BUA (Bakircay University Application)

Öğrenci No:
Ali Alper Şahin 200601022
Bora Körpe 200601003
Ahmet Bilgin 200601016
Grup No: 1

İçindekiler

İterasyon-1

Hafta-1 : Projenin Özeti

Bu proje İzmir Bakırçay Üniversitesi resmi uygulamasıdır. Uygulamanın iki temel işlevi vardır:

1. Üniversite giriş-çıkışı ve yemekhane girişini QR kod ile sağlamak
2. Yemekhane rezervasyonu yapmak

Üniversite giriş-çıkışı ve yemekhane girişini QR kod ile sağlamak
Turnikelerden geçiş işlemi için iki farklı yol izlenebilir. Birincisi turnike üzerindeki ekranda bulunan QR kod, telefondaki uygulamadan telefon kamerasına erişilerek okutulur. İkincisi ise uygulama üzerinde 60 sn'lik kişiye özel QR kod oluşturulur ve telefon ekranı turnikedeki QR kod okutma alanına okutulur. Böylelikle turnikelerden geçiş işlemi temassız hale getirilir.

Yemekhane rezervasyonu yapmak
Yemekhane rezervasyonu oluşturmak için uygulamadan rezervasyon ekranı açılır, uygun günler sepete eklenir, yeterli bakiye uygulama cüzdanına yüklenir ve ödeme yapılarak rezervasyon yapılır.

Uygulamayı üniversite sistemine kayıtlı olan herkes kullanabilir. Fiziksel öğrenci kartlarını kaldırıp sanal ortama taşımak ilk başta adaptasyon sorunları oluşturabilir. Bu sorun süreç ilerledikçe azalacaktır. Fiziksel öğrenci kartları sadece okula giriş için değil çoğu kurumda öğrenci avantajlarından yararlanmak için kullanılmaktadır. Bu kartı kaldırmak bu avantajları edinmeyi de kısıtlayabilir. Ancak uygulama üzerinde geliştirilecek öğrenci olma durumunu gösterir QR kod sorunu çözecektir. Şu anda üniversitenin bir uygulamaması bulunmamaktadır. Yemek rezervasyonu web site üzerinden yapılmaktadır ve bazı hatalar içermektedir. Örneğin bakiye yükledikten sonra yemek rezervasyonu yapmak istersek sistem hata verir. Rezervasyon yapabilmek için sistemden çıkıp tekrar giriş yapmamız gerekir. Uygulama bu sorunları ortadan kaldırmayı, daha hızlı bir şekilde rezervasyon yapmayı, ayrıca rezervasyon iptal sistemi getirerek bakiye iadesini sağlamayı amaçlamaktadır. Projenin ilerisindeki hedeflerden biri okulda kahvaltı ve akşam yemeği oluşturarak yemekhane rezervasyon sitemini bu öğünlerde de kullanmaktır. Diğeri ise QR kod sistemini kütüphaneye de entegre etmektir. Kütüphaneye QR kodla giriş yapılarak kişinin önceden oturacağı yer seçilir, doluluk oranı gösterilir ve kişiye 1 uzun ve 1 kısa olmak üzere iki mola tanımlanır. Böylece kütüphaneye eşya bırakıp uzun süre dönmeme, başkasın yer tutma gibi nedenlerden dolayı oluşan kütüphanede yer bulma problemi ortadan kaldırılır. Dönem boyunca uygulama kullanmaktaki amaç kampüsü daha akıllı hale getirmek, kampüs yaşantısını kolaylaştırmak ve yemek rezervasyonunu daha sağlıklı bir hale getirmektir. Bu proje sonucunda bütün üniversiteler bu uygulamanın altyapısını kullanarak projenin gerçek hayattaki uygulaması için potansiyel müşterileri olabilir. Fiziki kart maliyetinden kurtulmak, giriş çıkışlarda QR kod kullanmak ve yemek rezervasyonunu daha sağlıklı hale getirmek isteyen her kurum ve kuruluş potansiyel müşteri olabilir.

Hafta-2 : Başlangıç Analizi

YARARLAR:

Uygulamadan QR kod oluşturulması:

1. Fiziksel öğrenci kartı basım maliyetini ortadan kaldırır.
2. Öğrenci kartının unutulmasından dolayı üniversiteye girişte yaşanan problemleri çözer.
3. Öğrenci kartı kaybolduğunda yeni kart basılması gerekmesini ve bu kart için tekrardan bir maliyet oluşmasını önler.
4. Her QR kod kişiye özel olarak 60 sn'lik bir süre için oluşturulacağından üniversiteye kayıtlı olmayıp başkasının öğrenci kartıyla üniversiteye girmeye çalışan yabancılar engellenerek üniversite ortamı daha güvenli bir hale gelir.

Yemekhane rezervasyonunun uygulama üzerinden yapılması:

1. Rezervasyonu kolaylaştırır.
2. Bir hafta önceden kullanıcıya bildirim yollayarak rezervasyon yapılmasının unutulmasını engeller.
3. Rezervasyonun 6 saat öncesine kadar iptal edilmesini ve paranın iade edilmesini sağlar.

EKSİKLİKLER:

1. Turnike sistemlerinin değişmesi için bir maliyet oluşturur.
2. Uygulamaya bağlanıp üniversiteye giriş ve çıkış işlemleri için QR kod oluşturulacakken internet kesintisi yaşanabilir.
3. Turnikelere çok fazla güneş ışığının vurması nedeniyle turnike üzerindeki QR kodun telefona okutulamayabilir.
4. Telefon ekranındaki bir sorun nedeniyle telefondan oluşturulan QR kodun turnikeye okutulamayabilir.

AMAÇ:

Fiziksel kart maliyetini ortadan kaldırarak turnikelerden geçiş sürecini daha kolay hale getirmek ve yemekhane rezervasyon sürecini iyileştirmek

MALİYET:

1. Turnike Maliyetleri
2. Çalışan Maliyetleri
    a. 2 yazılımcı maaşı 15.000 TL
    b. 1 yönetici maaşı 25.000 TL
    c. 1 tester ücreti 12.000 TL
    d. Sunucu ücreti 10.000 TL

1 aylık toplam maliyet 15.000 + 25.000 + 12.000 + 10.000 = 62.000 TL olur. Proje geliştirme 12 ay süreceğinden yıllık maliyet 62.000 * 12 = 744.000 TL olur.

İZİNLER:

1. Öğrenci bilgilerinin paylaşımı
2. Bakırçay UBYS sistemi ile entegrasyon

ZAMAN:

1. Proje geliştirme süresi 12 ay olarak belirlenmiştir.
2. Okul açılış ve kapanış saatine uygun olarak sürekli çalışmalı.

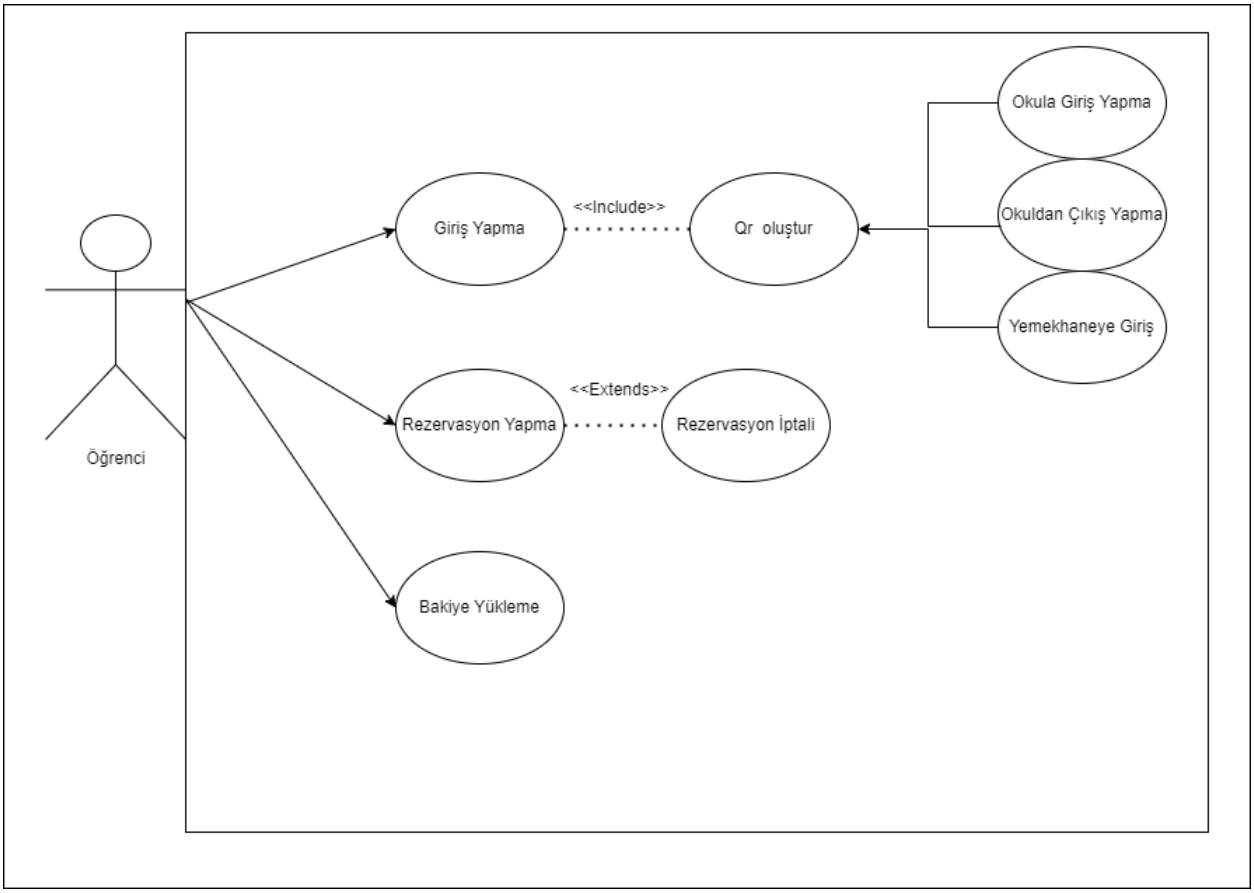
VERİ:

Verilerin yönetiminde AWS kullanılacaktır.

TESTLER:

- BOSS Test: Bütün durumlar en iyi şekilde gerçekleştiğinde sistem düzgün çalışmaktadır.
- EBP Test: Bir kullanıcı bir zamanda bir işlem gerçekleştirebildiği için proje bu testten geçmiştir.
- Size Test: Sistemdeki adım sayısı birden fazladır. Dolayısıyla bu da doğrudur.

Hafta-3 : Kullanım Durumları



USE CASE UC1: Giriş Yapma (Okula veya Yemekhaneye)
Scope: BUA (Bakırçay University Application)
Level: user goal
Primary Actor: Kullanıcı
Stakeholders and Interests:
-Kullanıcı: Üniversiteye kayıtlı kimse turnikelerden geçerek kampüse ve yemekhaneye giriş yapar.
Preconditions: Kişi üniversiteye kayıtlı olmalıdır. Giriş yapmak için QR kod oluşturmalı veya turnikedeki QR kodu telefona okutmalıdır.
Success Guarentee (or Postconsitions): Kişi turnikelerden geçerek başarılıyla giriş yapar.
Main Success Scenario (or Basic Flow):

1. Kullanıcı uygulamayı açar.
2. Kullanıcı öğrenci no ve şifre ile uygulamaya giriş yapar.
3. QR Kod işlemi gerçekleştirilir.
   ↳ *Bu işlem use case 2 olarak daha detaylı şekilde anlatılmıştır.*
4. Turnikeden geçilir
5. Giriş tamamlanır.

Extensions (or Alternative Flows):

1. QR kod turnike tarafından algılanmaz.
2. Kullanıcı tekrar QR kod oluşturur.
3. Turnikeden geçilir.
4. Giriş tamamlanır.

USE CASE UC2: QR İşlemini Gerçekleştirmek
Scope: BUA (Bakırçay University Application)
Level: user goal
Primary Actor: Kullanıcı
Stakeholders and Interests:
-Kullanıcı: Üniversiteye kayıtlı kimse turnikelerden geçmek için QR işlemlerini gerçekleştirir.
Preconditions: Kullanıcı uygulama üzerinden QR oluşturmalı veya QR kodu okutmalıdır.
Success Guarentee (or Postconsitions): Kullanıcı başarıyla QR üretmiş ya da QR okutmuş olur.
Main Success Scenario (or Basic Flow):

1. Kullanıcı uygulamayı açar.
2. Kullanıcı öğrenci no ve şifre ile uygulamaya giriş yapar.
3. QR Kod işlemi gerçekleştirilir. 2 farklı yol vardır.
   3.1. Telefondan QR kod oluşturmak
   3.2. Turnike üzerindeki QR kodu kullanmak
4. Başarıyla QR koda erişilir.

Extensions (or Alternative Flows):

1. Qr kod turnike tarafından okunmaz.
   i. Kullanıcı tekrar kod oluşturur.
   ii. Kullanıcı tekrar Qr kodu okutur.

2. Turnikedeki Qr kod fiziksel deformasyona uğradıysa veya telefon Qr'ı okuyamaz.
    i. Mobil uygulama üzerinden Qr kod oluşturulur.
    ii. Turnikeden geçiş işlemi sağlanır.

USE CASE UC3: Rezervasyon Yapma
Scope: BUA (Bakırçay University Application)
Level: user goal
Primary Actor: Kullanıcı
Stakeholders and Interests:
-Kullanıcı: Üniversiteye kayıtlı kimse uygun tarih ve menüyü seçerek rezervasyon işlemini gerçekleştirir.
Preconditions: Rezervasyon bir hafta önceden yapılmalı ve uygun günler seçilmelidir.
Success Guarentee (or Postconsitions): Kullanıcı başarıyla rezervasyon yapmış olur.
Main Success Scenario (or Basic Flow):
1. Kullanıcı uygulamayı açar.
2. Kullanıcı öğrenci no ve şifre ile uygulamaya giriş yapar.
3. Kullanıcı rezervasyon yap butonuna basar.
4. Kullanıcı uygun günleri seçerek sepetine ekler
5. Kullanıcı ödemesini yapar.
6. Kullanıcı başarıyla rezervasyonu tamamlar.

Extensions (or Alternative Flows):
1. Bakiye yetersizdir ödeme gerçekleşmez.
    i. Bakiye yetersiz uyarısı gösterilir.
    ii. Bakiye yükleme sayfasına yönlendirilir.
2. Kullanıcı rezervasyon yap butonuna basar.
3. Kullanıcı uygun günleri seçerek sepetine ekler
4. Kullanıcı ödemesini yapar.
5. Kullanıcı başarıyla rezervasyonu tamamlar.

USE CASE UC4: Rezervasyon İptali
Scope: BUA (Bakırçay University Application)
Level: user goal
Primary Actor: Kullanıcı
Stakeholders and Interests:
-Kullanıcı: Üniversiteye kayıtlı kimse iptal etmek istediği rezervasyonu seçer ve iptalini sağlar.
Preconditions: Kullanıcının rezervasyonu iptal edebilmesi için rezervasyon saatinden en az 6 saat önce işlemi gerçekleştirmesi gerekir.
Success Guarentee (or Postconsitions): Kullanıcı başarıyla rezervasyon iptali yapmış olur.
Main Success Scenario (or Basic Flow):
1. Kullanıcı uygulamayı açar.
2. Kullanıcı öğrenci no ve şifre ile uygulamaya giriş yapar.
3. Kullanıcı menüyü açar.

4. Kullanıcı rezervasyon iptal et sayfasına gider.
5. Kullanıcı iptal etmek istediği rezervasyonları seçer.
6. Kullanıcı iptal butonuna basar.
7. Kullanıcının iptal ettiği rezervasyonların ücreti sistem bakiyesine iade edilir.
8. Kullanıcı başarıyla rezervasyon iptalini sağlar.

Extensions (or Alternative Flows):

1. İptal edilmek istenen yemek rezervasyon saatinden 6 saatten kısa bir süre içerisinde ise;
    i. İptal gerçekleşmez.
    ii. Bakiye iadesi yapılmaz.

USE CASE UC5: Bakiye Yükleme
Scope: BUA (Bakırçay University Application)
Level: user goal
Primary Actor: Kullanıcı
Stakeholders and Interests:
-Kullanıcı: Üniversiteye kayıtlı kimse bakiyesine para yükler.
Preconditions: Kullanıcı**;** Bakiye yüklemek için içerisinde yeterli limit bulunan bir banka veya kredi kartına sahip olmalıdır.
Success Guarentee (or Postconsitions): Kullanıcı başarıyla bakiye yüklemiş olur.
Main Success Scenario (or Basic Flow):

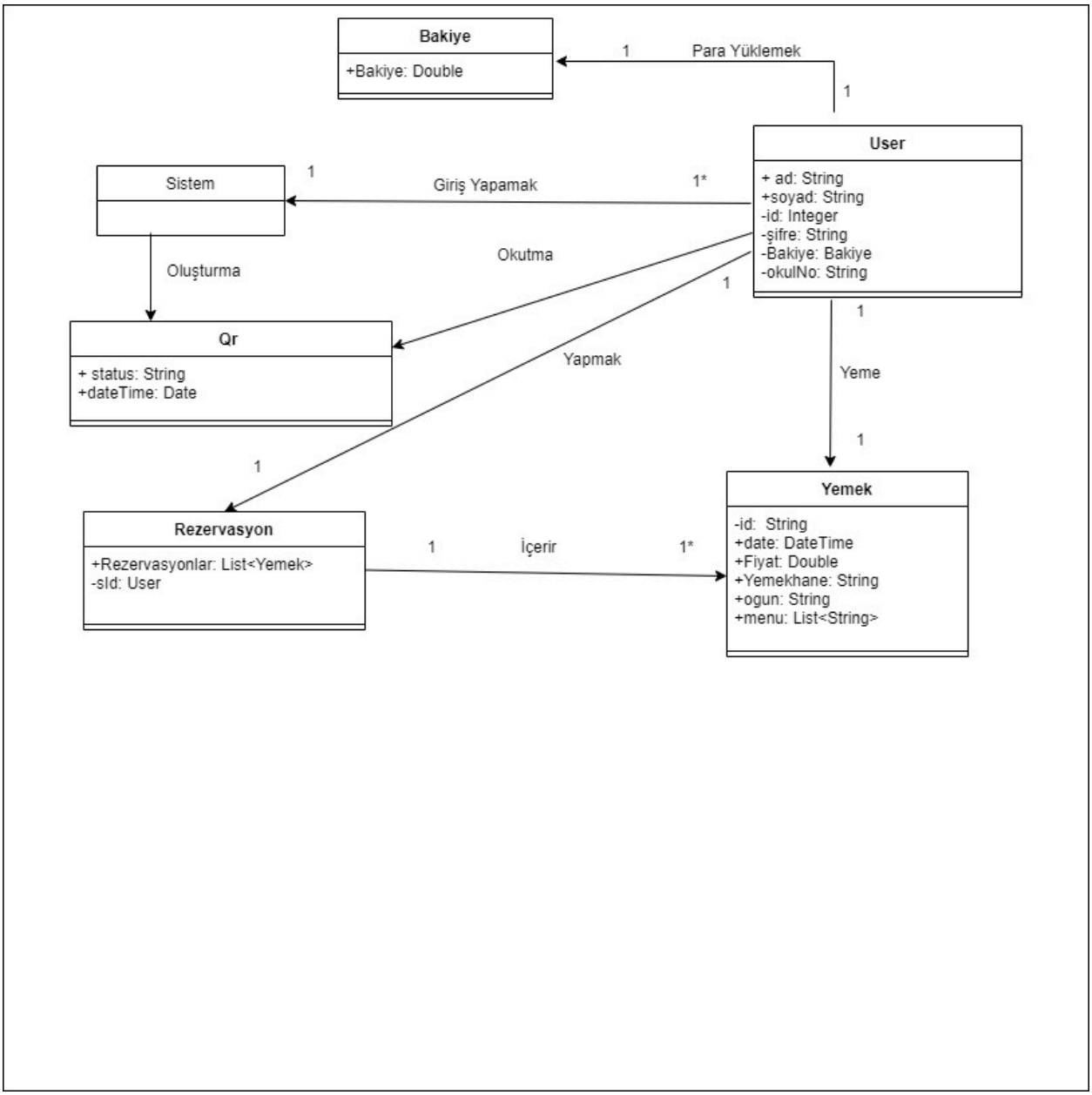
1. Kullanıcı uygulamayı açar.
2. Kullanıcı öğrenci no ve şifre ile uygulamaya giriş yapar.
3. Kullanıcı bakiye yükleme sekmesine gider.
4. Kullanıcı kart bilgilerini girer.
5. Kullanıcı yüklemek istediği tutarı seçer.
6. Kullanıcı ödemesini yapar.
7. Kullanıcı başarıyla bakiyesini yükler.

Extensions (or Alternative Flows):

1. Kullanıcı kart bilgileri hatalı olur.
    i. Kullanıcı tekrar kart bilgisini girer.
    ii. Onaylanırsa para yüklemesi gerçekleşir.
2. Yükleme yapılacak kartta yeterli bakiye yoksa.
    i. Kullanıcı yetersiz bakiye mesajı görüntüler.
    ii. Yeniden kart bilgileri istenir.

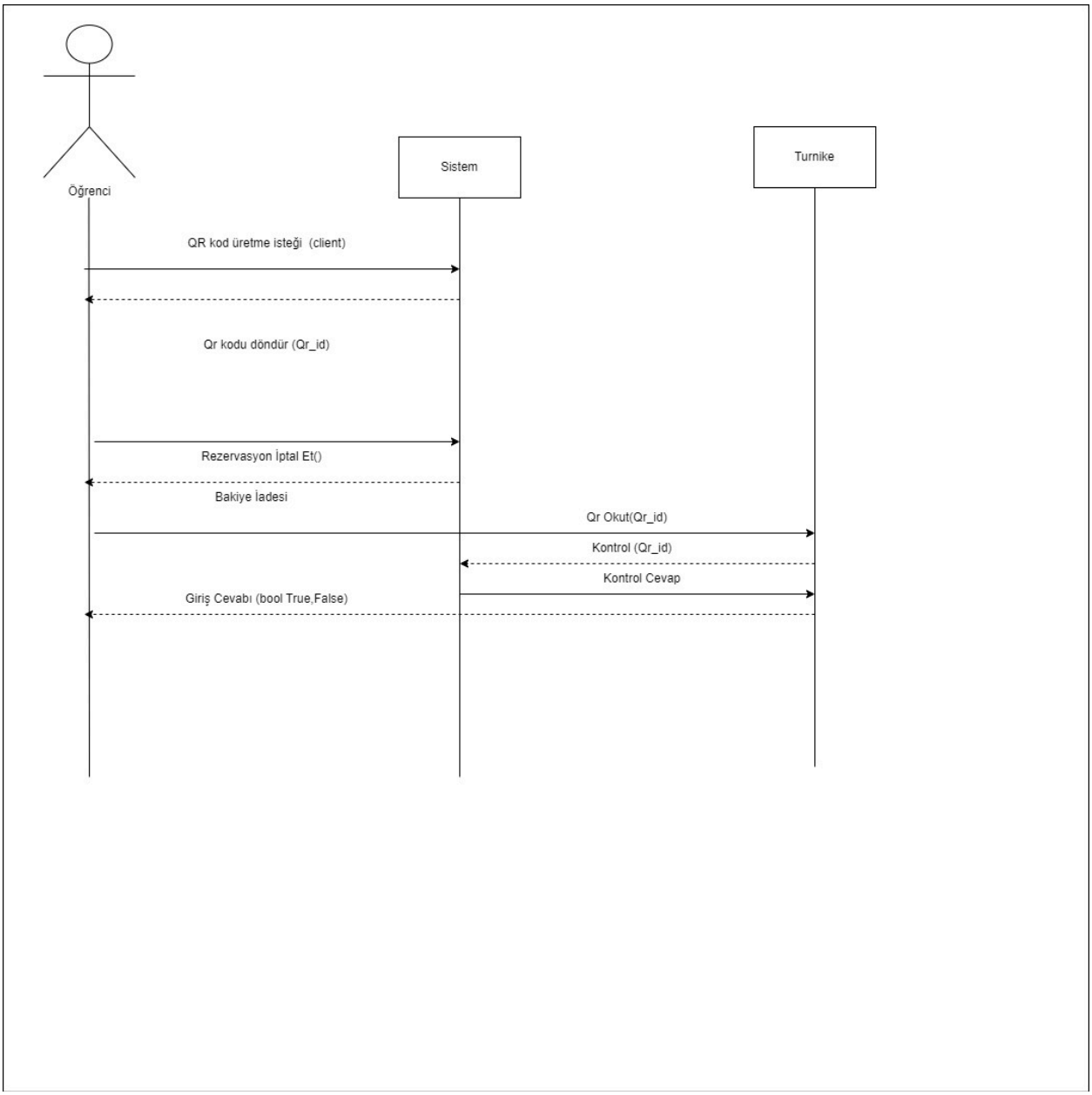
Hafta-4 : Değerlendirme ve Alan Modeli

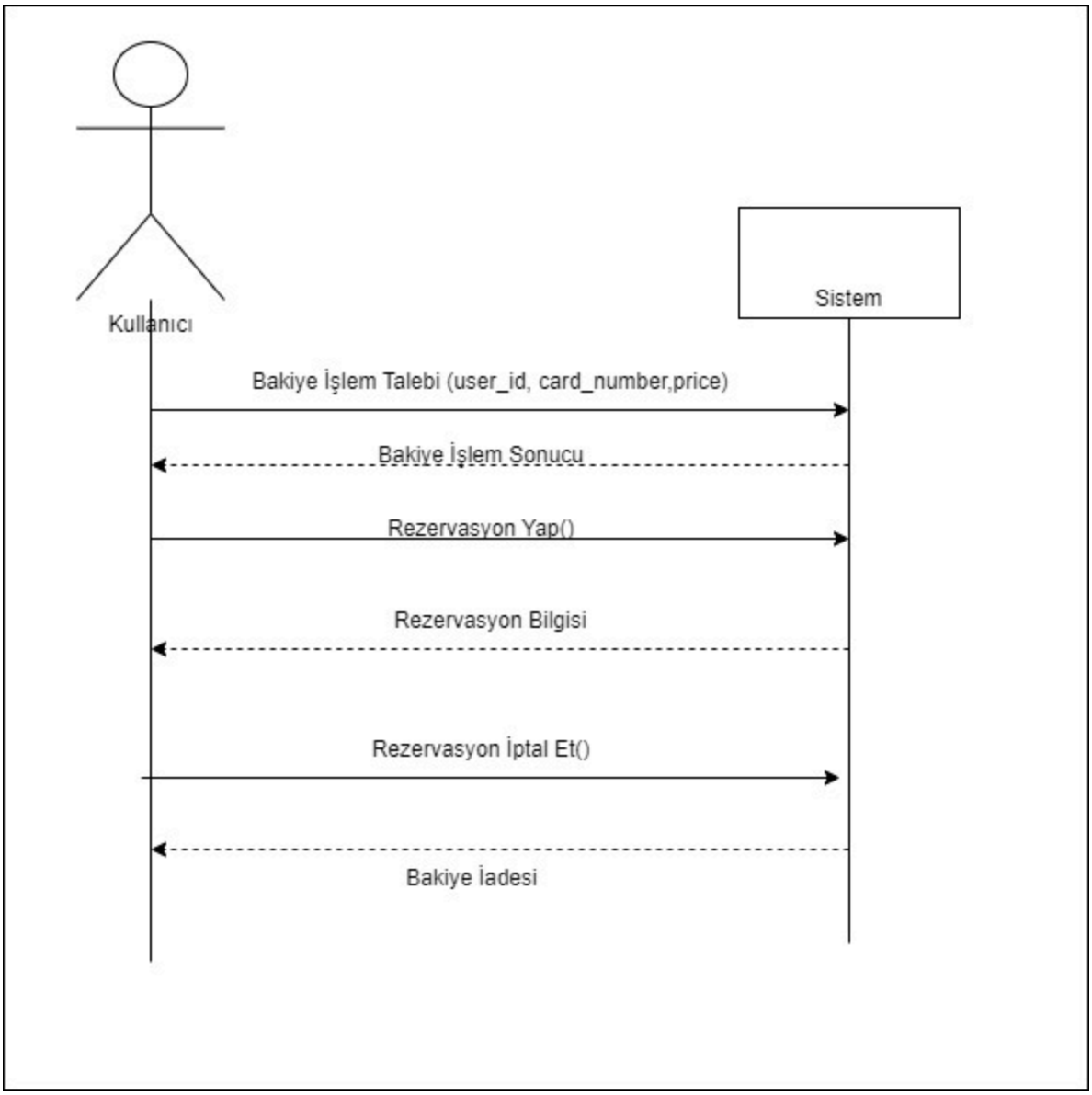
Bu bölümde değerlendirme ile birlikte oluşan kullanım durumlarının bir araya konduğunda oluşacak olan alan modeli ortaya konmalıdır. Bu alan modeli içerisinde konsept olarak bulunan sınıflar ortaya konarak aralarındaki ilişkiler ortaya konmalıdır.



Hafta-5 : Sistem Geçiş Diyagramları

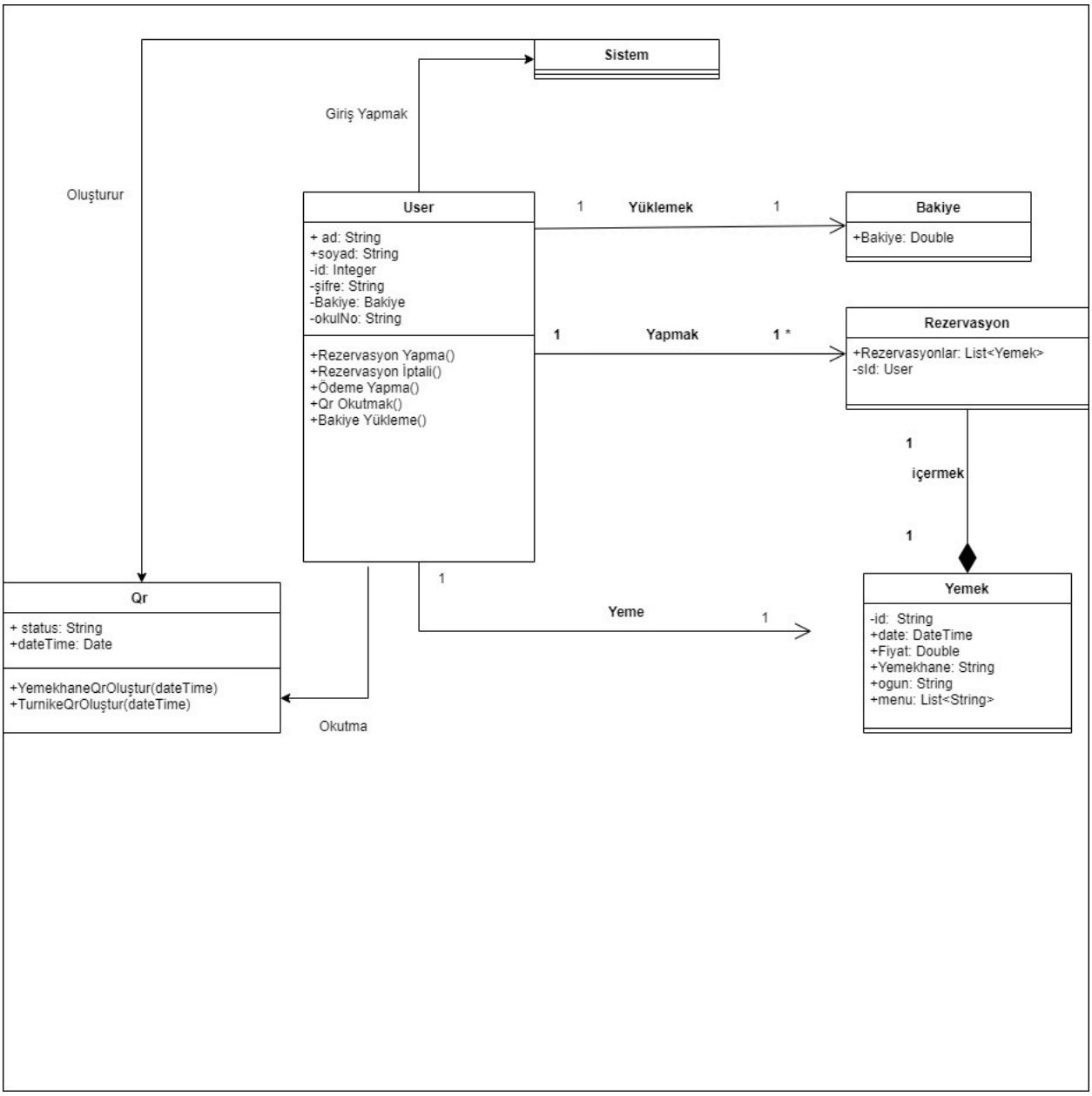
Aktörler, kullanım durumları ve alan modeli içerisideki sınıfların bir araya getirilerek sistem ile oluşan etkileşimler sistem geçiş diyagramları olarak çizilmelidir.




Kullanıcı
Sistem
Bakiye İşlem Talebi (user_id, card_number,price)
Bakiye İşlem Sonucu
Rezervasyon Yap()
Rezervasyon Bilgisi
Rezervasyon İptal Et()
Bakiye İadesi

Hafta-6 : Sınıf Siyagramları

Sistem Geçiş Diyagramları ile ortaya konan sistemin dış aktörler ve yardımcı sistemler ile olan etkileşiminin alt yapısını oluşturacak olan sınıflar alan modeli içerisindeki yapılar ve seçilen programlama dilinin izin verdiği nesneye yönelimli programlama temelinde ortaya konmalıdır. Bu çalışma içerisinde elle çizim ile başlayan süreç artık UML olarak net bir şekilde kodlanabilir netlikte olmalıdır.

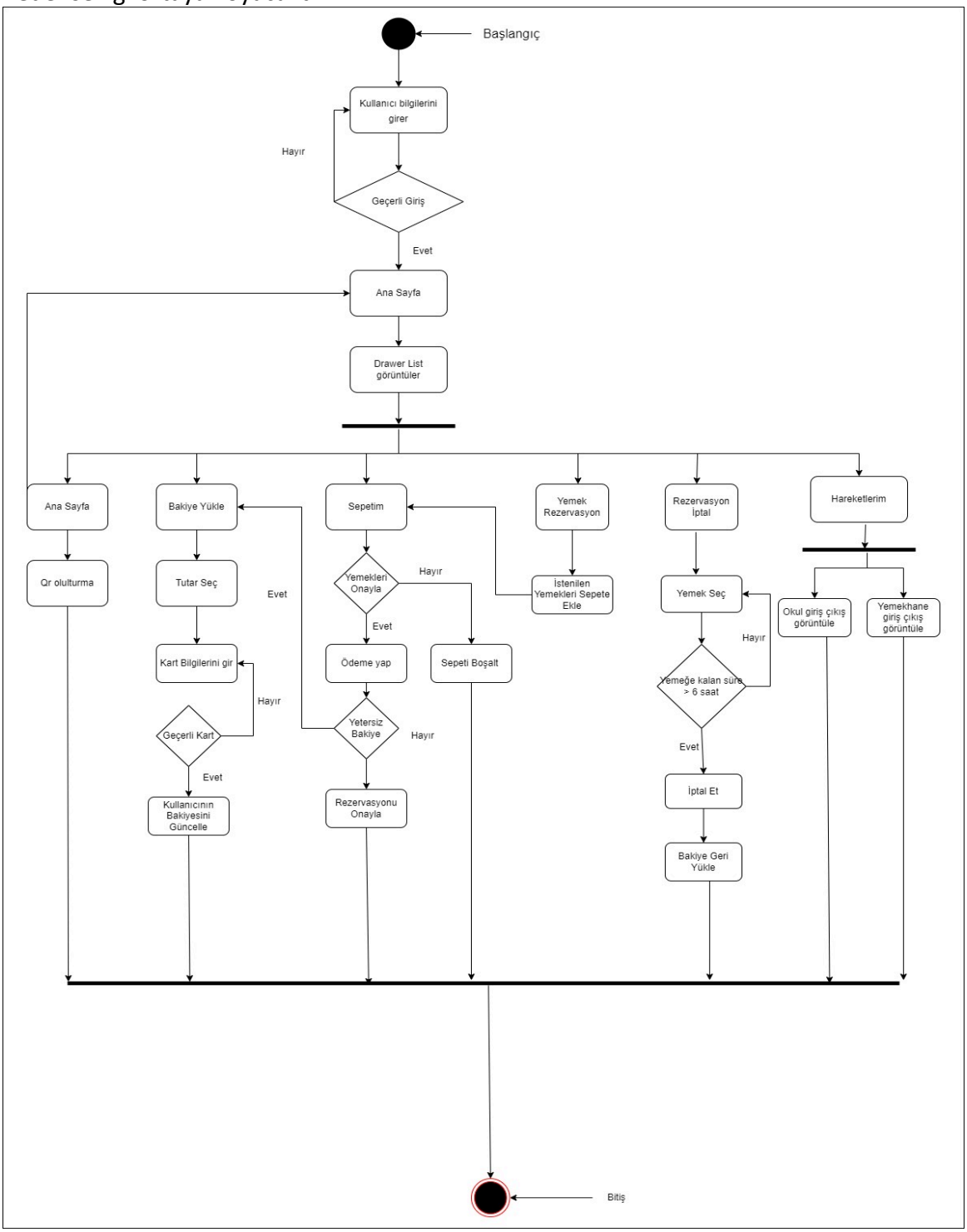


Hafta-7 : Sınıf Etkileşim Diyagramları

Sistem Etkileşim Diyagramlarından gelen sistemin fonksiyonalitesi sınıf etkileşim diyagramları ile gereksinimleri sağlayacak şekilde sınıfların nasıl gelen istekleri işlediği kullanım durumları özelinde ortaya konmalıdır.

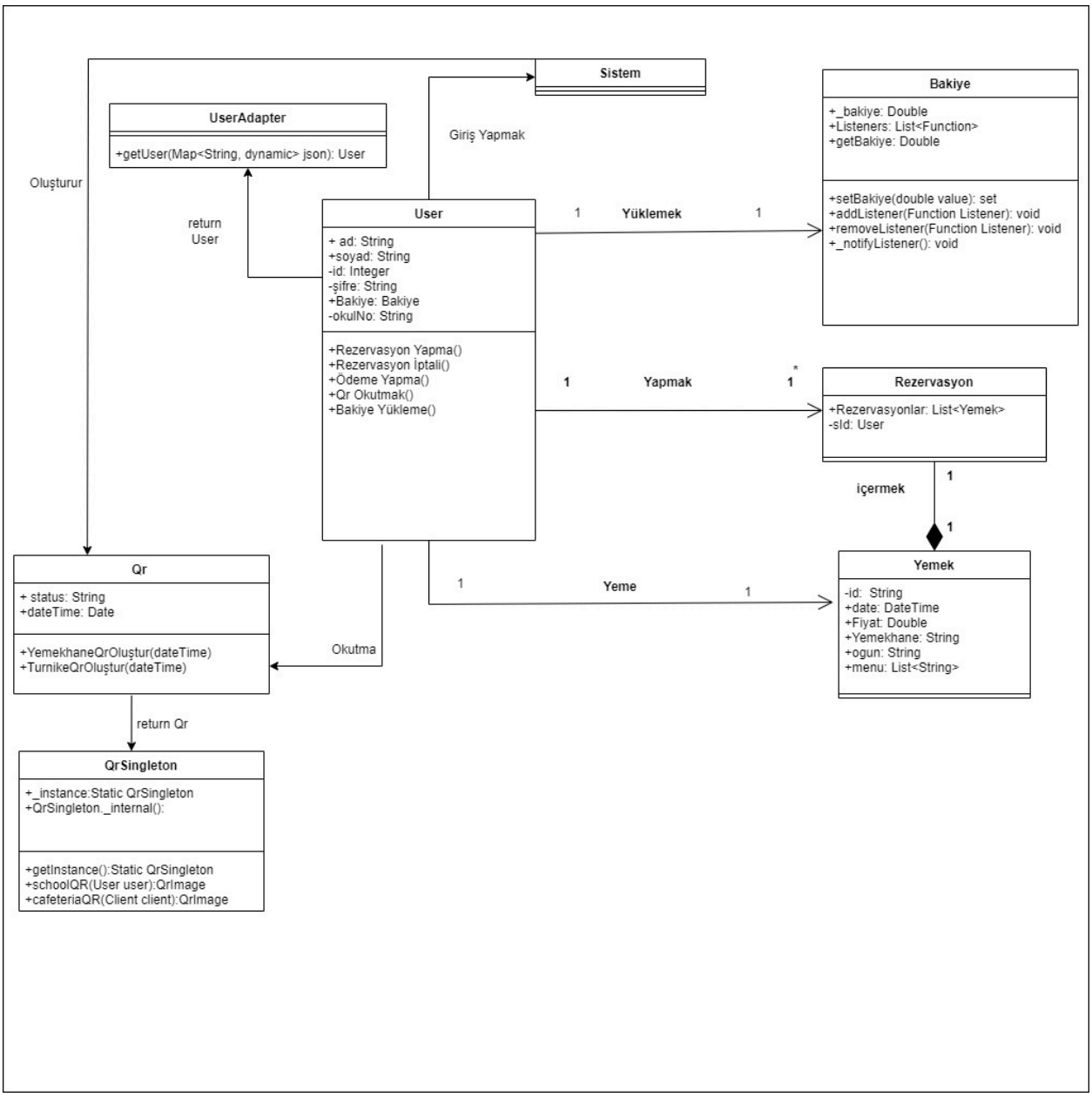
Hafta-8 : Aktivite Diyagramları ve Modelleme

Sınıf etkileşim diyagramları ile ortaya konan kullanım durumları ve sistem etkileşim diyagramları aktivite diyagramları ile var olan tüm sistemin temel bileşenlerinden başlayarak yakalanan farklı aktivitelerin ortaya konması bu bölüm içerisinde anlatılmalıdır. Bu bölüm içerisinde anlatılanlar tasarım desenleri olarak daha sonrasında yenilenen sınıf diyagramlarının ortaya çıkması için nedenselliği ortaya koyacaktır.



Hafta-9 : Tasarım Desenleri

Kullanılan tasarım desenleri, bu tasarım desenlerinin sınıf diyagramı ve aktivite diyagramı ile nasıl kurgulandığı ve oluşan tekil sınıf diyagramı bu bölüm içerisinde tasarım desenleri özelinde açıklanacaktır.



Hafta-10 : Sınıf Uygulanması

Tasarım deseni de içeren sınıf diyagramı ve aktivite diyagramı bu bölüm içerisinde nasıl uygulandığı kod parçacıkları ile açıklanacaktır.

```dart
class QrSingleton {
  static QrSingleton? _instance;

  QrSingleton._internal();

  static QrSingleton getInstance() {
    _instance ??= QrSingleton._internal();
    return _instance!;
  }

  QrImage schoolQR(User? user) {
    return QrImage(
      data:
          "${user!.sId} ${user.name} ${user.surName} ${DateTime.now()} okul ",
      version: QrVersions.auto,
      size: 200.0,
    );
  }

  QrImage cafeteriaQR(User? user) {
    return QrImage(
      data:
          "${user!.sId} ${user.name} ${user.surName} ${DateTime.now()} yemekhane",
      version: QrVersions.auto,
      size: 200.0,
    );
  }
}
```

Singleton pattern, bir sınıfın sadece bir örneğinin oluşabileceğini ve bu örneğin her yerden erişilebileceğini sağlamak için kullanılan bir tasarım kalıbıdır. Bu tasarım kalıbı, bir sınıfın birden fazla örneğinin oluşmasını engelleyerek, bellek kullanımını azaltır ve sınıfın örneği arasındaki senkronizasyon sorunlarını ortadan kaldırır. Ayrıca, bu tasarım kalıbı sayesinde, sınıfın örneği arasında global bir erişim sağlar ve sınıfın örneğinin tek bir yerden yönetilmesini mümkün kılar.

Bu tasarım kalıbı sayesinde QR sınıfının bir tanesi nesnesi oluşur. Fonksiyonlar yardımıyla gelen parametreler sayesinde bu sınıftan üretilecek bir QR'ın eşsiz olması sağlanır.

```dart
class UserAdapter {
  User getUser(Map<String, dynamic> json) {
    String schoolNo = json['no'];
    String name = json['name'];
    String surname = json['surname'];
    String pass = json['pass'];
    String sId = json['sId'];
    Bakiye bakiye = json['bakiye'];

    return User(
      no: schoolNo,
      name: name,
      surName: surname,
      pass: pass,
      bakiye: bakiye,
      sId: sId,
    );
  }
}
```

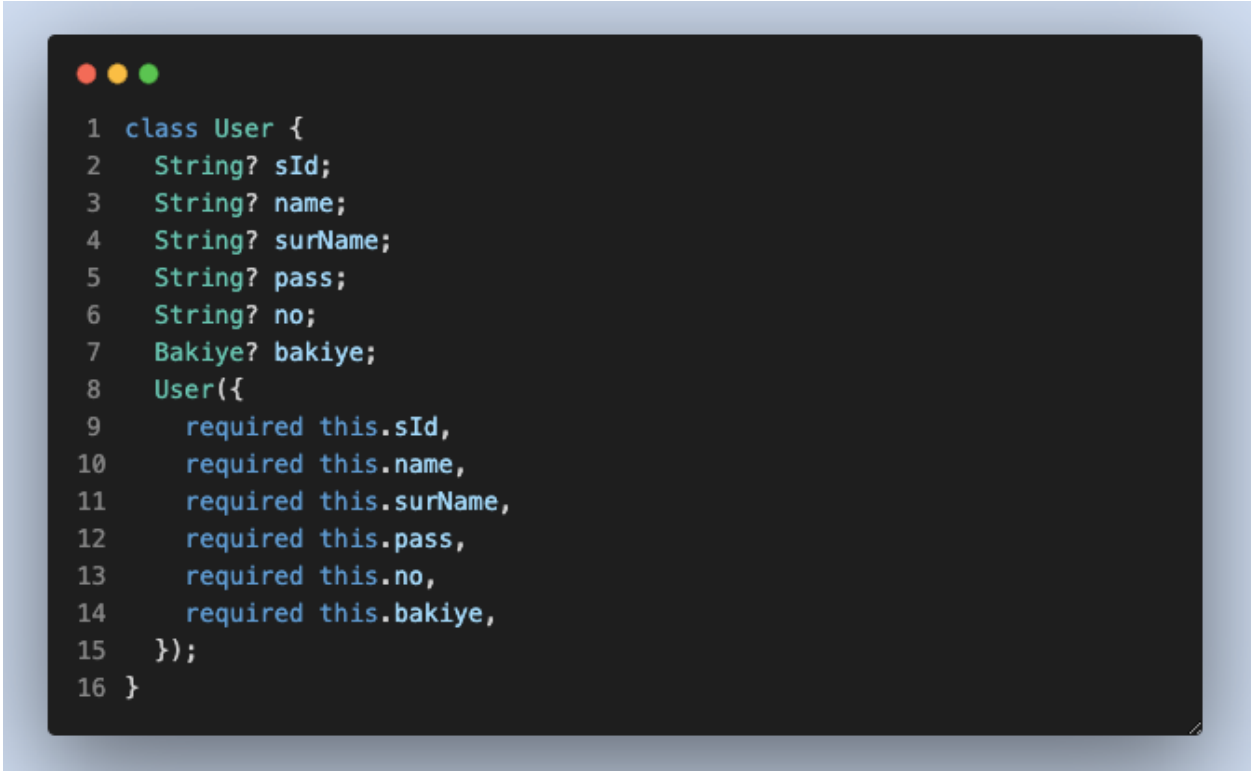
Adapter pattern, bir sınıfın veya nesnenin başka bir sınıf veya nesnenin beklentilerine uymasını sağlamak için kullanılan bir tasarım kalıbıdır. Bu tasarım kalıbı sayesinde, farklı sınıflar veya nesneler arasında uyumlu bir şekilde çalışabiliriz. Özellikle, farklı sistemler arasında veri alışverişi yaparken veya farklı API'lar kullanırken kullanışlıdır. Bu örnekte aşağıda gösterilmiş kodlar sayesinde bu sınıf kullanılarak API'den gelen cevabın UserModel'e set edilmesi gerçekleşir. API'den gelen response aşağıda gösterilmiştir.

```dart
UserAdapter _adapter = UserAdapter();
User user = _adapter.getUser(response.body); //response.body == json

```

```
"client": {
    "_id": "63bc76903743c4e90b1f1139",
    "name": "Bora",
    "surName": "Körpe",
    "pass": "$2a$08$YOV/GFmgLCNpyhRqNcLdZetIZzvceB6735wdrVTwimA9He9Q1PwQm",
    "no": "200601003",
    "bakiye": "172",
    "createAt": "2023-01-09T20:18:24.744Z",
    "__v": 0
}
```


Örnek response body



```dart
class User {
  String? sId;
  String? name;
  String? surName;
  String? pass;
  String? no;
  Bakiye? bakiye;
  User({
    required this.sId,
    required this.name,
    required this.surName,
    required this.pass,
    required this.no,
    required this.bakiye,
  });
}
```

Response kısmında dönen user sınıfı.

```dart
class Bakiye {
  double? _bakiye;
  List<Function> _listeners = [];

  double? get bakiye => _bakiye;

  set bakiye(double? value) {
    _bakiye = value;
    _notifyListeners();
  }

  void addListener(Function listener) {
    _listeners.add(listener);
  }

  void removeListener(Function listener) {
    _listeners.remove(listener);
  }

  void _notifyListeners() {
    _listeners.forEach((Function listener) => listener());
  }
}
```

Observer pattern, bir nesnenin değiştiğinde diğer nesnelerin bu değişiklikleri algılamasını ve bunlara göre hareket etmesini sağlamak için kullanılan bir tasarım kalıbıdır. Bu tasarım kalıbı, bir nesnenin (observable) değiştiğinde, bu nesne ile ilişkili diğer nesnelerin (observer) bu değişiklikleri algılamasını ve bunlara göre hareket etmelerini sağlar. Bu sayede, nesneler arasındaki ilişkileri daha dinamik ve esnek hale getirir. Bu sınıfın fonksiyonları kullanılarak değişen bakiye değeri için bütün observerlar aynı anda değişmiş değeri göstermesi sağlanır. Böylelikle kullanıcı bakiye ile ilgili işlem yaptığında bütün ekranlarda değerler güncel şekilde gösterilir.

```dart
class Rezervasyon {
  User? sId;
  List<Yemek> rezervasyonlar = [];

  void addReservation(Yemek yemek) {
    rezervasyonlar.add(yemek);
  }

  void RemoveReservation(Yemek yemek) {
    rezervasyonlar.remove(yemek);
  }
}
```

```dart
class Yemek {
  String? sId;
  DateTime? date;
  double? fiyat;
  String? yemekhane;
  String? ogun;
  List<String>? menu;
  int? iV;

  Yemek({
    this.sId,
    this.date,
    this.fiyat,
    this.yemekhane,
    this.menu,
    this.iV,
    this.ogun,
  });
}

```

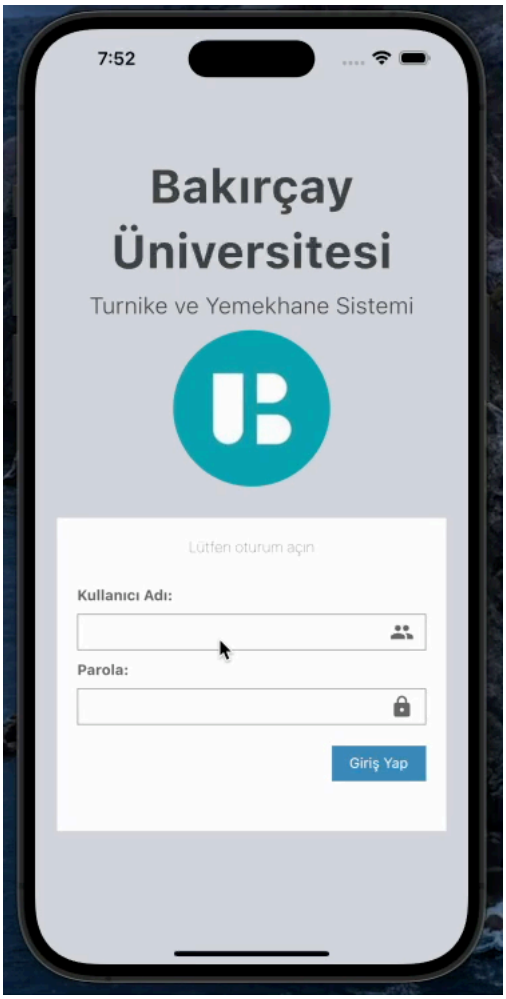
Hafta-11 : Değerlendirme

Proje genelinde ortaya konan içerik, tasarım ve gerçekleştirim bu bölüm içerisinde değerlendirilerek, kabul edilen sınırlamalar, kullanılan kaynaklar ve varolan tasarımlar bu bölüm içerisinde ileriye dönük olarak değerlendirilmelidir.

Bu proje akıllı üniversite kapsamında kampüs yaşantısını daha verimli hale getirmeyi amaçlamaktadır. Dolayısıyla uygulama gerçekleştiriminde seçilen design pattern'leri bu amaca hizmet etmektedir. Proje hayata geçirildiğinde uygulamanın verimliliğine göre ilerleyen zamanlarda uygulama performansını geliştirmek için bu patternler güncelleştirilebilir. Her ne kadar fiziksel öğrenci kartını kaldırmak üniversiteyi belirli bir maliyetten kurtarsa da turnikelerin yenilenmesi ve uygulama geliştirilmesi yeni maliyetler oluşturacaktır. Bu yüzden ortadan kalkan maliyetler ve yeni oluşan maliyetler iyi bir şeklide hesaplanmalıdır. Uygulamanın uzun vadede okula avantaj sağlaması amaçlanmaktadır. Bütün turnikelerin birden değiştirilmesi ve uygulamanın kullanılmaya başlanması başlarda adaptasyon sorunu oluşturabileceğinden önce okulda küçük bir grup üzerinde pilot uygulama yapılması ve daha sonra bütün okula yayılması hedeflenmektedir. Ayrıca yemekhane rezervasyonunun iptal edilmesi özelliği getirileceğinden alınan ücretin bankaya iade edilmesi süreci herhangi bir olumsuzluk yaşanmaması adına dikkatli bir şekilde takip edilmelidir.

Hafta-12 : İterasyon-2 Değerlendirme Toplantısı ve Yapılacaklar

Bu bölüm içerisinde ilk iterasyon sonucunda oluşan tasarımın eksikleri ve bu tasarım eksiklerinin nasıl giderileceği bir takvim ortaya konarak tartışılmalıdır.


7:52
Bakırçay Üniversitesi
Turnike ve Yemekhane Sistemi
Lütfen oturum açın
Kullanıcı Adı:
Parola:
Giriş Yap