması ödeme tamamlama sayfası, sisteme giriş yapan kullanıcıların ödeme işlemini gerçekleştireceği zaman etkileşimde bulunduğu, ödeme verilerinin doğrulanması ve ödeme işlemine devam edilebilmesi için onaylanması gereken veya ödeme iptal işlemi gerçekleştirilebilen işlem sayfasıdır.



Görsel 3.4.1.4 Ödeme Uygulaması Ödeme Tamamlama Ekranı Tasarımı

3.4.1.5. SauPay ödeme uygulaması ödeme işlemi kart seçimi sayfası mobil arayüz tasarımı

Tasarımı yapılan SauPay ödeme uygulaması ödeme işlemi kart seçim sayfası, sisteme giriş yapan kullanıcıların ödeme işlemini gerçekleştireceği zaman ödeme tamamlama onayı verildikten sonra etkileşimde bulunduğu, ödeme işlemini gerçekleştireceği kartını seçtiği veya işlemi iptal edebildiği işlem sayfasıdır.



Görsel 3.4.1.5 Ödeme Uygulaması Kart Seçim Ekranı Tasarımı

3.4.1.6 SauPay ödeme uygulaması 3D secure/ödeme sayfası mobil arayüz tasarımı

Tasarımı yapılan SauPay ödeme uygulaması ödeme işlemi kart seçim sayfası, sisteme giriş yapan kullanıcıların ödeme işlemini gerçekleştireceği zaman ödeme tamamlama onayı verildikten ve ödeme yapacağı kartını seçtikten sonra etkileşimde bulunduğu, ödeme işlemi için sms ile doğrulama yaptığı ödeme doğrulama sayfasıdır.

The screenshot shows a mobile application interface for SauPay. At the top, there is a purple header with the 'SauPay' logo. Below the header, the title '3D Secure Ödeme Doğrulama' is centered. The main content area is a light gray box containing the following information:

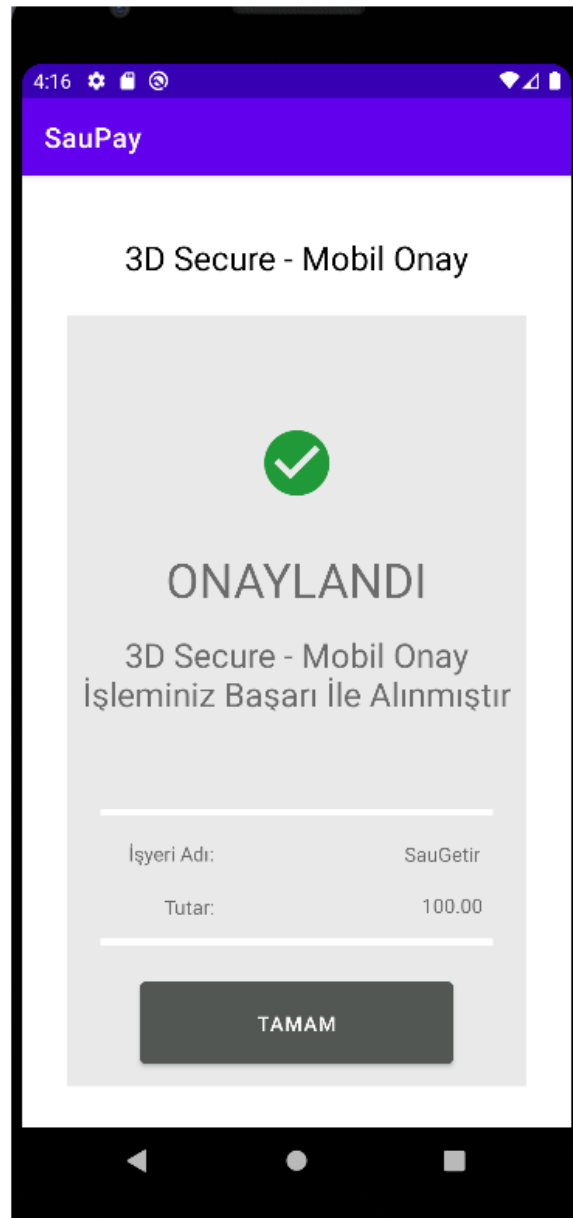
- İşyeri Adı:** SauGetir
- Tutar:** 100.00
- Tarih:** 2023-06-01 22:48:04
- Saat:** 2023-06-01 22:48:04
- Kart Numarası:** 4038105454545454
- Tel. No:** 5370411509
- Doğrulama Kodu:** Kodu Giriniz

Below the input field, there is a link: [Doğrulama Kodu Almadıysanız Tıklayınız.](#) At the bottom of the gray box, a timer shows '00:00'. At the very bottom of the screen, there is a green button labeled 'TAMAM'.

Görsel 3.4.1.6 Ödeme Uygulaması 3D Secure / Ödeme Ekranı Tasarım

3.4.1.7 SauPay ödeme uygulaması ödeme onayı sayfası mobil arayüz tasarımı

Tasarımı yapılan SauPay ödeme uygulaması ödeme işlemi kart seçim sayfası, sisteme giriş yapan kullanıcıların ödeme işlemini gerçekleştireceği zaman ödeme tamamlama onayı verildikten, ödeme yapacağı kartını seçtikten ve sms ile doğrulama yaptıktan sonra etkileşimde bulunduğu, ödeme işleminin başarılı sonuçlandığı bilgisinin yer aldığı onay sayfasıdır.






Görsel 3.4.1.7 Ödeme Uygulaması Ödeme Doğrulama Ekranı Tasarımı

3.4.2 SauGetir Alışveriş Uygulaması Mobil Arayüz Tasarımı

SauGetir alışveriş uygulaması kullanıcıların ürün siparişi verebildikleri, ödeme işlemlerini online olarak gerçekleştirebildikleri çevrim içi çalışan alışveriş uygulamasıdır. Bu uygulama, sistemimizle birlikte ile test edilebilmesi için ‘SauGetir ile öde’ seçeneğinin seçildiği ödeme işleminin akışına göre tasarlanmıştır. Bu ödeme işlemi testi için sepete ürünler eklenmiş ve ödeme aşamasına getirilmiş şekilde tasarımı yapılmıştır.

3.4.2.1 SauGetir alışveriş uygulaması sipariş tamamlama sayfası mobil arayüz tasarımı

Tasarımı yapılan SauGetir alışveriş uygulaması sipariş tamamlama sayfası, sepetteki ürünlerin siparişinin ödemesinin gerçekleştirileceği ödeme tamamlama sayfasıdır. Bu sayfada ‘SauPay ile Öde’ seçeneği seçilecektir.

1		Yumoş Extra Konsantre Çamaşır Yumuşatıcısı Lilyum 1440 ML	50.00 TL
1		Sırma Plus Narlı 6x200 ml Meyveli Soda	15.00 TL
1		Çerezza Cips Tv 68 gr Süt Mısır Aile	15.00 TL
Ara Toplam		90.00 TL	
İndirim		- 30.00 TL	
Gönderim Ücreti		7.00 TL	
Toplam		67.00 TL	
<p>Ödemeyi Tamamla</p> <p>SauPay ile Öde</p>			

Görsel 3.4.2.1. E Ticaret Uygulaması Sipariş Tamamlama Ekranı Tasarımı

3.4.3 SauBank Banka Uygulaması Mobil Arayüz Tasarımı

SauBank banka uygulaması, kullanıcının online ödeme yapacağı kartının ait olduğu bankanın simüle edildiği, kullanıcının ana hesap miktarını tutan uygulamadır. Ödeme sisteminin ödeme aldığı bilgisi kullanıcıyla etkileşime girerek bildirebileceği tasarlanan banka uygulamasıdır.

3.4.3.1 SauBank banka uygulaması anasayfa sayfası mobil arayüz tasarımı

Tasarımı yapılan SauBank banka uygulaması anasayfa sayfası, banka kullanıcısının ana hesap miktarını görüntüleyebildiği ve hesaplarının bilgilerini ve hesap tutarlarının verilerine erişebildiği, ödeme sistemi için ana amacı bilgi aktarımı olan sayfadır.



Görsel 3.4.3.1. Banka Uygulaması Anasayfa Tasarımı

BÖLÜM 4. Ödeme Sistemi Testi İçin Tasarlanan Uygulamalar

SauPay Ödeme sistemi, işletmelerin mobil ödeme işlemleri için gerçekleştirilecek bir mobil uygulamadır. Bu uygulamanın önyüzü Android platformunda kotlin dili ile arka yüzü ise Java / Spring Boot ile gerçekleştirilecektir.

4.1 Gerçekleştirilen Uygulamalar

SauPay ödeme uygulaması test edilmesi için bu sisteme entegre olmuş işletme ve ödeme alınacak banka uygulaması gereklidir. Bu ihtiyaca binaen sahte bir işletme uygulaması ve sahte banka uygulaması gerçekleştirilecektir. Fakat bu uygulamalar sahte olmasının yanı sıra veriler gerçek dünya verilerine eş yakınlıkta olacaktır. İşletme, SauPay sistemine başvurmuş ve sisteme entegre olmuş bir vaziyette gerçekleştirilecektir.

4.1.2. SauPay uygulaması

SauPay uygulaması, kullanıcıların işletmeler üzerinden yaptıkları ödeme işlemlerini gerçekleştirmek için Android platformunda kotlin dili ile yazılmıştır. Bu uygulama işletme ve kullanıcının kullandığı banka ile irtibata geçer ve ödeme yapabileceği kartlarını barındırarak ödeme işlemlerini gerçekleştirir.

4.1.3. SauGetir uygulaması

SauGetir, mobil uygulaması aracılığı ile kullanıcılarının ihtiyaçlarına binaen market ürünleri için teslimat hizmeti veren bir işletmedir. Bu işletme SauPay ödeme sistemine entegre olmuş ve ödemelerini SauPay ile almaktadır. SauGetir mobil uygulaması Android platformunda kotlin dili ile yazılacaktır.

4.1.4. SauBank uygulaması

SauBank uygulaması, kullanıcıların işletmeler üzerinden yaptıkları ödeme işlemlerini gerçekleştirirken kart ve cari hesap bilgilerini içeren banka uygulamasıdır. Bu uygulama Android platformu üzerinde kotlin dili ile yazılacaktır.

4.2 Gerçekleştirilen Uygulamalar Arası İletişim

Ödeme uygulaması testi için gerçekleştirilen uygulamalar birbirleri ile iletişim halindedirler ve bilgi alışverişinde bulunurlar. Bu haberleşmelerde önyüz için farklı arka yüz için farklı teknolojiler kullanılır.

4.2.1. Önyüz Haberleşmesinde Kullanılan DeepLink Yöntemi

Mobil uygulamalarda deep linking, uygulamayı başlatmak yerine mobil uygulama içerisinde yer alan bir screen'i (içerikler, işlemler vb.) işaret eden bir URI (Uniform Resource Identifier) ile ifade edilebilir. Mobil cihaz platformuna bağlı olarak, uygulamayı tetiklemek için gereken URI farklılık gösterebilir. Eğer bir uygulama içerisindeki bir içeriği bir başka kullanıcıyla paylaşmak isterseniz ihtiyacınız olan ilgili içeriğe ait bir derin bağlantı olacaktır. Aksi durumda, kullanıcı ilgili uygulamayı uygulama marketinde aratmalı, bulup indirmeli ve ilgili içeriğe uygulama içerisinden ulaşmaya çalışmalıdır. Elbette bu kullanıcı deneyimi açısından oldukça sorunlu bir yaklaşım olacaktır. Mobil uygulamalar (mobile app) aracılığıyla ele alacağımız derin bağlantıları özelliklerine göre 3 başlık altında değerlendirebiliriz. [4].

Uygulamalar üzerinde mobil arayüz ile mobil arayüz arasındaki haberleşmede kullanılacaktır.

4.2.2. Arka Yüz Haberleşmesinde Kullanılan Rest Api Yöntemi

Rest Api, geliştiricilerin http protokolü üzerinden Get, Post, Put, Delete isteklerini atabildiği ve arzulanan işlemlerin yapıldıktan sonra geriye Json, Html, Xml vs. gibi farklı formatlarda sonuç alabildiği bir sistemdir. Rest Api, http protokolünü kullanıyor olması nedeniyle **herhangi bir programlama dilinde kullanılabilir**. Dolayısıyla dilden, bir teknoloji veya platformdan

bağımsızdır. Sunucunun istekleri nasıl işlediğini ve bu isteklere sunucunun nasıl yanıt verdiğini açıklar. Uygulamalar üzerinde mobil arayüz ile arka yüz servisleri arasında ve arka yüz servisleri ile arka yüz servisleri arasındaki haberleşmede kullanılacaktır.

BÖLÜM 5. Güvenli Ödeme Uygulaması Kullanımı ve Sonuçları

5.1 SauPay Ödeme Sistemi Kullanımının Amaçlanması

SauPay ödeme sistemi, kullanıcıların bu sisteme entegre olmuş işletmeler üzerinden kart bilgilerinin saklanması ve veri akışının güvenli iletilmesiyle birlikte sorunsuz, güvenli ve ödeme sahtekarlığının önüne geçilen bir ödeme işlemi hizmeti sunmayı amaçlamaktadır. Ayrıca SauPay, kullanıcının bu sistemi kullanarak farklı bankalar üzerinden, farklı kartlar ile yaptığı ödeme işlemlerinin geçmişini ve bilgilerini kullanıcıya sunmayı amaçlamaktadır.

5.2 SauPay Ödeme Sistemi Kullanımının Avantajları

- İşletmeler SauPay ödeme sistemine kolayca entegre olabilir.
- Kullanıcının farklı bankalara ait farklı kart bilgileri kaydedilebilir ve kullanılabilir.
- Ödeme işlemleri güvenli bir şekilde gerçekleşir.
- 3D Secure prensipleriyle çalışır.
- Saldırıları karşı şifreleme algoritmaları kullanılır.
- Ödeme işleminde çift taraflı doğrulama kullanılır.
- Kullanıcıdan ödeme işleminde doğrulama kodu alır ve ödeme sahtekarlığının önüne geçer.
- Ödeme akışında sms ile doğrulama için OTP (Tek kullanımlık şifre) kullanılır.
- Tüm kartlardaki geçmiş ödeme detayları tek bir platformdan görüntülenebilir
- İşletmelere kart bilgisi verilmeden ödeme işlemi yapılabilir.
- Mobil arayüz kullanımı kolay ve etkilidir.

5.3 İşletmelerin SauPay Ödeme Sistemine Entegre Olması

İşletmeler SauPay ödeme istemine entegre olabilmek için SauPay'e saupay54@gmail.com adresine online başvuruda bulunurlar. SauPay bu e-posta aracılığı ile işletmelere başvuru formu gönderir. Bu başvuru formunda işletmelere ait bilgiler istenir ve işletmeler bu formu doldurup başvuru isteklerini SauPay'e bildirirler. Daha sonra başvurular değerlendirmeye alınır ve değerlendirme sonucunda onay verilirse işletmelere bildirilir ve işletmelere özel Merchant code ve secret key iletilir. Artık sisteme entegre olan işletmeler, SauPay ödeme sistemini kullanacağı ödeme işlemlerinde kendilerine gönderilen merchant code ile istek atarlar. İstekleri karşılayan ödeme sistemi servisi merchant code doğrulama işlemi sonucunda gerekli akışı devam ettirir veya sonlandırır.

KAYNAKLAR

- [1] <https://www.ibm.com/cloud/learn/java-spring-boot>
- [2] <https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/>
- [3] [https://tr.wikipedia.org/wiki/RSA_\(%C5%9Fifreleme_y%C3%B6netimi\)#cite_note-:0-1](https://tr.wikipedia.org/wiki/RSA_(%C5%9Fifreleme_y%C3%B6netimi)#cite_note-:0-1)
- [4] <https://ceaksan.com/tr/deep-linking>

ÖZGEÇMİŞ

Ömer Faruk Güzel, 20.09.2000 de Bayburt 'da doğdu. 2006 yılında ailesiyle birlikte İstanbul 'a taşındı. İlk ve orta öğrenimini Gaziosmanpaşa 'da Gaziosmanpaşa ilk ve orta okulunda tamamladı. 2018 yılında Çemberlitaş Anadolu Lisesinden mezun oldu. 2019 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nü kazandı. 2021 yılında Initial Code Software Solutions şirketinde yazılım stajını yapmıştır. Şu anda Bilgisayar Mühendisliği 4. Sınıfta öğrenimini devam ettirmektedir.

BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU: Güvenli Ödeme Sistemi

ÖĞRENCİLER (Öğrenci No/AD/SOYAD): G191210068/Ömer Faruk Güzel

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuzuna uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN: ÜNAL ÇAVUŞOĞLU

DANIŞMAN İMZASI: