Софийски университет “Св. Климент Охридски”

Факултет по математика и информатика

Софтуерни архитектури и разработка на софтуер

Курсов проект

DronesPark

Изготвили:

Мая Росенова Бораджиева, ФН 62335, група 5

Петя Стефанова Ляскалиева, ФН 62297, група 4

гр. София

2020г.

Съдържание на документа

Въведение

Обща информация за текущия документ

1. Предназначение на документа

В документа е представена архитектурата на софтуерната система за следене и управление на свободните паркоместа DronesPark.

2. Описание на използваните структури на архитектурата

- Декомпозиция на модулите
    - Показва разделението на системата на модули и подмодули. В декомпозицията на модулите е показана и връзката между отделните модули. Системата е разделена на 6 основни модула, всеки от които е особено важен за правилното ѝ функциониране. Те са: **Server**, **Web App**, **Mobile App**, **Drones**, **Database** и **Backup Database**. Сървърът съдържа модулите **Drones Logic**, **Parking Manager**, **Weather Service**, **Maps Service**, **Payment System Service**, **Authentication Service**, **Authorization Service**, **Database Manager**. Най-важни от тях за функционалността на системата са: **Drones Logic** и **Parking Manager**. Сървърът комуникира с клиентите на системата чрез неговия презентационен слой, а с базите от данни чрез **Database Manager**. Базата от данни съдържа информация за всички необходими записи (records), с които да се осигури адекватната работа на приложението и сайта. **Mobile App** и **Web App** имат модули, които предоставят възможност на потребителите да използват системата. Връзката на мобилното приложение и сайтът със сървъра се осъществява чрез **DronesPark API** и **REST API** респективно.
- Структура на процесите
    - Структурите показват както процеса на работа на дроновете, така и процеса на използване на приложението от потребителите. Те целят да покажат нагледно функционалностите на системата, т.е. какви са възможностите на потребителя, като са съобразени с изискванията 6, 8, 10 и 15.
- Структура на внедряването
    - Структурата на внедряването показва как са разпределени софтуерните и хардуерните компоненти, необходими за работата на системата. Елементите на структурата са хардуерни устройства и комуникационни канали. Показани са специфичните

архитектурни решения, като например използването на две бази от данни (основна и вторична) с цел бързодействие и подсигуряване на наличност във всеки един момент, както и връзката между тях (синхронизацията).

3. Структура на документа
    - Въведение (секция 1)
    - Декомпозиция на модулите (секция 2)
        - Общ вид на декомпозицията на модулите
        - Контекстна диаграма
        - Подробно описание
        - Описание на възможните вариации
    - Структура на процесите (секция 3)
        - Първично представяне
        - Описание на елементите и връзките
        - Описание на обкръжението
        - Описание на възможните вариации
    - Структура на внедряването (секция 4)
        - Първично представяне
        - Описание на елементите и връзките
        - Описание на обкръжението
        - Описание на възможните вариации
    - Архитектурна обосновка (секция 5)
    - Допълнителна информация (секция 6)

Общи сведения за системата

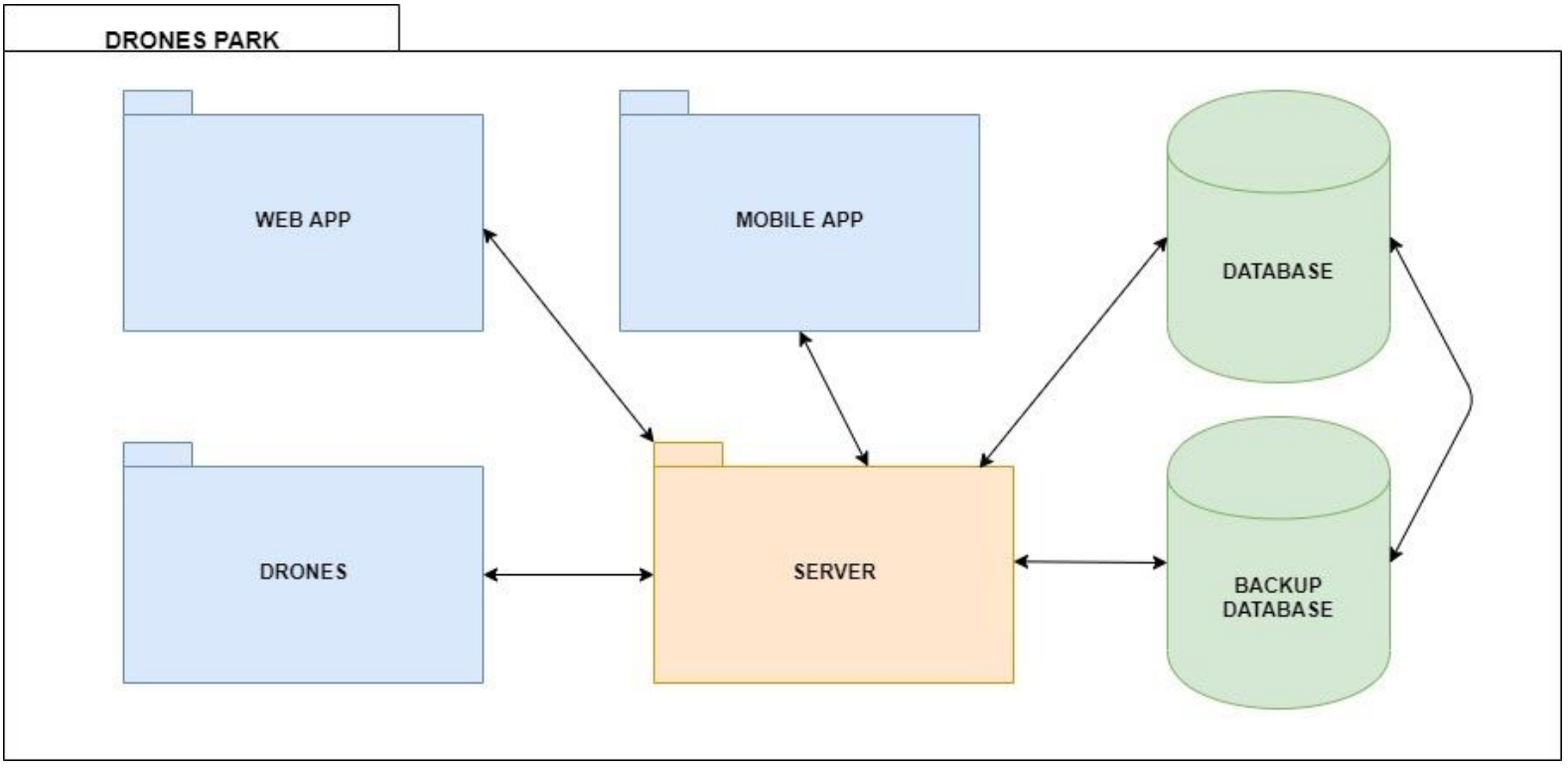
В последните години намирането на места за паркиране в големите градове се превръща в голямо предизвикателство. Приложението DronesPark цели да разреши именно този проблем, като подпомага своите потребители в намирането на свободни места и като им дава възможност да се абонират за място по избор. Качествените изисквания, които са застъпени в системата, са: сигурност (security), availability, отказоустойчивост, performance, backup, extensibility, maintainability, testability.

Терминологичен речник

- ★ **Декриптиране -** процесът на "отключване" на криптирани файлове.
- ★ **Криптиране -** процесът на кодиране на информацията по начин, който предотвратява достъпа на неоторизирани лица до нея.
- ★ **MAVLink** - е протокол за съобщения за комуникация с дронове (и между бордовите компоненти на дронове).
- ★ **RESTful API** - Application Program Interface (API), използващ HTTP заявки за GET, PUT, POST и DELETE на данни.

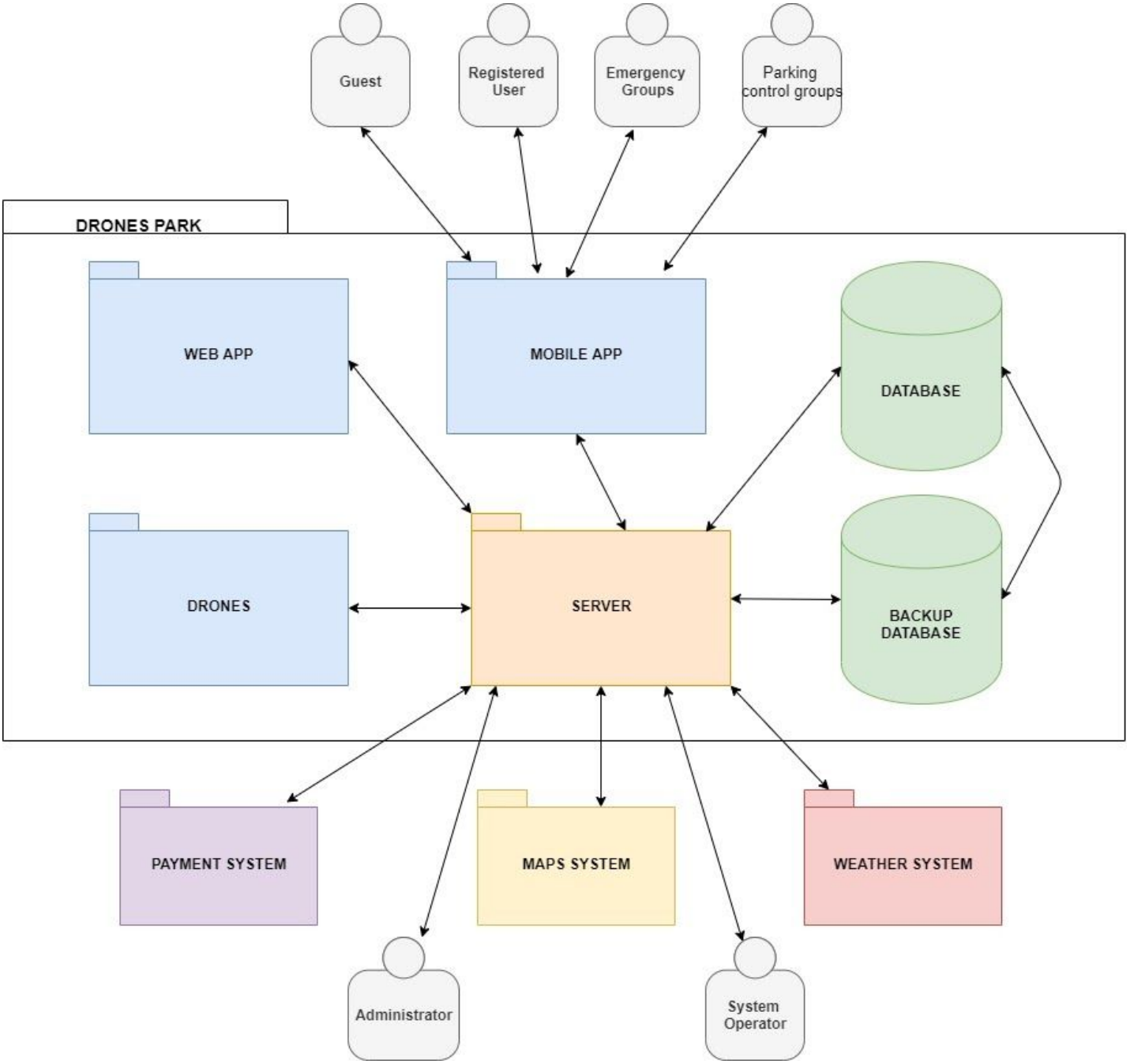
Декомпозиция на модулите

Общ вид на декомпозицията на модули за системата



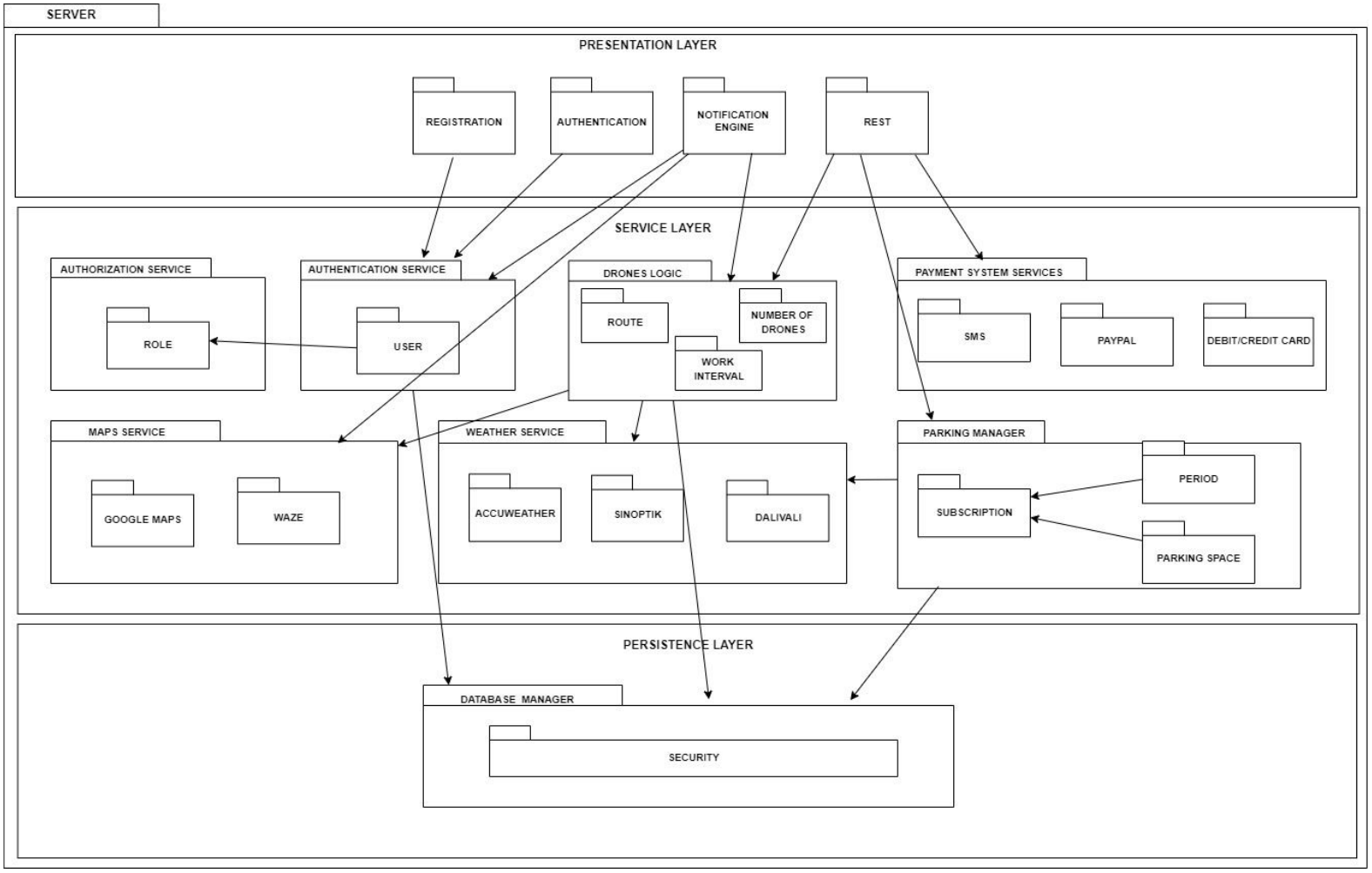
За реализирането на системата е използвана т.нар. клиент-сървър архитектура. Сървърът представлява връзката между базата данни и клиентите. Клиенти в системата са мобилното приложение, сайтът и системата от дронове. Базата от данни съдържа всички необходими данни, за да работи коректно системата.

Контекстна диаграма



В контекстната диаграма са представени външните услуги и връзката им със системата. Също и как потребителските групи използват системата, например чрез мобилното приложение и др.

SERVER



Предназначение на модула
Модулът представлява връзката между базата данни, мобилното приложение, сайтът и системата от дронове. Тук се съдържа основната логика на системата.

Основни отговорности на модула в системата
Задачата на модула е да предостави основните функционалности на системата. Именно затова бива разделен на три слоя. **Presentation слоят** е отговорен за комуникация с клиента (мобилното приложение, сайтът и системата от дронове). **Service слоят** съдържа бизнес логиката и служи за връзка с външните услуги (Weather, Maps & Payment Services). **Persistence слоят** комуникира с базите от данни (Primary & Backup Database).

Описание на интерфейсите на модула

- **Drones Logic**

Тук се съдържа логиката относно определянето на броя на летящите в момента дронове и маршрутът им, на базата на предвиждане за честотата на заемане/освобождаване на места в съответните зони. Това предвиждане зависи от натрупаните данни за динамиката на паркиране в съответния ден и час от седмицата (предоставени от **Database Manager**) и метеорологичните условия (от **Weather Services**).

public int getNumberOfDrones(double parkingDynamics, String weatherInfo) throws ReducedVisibilityException

- ❏ Входни данни - процент на заетите паркоместа в съответния ден и час от седмицата и информация за метеорологичните условия
- ❏ Изходни данни - колко на брой летящи дронове са необходими в момента
- ❏ Семантика - За изчисляване какъв да е броя на летящите дронове в момента.
- ❏ Грешки и изключения:

ReducedVisibilityException - При намалена видимост (напр. мъгла) трябва да се извести оператора на системата.

- ❏ Зависимости от други елементи - Използва се Database Manager & Weather Services.

public Queue<String> getDronesRoute(Map<String, Double> parkingDynamicsByArea)

- ❏ Входни данни - мап от районите и съответния процент заети паркоместа в съответния ден и час от седмицата
- ❏ Изходни данни - какъв да е маршрутът на летящите дронове – опашка от стрингове за реда на обхождане на районите
- ❏ Семантика - За изчисляване какъв да е маршрута на летящите дронове в момента.
- ❏ о Зависимости от други елементи - Използва се Database Manager и Maps Service

public String getState()

- ❏ Изходни данни- информация за състоянието на дроновете.
- ❏ Грешки и изключения- При грешка предоставя предполагаемия район на дрона и трябва да се изпрати съобщение до аварийните групи.

- **Parking Manager**

Тук се съдържа логиката за заемането на свободните паркоместа.

public Set<Integer> getParkingSlotsWithSubscribtion()

- ❏ Изходни данни - номерата на всички паркоместа с абонамент
- ❏ Семантика – дава ни паркоместата, за които е заплатен абонамент и те се водят като заети независимо дали има паркиран автомобил на тях.
- ❏ Зависимости от други елементи - Използва се Database Manager, за да се получи информация от базата данни за абонаментите.

public double getParkingSlotsPrice (double parkingDynamics, String weatherInfo)

- ❏ Входни данни- процент на заетите паркоместа в съответния ден и час от седмицата и информация за метеорологичните условия
- ❏ Изходни данни- цената на свободните паркоместа. Може и да са безплатни.
- ❏ Семантика – За изчисляване цената на свободните места за паркиране, на базата на предвиждане, за честотата на заемане/освобождаване на места в съответните зони (предоставени от Database Manager) и метеорологичните условия (от Weather Services).
- ❏ Зависимости от други елементи - Използва се Weather Services & Database Manager

- **DataBase Manager**

Модулът е отговорен за връзката между сървъра и базата данни. Той е от водещо значение, защото ако настъпи проблем с главната база (Primary DataBase), модулът трябва да има механизъм, с който да използва данните от вторичната база данни (Backup DataBase). При нормална работа DataBase Manager отговаря за изпращането на данни до модулите Drones Logic (натрупаните данни за динамиката на паркиране в съответния ден и час от седмицата), Parking Manager (абонаментите), Authorization Service и Authentication Service (регистрираните потребители, техните роли в системата и техните данни).

public List<String> encrypt(User toEncrypt)

- ❏ Входни данни - информация за потребителя, която трябва да бъде криптирана
- ❏ Изходни данни - криптирана информация
- ❏ Семантика – криптира се конфиденциалната информация на потребителя

- ❏ Зависимости от други елементи - Използва се Authentication Service.

public boolean isSynchronized()

- ❏ Изходни данни - връща дали информацията между двете бази данни е синхронизирана
- ❏ Семантика - проверява дали двете бази данни са синхронизирани
- ❏ Зависимости от други елементи - Използват данните от Database & Backup Database.

public void uploadDronesImages(List<Object> photos) throws FailedUploadException

- ❏ Входни данни - списък от снимки, заснети от дроновете на паркоместата
- ❏ Семантика - използва се за качване на заснетите от дроновете снимки в базата от данни
- ❏ Зависимости от други елементи - Използват данните от Drones.
- ❏ Грешки и изключения:
  - ❏ FailedUploadException - При неуспешно качване на снимките

public void registerUser(User user) throws InvalidInformationException

- ❏ Входни данни - информация за потребителя
- ❏ Семантика - информация за новорегистрирал се потребител, която трябва да бъде записана в базата от данни
- ❏ Зависимости от други елементи - Използва се Authentication Service.
- ❏ Грешки и изключения:
  - ❏ InvalidInformationException - При невалидна информация от страна на потребителя

- **Notification Engine**

Модулът отговаря за изпращането на известия до съответните потребителски групи, използващи мобилното приложение, като всяка потребителска група получава различен тип известия. Той използва информацията от Authorization Service модула, за да определи кой да извести, както и причината за известието. Най-вече причините са – неизправности при дроновете (уведомяват се аварийните групи), както и лоши метеорологични условия, т.е. е невъзможно заснемането на паркоместа (известява се оператора на системата). Поради тази причини модулът Notification Engine комуникира и с Drones Logic, като взима информация за състоянието на дроновете (getState()), както и при неблагоприятни условия метода getNumberOfDrones(..) хвърля ReducedVisibilityException и тук той се обработва, като се известява оператора на системата.

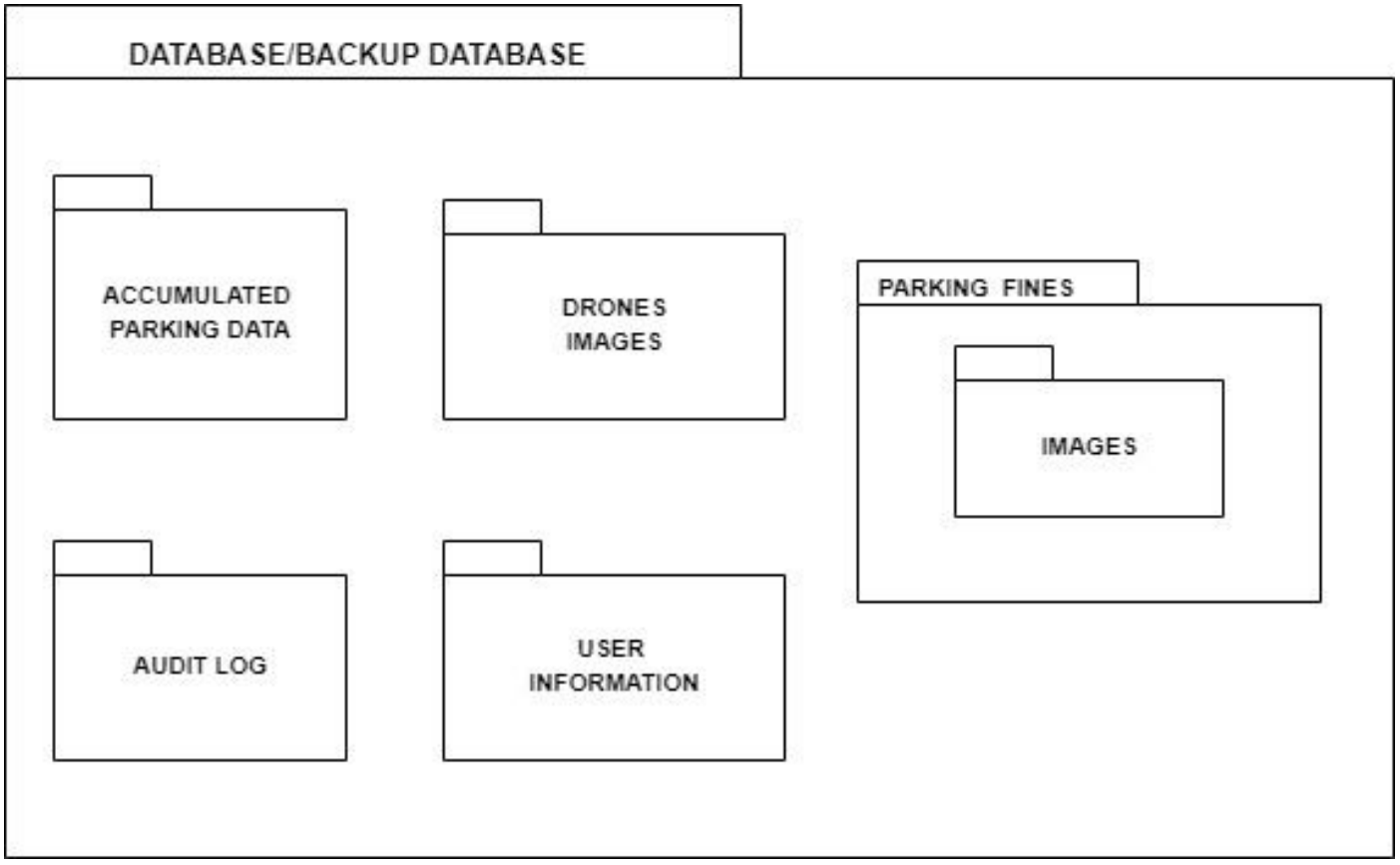
public void sendMessage(String message, Role role) throws MessageFailedException

- Входни данни - съобщението, което се изпраща и на кои групи потребители
- Семантика - за изпращане на съобщение до потребителите с определена роля.
- Зависимости от други елементи - Използва се Drones Logic, Authorization Service
- Грешки и изключения:
  - MessageFailedException - При проблем с изпращането на съобщението.

public void sendMessage(String message, Role role, double radius) throws MessageFailedException, UserNotFoundException

- Входни данни - съобщението, което се изпраща, на кои групи потребители и радиус на обхват на съобщението
- Семантика - за изпращане на съобщение до потребителите с определена роля (в случая аварийните групи), като се посочва радиус на известяване, т.е. до кои потребители да се простира съобщението (най-близките).
- Зависимости от други елементи - Използва се Drones Logic, Authorization Service
- Грешки и изключения:
  - MessageFailedException - При проблем с изпращането на съобщението.
  - UserNotFoundException - Няма намерени аварийни групи в посочения радиус

DATABASE & BACKUP DATABASE



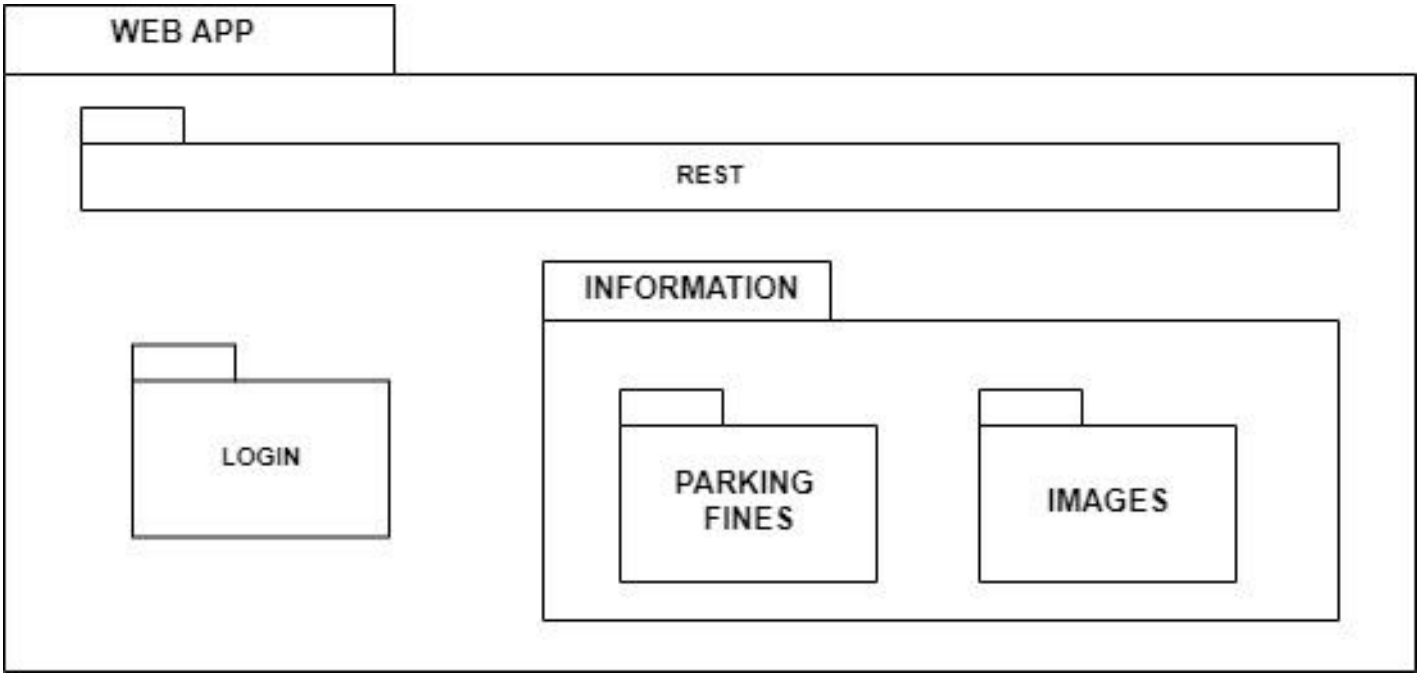
Предназначение

- ❏ Използваме втора база данни, за да подсигурим наличността на данните във всеки един момент. (Active Redundancy)
- ❏ Двете са идентични по съдържание, като има постоянно синхронизиране между тях.

Основни отговорности

- ❏ Двете бази от данни съхраняват данните за потребителите (**User Information**), фишовете и снимките на нарушенията (**Parking Fines**), последните данни от свободните места (**Drones Images**), **Audit Log,** както и натрупаните данни за динамиката на паркирането (**Accumulated Parking Data**), чрез което се предвижда броя на летящите дронове и маршрута им.

WEB APP



Предназначение

- ❏ Считаме, че сайтът е различен по съдържание и функционалности от мобилното приложение.
- ❏ Сайтът играе ролята на регистър за направените нарушения, като за достъп до съответната снимка и фиш не е необходима регистрация, а само попълване на съответните лични данни. Използваме RESTful API за комуникацията със сървъра (извличането на данни).

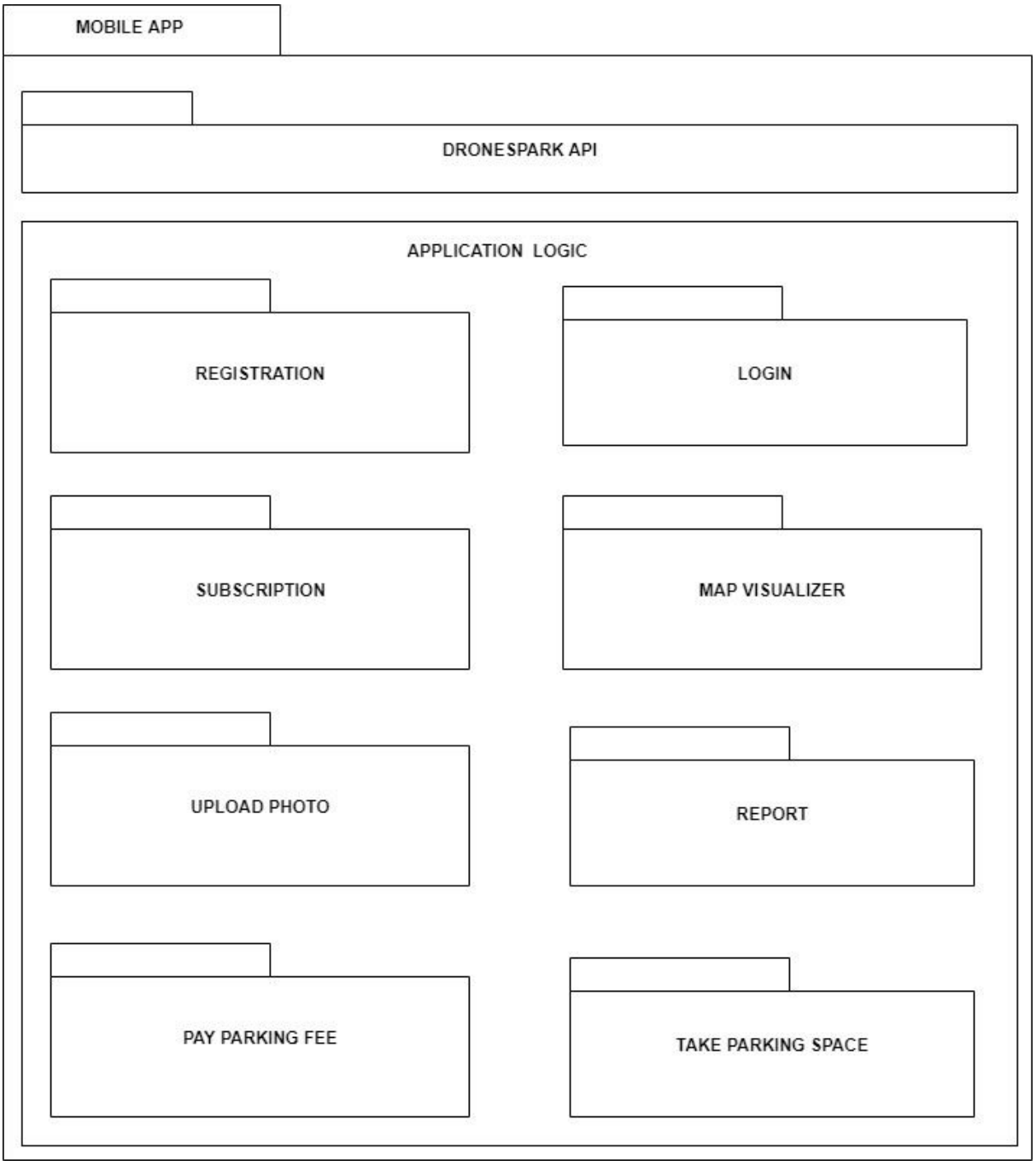
Основни отговорности

- ❏ Дава информация за направените нарушения от съответното лице, използващо сайта.

Описание на интерфейсите

- ❏ Графичен потребителски интерфейс, който позволява потребителите да достъпят информацията за направените от тях нарушения.

MOBILE APP



Предназначение

- ❏ Управление и достъп до потребителски профил.
- ❏ Заемане на свободно паркомясто и възможност за заплащане и абонамент за него.

- ❏ Докладване за неправомерно заето от друг паркомясто, за което има абонамент и качване на снимка на нарушителя.
- ❏ Използване и достъп до карти с цел визуализиране на свободните паркоместа.

Основни отговорности

- ❏ Модулът Drones Park API служи за обмяна на информация между сървъра и мобилното приложение.

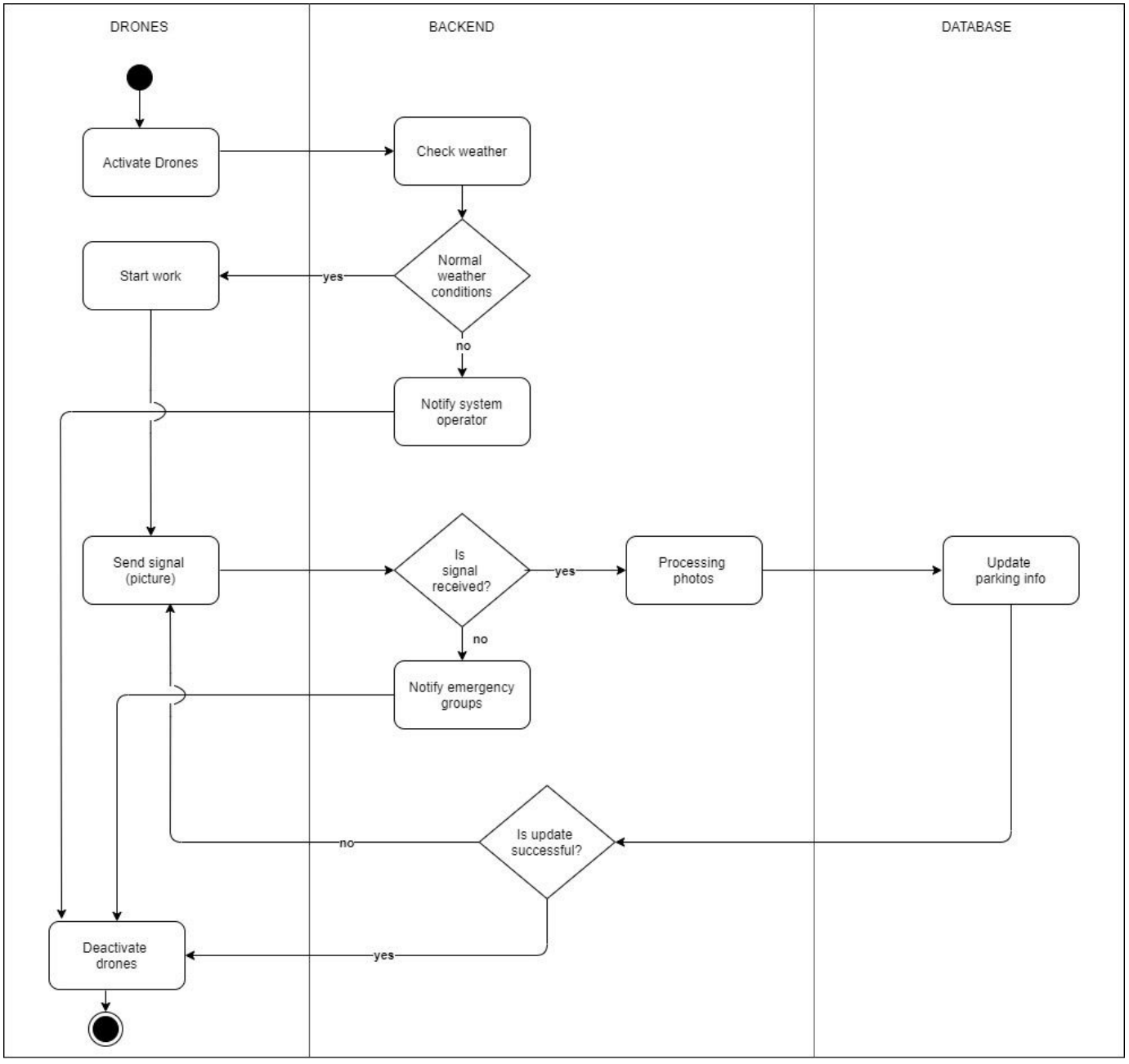
Описание на интерфейсите

- ❏ Графичен потребителски интерфейс, който позволява на обикновените потребители, регистрираните потребители, аварийните групи и групите по контрол на паркирането да използват всички функционалности на системата.

Описание на допълнителните структури

Избрахме долупосочените структури, понеже целяхме да изобразим изискванията към системата DronesPark. На фигура 1 сме взели предвид изискване 6 и изискване 8. На фигура 2 - изискване 10, а на фигура 3 - изискване 15.

1. Структура на процесите



Фиг. 1

На диаграмата от Фиг. 1 са илюстрирани последователността от процесите, протичащи при нормална работа на дроновете. В случай на извънредни обстоятелства са показани и какви действия се предприемат. Задачата на дроновете в нашата система е да изпращат сигнали (снимки от района, в който са разпределени) и нашата система да си послужи с тези снимки, като извлече информация за свободните в момента паркоместа. Тази информация е

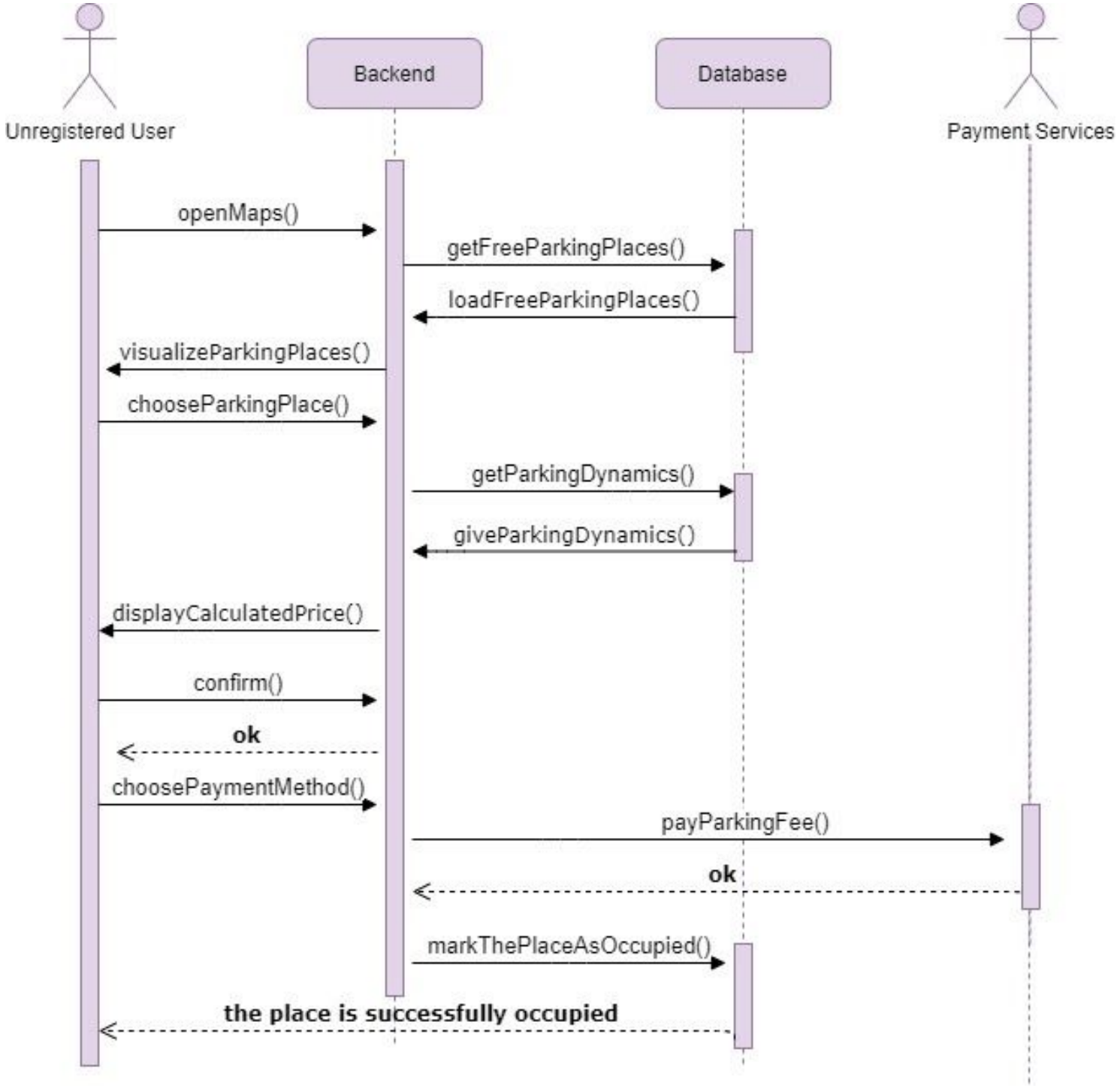
необходима, защото чрез нея се визуализират на картата в приложението свободните места и защото тя служи за съставяне на статистика за динамиката на паркиране в съответния ден и час, което пък от своя страна е ключово за логиката за определяне на броя летящи дронове и маршрутът им. Тази логика се помещава в модула **Drones Logic**.

Описание на елементите и връзките

Този процес се изпълнява от модула **Drones Logic**, който определя работата на дроновете. Дроновете получават сигнал от нашата система да се задействат. **Backend**-ът проверява информацията за времето чрез **Weather Service**. При нормални метеорологични условия дроновете получават инструкции (район за снимане) и започват работа. При условия на трайно намалена видимост (напр. мъгла) се известява оператора на системата и дроновете се деактивират. В противен случай дроновете в ход на работа изпращат сигнали (снимки) до системата. Ако системата не получи сигнал от някой дрон, то има вероятност дронът да е повреден. В такива случаи се изпраща съобщение с информация за предполагаемия район на дрона до **аварийните групи**, които да отстранят проблема. Дронът и в този случай се деактивира. При успешно получен сигнал, информацията за паркирането от снимката се записва в базата данни и системата ни проверява дали качването да се брои за успешно, защото снимката може да не е била достатъчно ясна, че да може да се извлече информация. Ако информацията е записана и е пълна, дронът се деактивира. В случай, че информацията е непълна, дронът получава инструкции за нова снимка.

Описание на обкръжението

Оценката на това дали дроновете да започнат работа или не, става въз основа на информацията за метеорологичните условия, която се предоставя от външни услуги за прогноза за времето.



Фиг. 2

Тук е представен процесът по заемане на свободно паркомясто, за който не е нужно потребителят да е регистриран. Представили сме случая, в който паркоместата се таксуват динамично.

Описание на елементите и връзките

Този процес се изпълнява от модула Parking Manager, който съдържа логиката за заемането на свободните паркоместа. Потребителят използва тази функционалност чрез мобилното приложение на системата. Първо е необходимо да отвори картите. Отзад Backend-а взима информацията за свободните паркоместа от базата от данни и ги представя на картата. Когато потребителят си избере и маркира свободно паркомясто, Backend-а взима информацията за динамиката на паркиране и връща на потребителя изчислената цена на паркомястото. Потребителят следва да потвърди и да избере начин за плащане (SMS, PayPal или дебитна/кредитна карта). Информацията за плащането се обработва от съответната услуга за плащане и системата маркира в базата данни даденото паркомясто като заето. За финал съобщение за успешно заето място бива изпратено до потребителя.

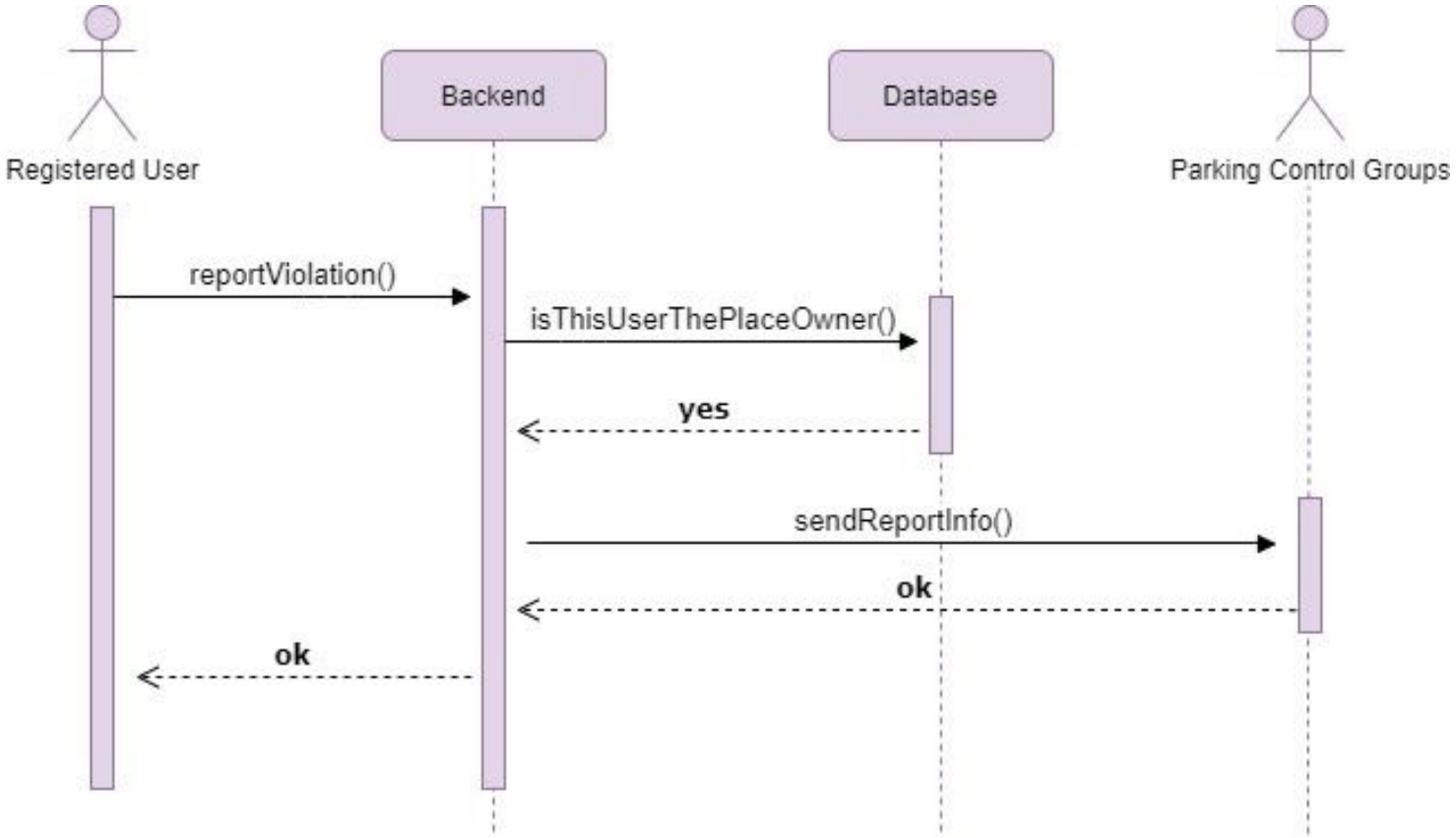
Описание на обкръжението

Изборът на свободно място се осъществява чрез карти със свободните места, които се предоставят от външна услуга за карти. Заплащането на избраното паркомясто става чрез външна услуга за плащане.

Описание на възможните вариации

Към системата могат да се добавят и други начини на разплащане в бъдеще.

Свободните места за паркиране може да са безплатни или да се таксуват динамично.



Фиг. 3

На диаграмата от Фиг. 3 е представен процесът по подаване на сигнал до групите по контрол на паркирането за неправомерно заето от друг място, за което са абонирани.

Описание на елементите и връзките

За да може потребителят да уведоми групите по контрол на паркирането за нарушението, първо трябва да е влезнал в системата (т.е. да е регистриран). Регистрираният потребител изпраща сигнал, след което Backend-ът проверява дали потребителят наистина е абониран за съответното място в базата от данни. Ако това е така, тогава сигналът достига до групите по контрол на паркирането и потребителят получава съобщение, че успешно е уведомил за нарушението.

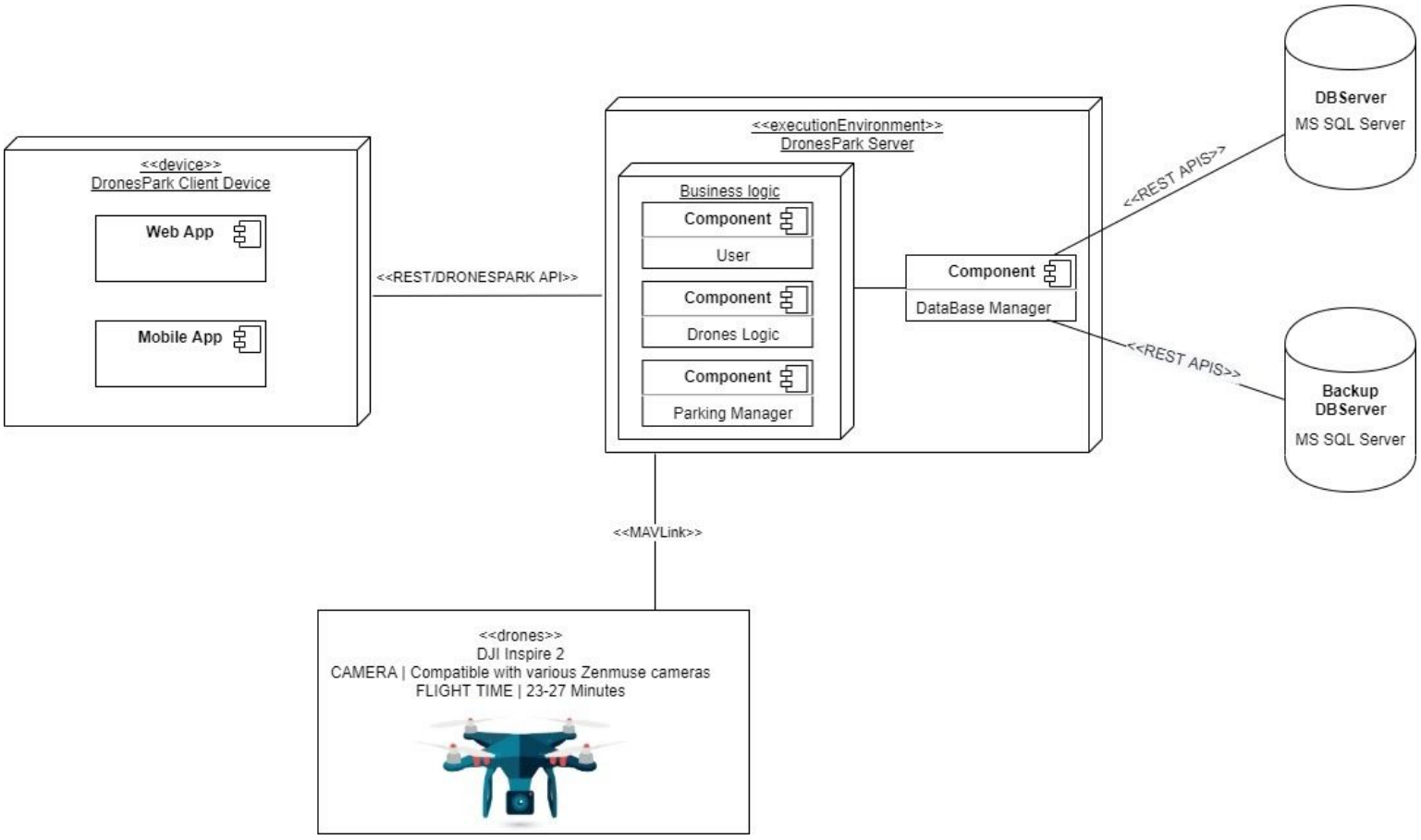
Описание на обкръжението

Това бива възможно, понеже регистрираният потребител има опцията ‘Report’, която се свързва с модула Notification Engine. В този модул се изяснява причината за сигнала и съответната потребителска група, която трябва да бъде известена. Проверява се информацията в базата от данни - дали потребителят наистина е абониран за това място, тоест дали информацията е достоверна, преди да уведоми групите по контрол на паркирането.

Описание на възможните вариации

Към опцията Report може да се добави и опция ‘Прикачи снимка’, с която да се докаже нарушението в случай, че нарушителят избяга преди да пристигнат групите по контрол на паркирането.

2. Структура на внедряването



Фиг. 4

На диаграмата от Фиг. 4 е представена структура на внедряването на системата DronesPark. Структурата е подходяща, защото е необходимо да се покаже как хардуерните и софтуерните компоненти взаимодействат помежду си.

Описание на елементите и връзките

Представен е сървърът като хардуерна част, съдържаща в себе си софтуерните компоненти, отговорни за най-важните функционалности на системата. Сървърът е ключовото звено между всички елементи на DronesPark.

Осъществява връзката с клиента - с Web App съответно чрез REST API, а с Mobile App - чрез Drones Park API. Клиентската част се грижи за предоставянето на потребителски интерфейс на системата - браузър или мобилно устройство.

Дроновете са най-съществената част в системата. Те също трябва да комуникират със сървъра, като комуникацията се осъществява чрез MAVLink. За целите на системата сме избрали определен модел дронове - DJI Inspire 2, понеже времето за полет 23-27 минути е подходящо (желателна е честа профилактика на дроновете преди полет), а камерата също съответства на нуждите.

Използваме две бази от данни, за да подсигурим наличността на данните във всеки един момент. Двете са идентични по съдържание, като има постоянно синхронизиране между тях. Синхронизацията се осъществява чрез компонента DataBase Manager, която също така е отговорна за евентуалния достъп до вторичната база от данни, ако настъпи проблем с основната. Сървърният компонент DataBase Manager е основната връзка на базите със сървъра, като комуникацията се осъществява чрез REST APIs. Избрали сме Microsoft SQL Server за сървърна система за управление на бази от данни.

Архитектурна обосновка

Изисквания по системата

1. Свободните паркоместа се идентифицират от система от дронове, които обикалят града и заснемат зоните за паркиране (отгоре).
   - ➔ В системата дроновете са представени като “клиенти”, които си комуникират със сървъра. Снимките на заснетите паркоместа се запазват в базата от данни, като за идентифицирането им като свободни или заети се грижи модула **Parking Manager**. Мястото на дроновете в системата е показано в Декомпозиция на модулите и Структурата на внедряването.

2. Броят на летящите в момента дронове и маршрутът на всеки от тях се определя динамично, на базата на предвиждане, за честотата на заемане/освобождаване на места в съответните зони. Това предвиждане зависи от натрупаните данни за динамиката на паркиране в съответния ден и час от седмицата и метеорологичните условия.

➔ Натрупаните данни за динамиката на паркиране се пазят в базата от данни като **Accumulated Parking Data.** А информацията за метеорологичните условия е достъпна чрез модула **Weather Services.** Логиката относно определянето на броя на летящите в момента дронове (**Number Of Drones**) и маршрутът им (**Route**) се съдържа в **Drones Logic**, използващ споменатите преди това модули. Разположението на тези модули се намира в Декомпозиция на модулите.

3. За определяне на метеорологичните условия да се ползва външна услуга за прогноза за времето.

➔ Информацията за метеорологичните условия е достъпна чрез модула **Weather Services**, който представлява код за връзка с външните услуги за прогноза за времето. Weather Services се намира в Service Layer-а на сървъра (Декомпозиция на модулите).

4. Системата използва специфичен алгоритъм за разпознаване на свободните места, на база на заснетите изображения.

➔ Този специфичен алгоритъм се намира в модула **Parking Manager**, който е отговорен именно за разпознаването на свободните паркоместа (Декомпозиция на модулите).

5. Системата поддържа следните групи потребители:
a. Администратор
b. Оператор
c. Аварийни групи
d. Групи по контрол на паркирането (т.нар. „паяци“)
e. Регистрирани потребители
f. Обикновени потребители

➔ Всеки потребител (**User**) има дадена роля в системата, която се определя от **Role** в **Authorization Service**. За всяка роля има определен списък с права (permissions). В базата от данни се съхраняват данните на потребителите (име, имейл, криптирана парола и т.н.), както и съответните им роли в системата. Връзката между потребителите и ролите може да проследи в Декомпозиция на модулите.

6. Ако някой дрон излезе от строя, незабавно трябва да се уведомят аварийните групи, които да получат информация за предполагаемия район, в който се намира дрона и да отстранят повредата.

➔ На фигура 1 от структура на процесите е показан случаят при неполучен сигнал от дрона и последващото уведомяване на аварийните групи. Това уведомление се изпраща чрез **Notification Engine** модула, който е отговорен да обработи причината за възникналия проблем и уведомяването на съответната група за него. Чрез външната услуга **Maps Services** се изпраща и предполагаемия район, в който се намира дронът.

7. Информацията за свободните места се обновява на определен интервал от време, който се задава от оператора на системата и може да е най-малко 1 минута.

- ➔ В **Drones Logic** се съдържа и интервалът от време (**Work Interval**), който казва на какъв интервал от време да се активират дроновете и съответно да се обновява информацията за свободните паркоместа. Той се задава от оператора на системата.

8. При трайно намалена видимост (напр. мъгла), която води до невъзможност да се заснемат паркоместата, да се вземат мерки за известяване на оператора на системата.

- ➔ На фигура 1 от структура на процесите е показан случаят при лоши метеорологични условия, при което дронът не може да заснеме паркоместата, и последващото уведомяване на оператора на системата. Вземането на информация за времето се извършва от **Weather Services**. Известието до оператора се изпраща чрез **Notification Engine** модула, който е отговорен да обработи причината за възникналия проблем и уведомяването на съответната група за него.

9. Регистрираните потребители могат да заплащат абонамент за определено парко-място, което се маркира като заето в рамките на периода на абонамента, независимо дали заснетите от дроновете изображения, показват наличието на автомобил на него или не.

- ➔ Модулът **Parking Manager**, освен, че е отговорен за разпознаването на свободните места чрез специфичен алгоритъм, предоставя и възможност за правенето на абонаменти от регистрираните потребители. Подмодулите **Subscription**, **Period** и **Parking Space** показват за колко време е абонаментът за определено място, като то се счита за заето в рамките на този период. Модулът **Subscription** в мобилното приложение отговорен за заявяването на абонамент от потребителя. Модулите, свързани с абонирането за паркоместа, се намират в **Mobile App** и **Server** в Декомпозиция на модулите.

10. Ако няма абонамент, свободните места за паркиране може да са безплатни или да се таксуват динамично, като цената се определя според предвиждане за честотата на заемане/освобождаване в съответния ден/час, както и от прогнозата за времето.

- ➔ Във фигура 2 е показан процесът по заемане на свободно паркомясто. Цената на паркоместата се определя динамично от:
  - ◆ Натрупаните данни за динамиката на паркиране се пазят в базата от данни като **Accumulated Parking Data.**
  - ◆ Вземането на информация за времето се извършва от **Weather Services**.

Плащането на изчислената сума, ако мястото не е определено като безплатно, се извършва от модула **Payment Services**, а заявяването за плащане се извършва с помощта на модула **Pay Parking Fee** от мобилното приложение. Модулът **Parking Manager** маркира мястото като заето.

11. Плащането може да се извършва чрез дебитна/кредитна карта, PayPal или СМС, като в бъдеще може да се добавят и други начини на разплащане.

- ➔ Модулът **Payment Services**, който е код за връзка с външните услуги за плащане. На този етап той съдържа модулите **PayPal**, **SMS**, **Debit/Credit Card**, като варианти за плащане. Отделянето му в самостоятелен модул позволява евентуалното добавяне на нови начини на разплащане.

12. Обикновените потребители, регистрираните потребители, аварийните и групите по контрол на паркирането използват системата през мобилно приложение, като може да заемат само свободните места, за които няма абонамент.

- ➔ Заемането на свободно паркомясто се осъществява чрез модула **Take Parking Space** в мобилното приложение. А в сървъра **Parking Manager** проверява дали за мястото вече няма абонамент и ако няма го маркира като заето.

13. Останалите потребители трябва да имат 100% защитен от външна намеса достъп до системата.

- ➔ Модулът **Security** в **Database Manager** криптира и декриптира конфиденциалната информация като потребителски пароли и номера на дебитни/кредитни карти.
- ➔ **Authentication Service** подсигурява, че потребителят е този, за когото се представя.
- ➔ **Authorization Service** подсигурява, че потребителят има права да извърши дадено действие.
- ➔ С цел следене на състоянието на системата и евентуално идентифициране на атакуващия използваме **Audit Log** в **Database**, който пази информация за различни събития, свързани със сигурността, които са настъпили в системата ни. Достъпът до **Audit Log**-а е строго ограничен и до него има достъп само администратора.

14. Групите по контрол на паркирането следят дали няма нарушители (неплатили или заели място за което нямат абонамент). При засичане на нарушител, освен принудителното преместване на автомобила, заснемат настъпилото събитие, като снимката се съхранява директно в системата и след това се издава електронен фиш за глоба. Снимката и фишът трябва да са достъпни и през публичен сайт, който се зарежда чрез уеб-браузър.

- ➔ Качването на снимката, показваща нарушението, става чрез модула **Upload Photo** в **Mobile App.** Информацията за нарушението (**Parking Fines & Images**) се съхраняват в базата данни и са достъпни през публичния уеб сайт (**Web App**) в модула **Information**. С оглед на това, че информацията е конфиденциална, за достъпа до нея е необходимо потребителят да въведе личните си данни (**Login**).

15. Регистрираните потребители може да подават сигнал до групите по контрол на паркирането за неправомерно заето от друг място, за което са абонирани.

- ➔ Това изискване е представено във фигура 3 от структурата на процесите.

16. Системата да работи 100% без отказ в рамките на светлата част на работния ден (9:00 до 17:00 зимно време и 8:00 – 19:00 лятно време).

➔ За да се подсигури 100% наличност на системата, сме използвали тактиката за активен излишък (Active redundancy). Именно за това са представени две бази данни **Primary & Backup Database** в Структурата на внедряването.Те се синхронизират помежду си с помощта на **Database Manager**.

17. Системата да поддържа архив на данните за динамиката на паркирането и всички издадени фишове за глоби за 25 години назад във времето, както и архив на заснетите изображения за 3 години назад.

➔ В Database се пази сортирана по дата информация за заснетите нарушения и издадените фишове (**Parking Fines & Images**). Данните за динамиката на паркирането се пазят в **Accumulated Parking Data**. Снимките от дроновете се пазят в **Drones Images**.

Архитектурни драйвери

1. Свободните паркоместа се идентифицират от система от дронове, които обикалят града и заснемат зоните за паркиране (отгоре).
Това е функционално изискване и отговаря на въпроса какво ще прави системата. То влияе особено на архитектурата, понеже е важно да се изясни как дроновете ще си комуникират с останалата част от системата. Логиката по заемането на паркоместата се основава на информацията, която се получава от дроновете.

3. За определяне на метеорологичните условия да се ползва външна услуга за прогноза за времето.
Системата ни разчита на външна система - услуга за прогноза за времето, от която се разчита за това до каква степен може да е налична системата от дронове (availability). Използването на външни услуги влияе на архитектурата, защото трябва да се определи кои модули ще обменят информация с тази услуга.

5. Системата поддържа следните групи потребители:
a. Администратор
b. Оператор
c. Аварийни групи
d. Групи по контрол на паркирането (т.нар. „паяци“)
e. Регистрирани потребители
f. Обикновени потребители
Това ни дава информация за това какви потребителски групи трябва да има в нашата система.

9. Регистрираните потребители могат да заплащат абонамент за определено парко-място,което се маркира като заето в рамките на периода на абонамента, независимо дали заснетите от дроновете изображения, показват наличието на автомобил на него или не. Трябва от началото да имаме предвид, че имаме определена група, за която се отнася това изискване. Да се вземе предвид, че се състои плащане и че дроновете трябва да отчитат местата като заети.

11. Плащането може да се извършва чрез дебитна/кредитна карта, PayPal или СМС, като в бъдеще може да се добавят и други начини на разплащане. Трябва да се предвиди, че при определени условия, ще се таксуват потребителите на услугата(използващите паркоместа) и също така е важно, че системата трябва да е extendable (да има възможност за добавяне на нови неща в бъдеще). Тук отново се използват външни услуги. Следователно това влияе на архитектурата, защото трябва да се определи кои модули ще обменят информация с тази услуга.

12. Обикновените потребители, регистрираните потребители, аварийните и групите по контрол на паркирането използват системата през мобилно приложение, като може да заемат само свободните места, за които няма абонамент.
Явява се като драйвер, понеже тук става ясно, че съответните групи ще използват системата чрез мобилно приложение, откъдето следва, че трябва да се предостави такова и да се предвиди достъпността му за тези групи.

13. Останалите потребители трябва да имат 100% защитен от външна намеса достъп до системата.
Това е важно по отношение сигурността на потребителската информация (макар и да се подразбира, важно е да се предвиди в архитектурата).

14. Групите по контрол на паркирането следят дали няма нарушители (неплатили или заели място за което нямат абонамент). При засичане на нарушител, освен принудителното преместване на автомобила, заснемат настъпилото събитие, като снимката се съхранява директно в системата и след това се издава електронен фиш за глоба. Снимката и фишът трябва да са достъпни и през публичен сайт, който се зарежда чрез уеб-браузър. Трябва да се предвиди ситуацията, в която някой нарушава установената система, както и съответните мерки за таксуване от страна на нашия софтуер и публикуването на електронния фиш и снимката като доказателство в сайта.

16. Системата да работи 100% без отказ в рамките на светлата част на работния ден (9:00 до 17:00 зимно време и 8:00 – 19:00 лятно време). Защото се разчита, че системата ще е налична на 100% в най-натовареното време. (availability, отказоустойчивост)

17. Системата да поддържа архив на данните за динамиката на паркирането и всички издадени фишове за глоби за 25 години назад във времето, както и архив на заснетите изображения за 3 години назад. Защото съхранената информация служи за “гаранция”, че нарушител е отразен правилно и че е издаден фиш.

Допълнителна информация

ТАКТИКИ

Тактика за изправност (Активен излишък) -> Използваме втора база данни (BACKUP DATABASE), за да подсигурим наличността на данните във всеки един момент.

Тактика за изправност (Heartbeat/Keepalive) -> Дроновете периодично изпращат информация към сървъра и така се разбира дали те са в изправност. Сигналът служи не само като heartbeat, а също и за изпращане на данни към сървъра (снимки), както и за района, в който се намира. Ако сигналът не бъде получен, то следва да се уведомят аварийните групи.

Тактика за сигурност (Автентикация и оторизация на потребителите) -> Чрез автентикация се цели да се провери дали потребителят е този, за когото се представя. Чрез оторизацията се гарантират различните права на потребителските групи.

Тактика за сигурност (Audit Log) -> С цел следене на състоянието на системата и евентуално идентифициране на атакуващия използваме Audit Log, който пази информация за различни събития, свързани със сигурността, които са настъпили в системата ни. Достъпът до Audit Log-а е строго ограничен и до него има достъп само администратора.