Drones Park

Курсова работа по Софтуерни архитектури и разработка на софтуер

Съдържание

Въведение

Обща информация за текущия документ

Документът има за цел да представи архитектурата на приложението Drones Park

Описание на използваните структури

- Декомпозиция на модулите:

  Показва разпределението на системата на модули и съответните подмодули, като по този начин се предоставя по-ясна визия върху това какви са основните функционалните изисквания, разпределението на кода, изграждащ системата, как ще повлияе евентуално промяна на останалата час от системата. Освноните модули са **User Managament, Drones Managament, Drones, API Module, Mobile App.**

- Структура на внедряването**:**

  Показва разпределенито на различните модули върху съответните хардуерни или софтуерни системи. Със използването на тази структура се показват решения, взети на базата на качествените изисквания (**наличност, сигурност**).

- Структура на процесите:

  Показва основните функционалности на системата, като е насочена главно към потребителите и заинтересовани лица, които не са свързани с разработката на сисистемата. Целта е да се придобие по-задълбочено разбиране за това как се използва системата.

Структура на документа

- **Въведение** (секция 1)
- **Декомпозиция на модулите** (секция 2)
  - Общ вид на декомпозицията
  - Контекстна диаграма
  - Подробно описание на всеки модул
- **Структура на внедряването** (секция 3)
  - Първично представяне
  - Описание на елементите и връзките
  - Описание на възможните вариации

- **Структура на употребите** (секция 4)
    - Първично представяне
    - Описание на елементите и връзките
- **Структура на процесите** (секция 5)
    - Първично представяне
    - Описание на елементите и връзките
- **Архитектурна обосновка** (секция 6)

Общи сведения за системата

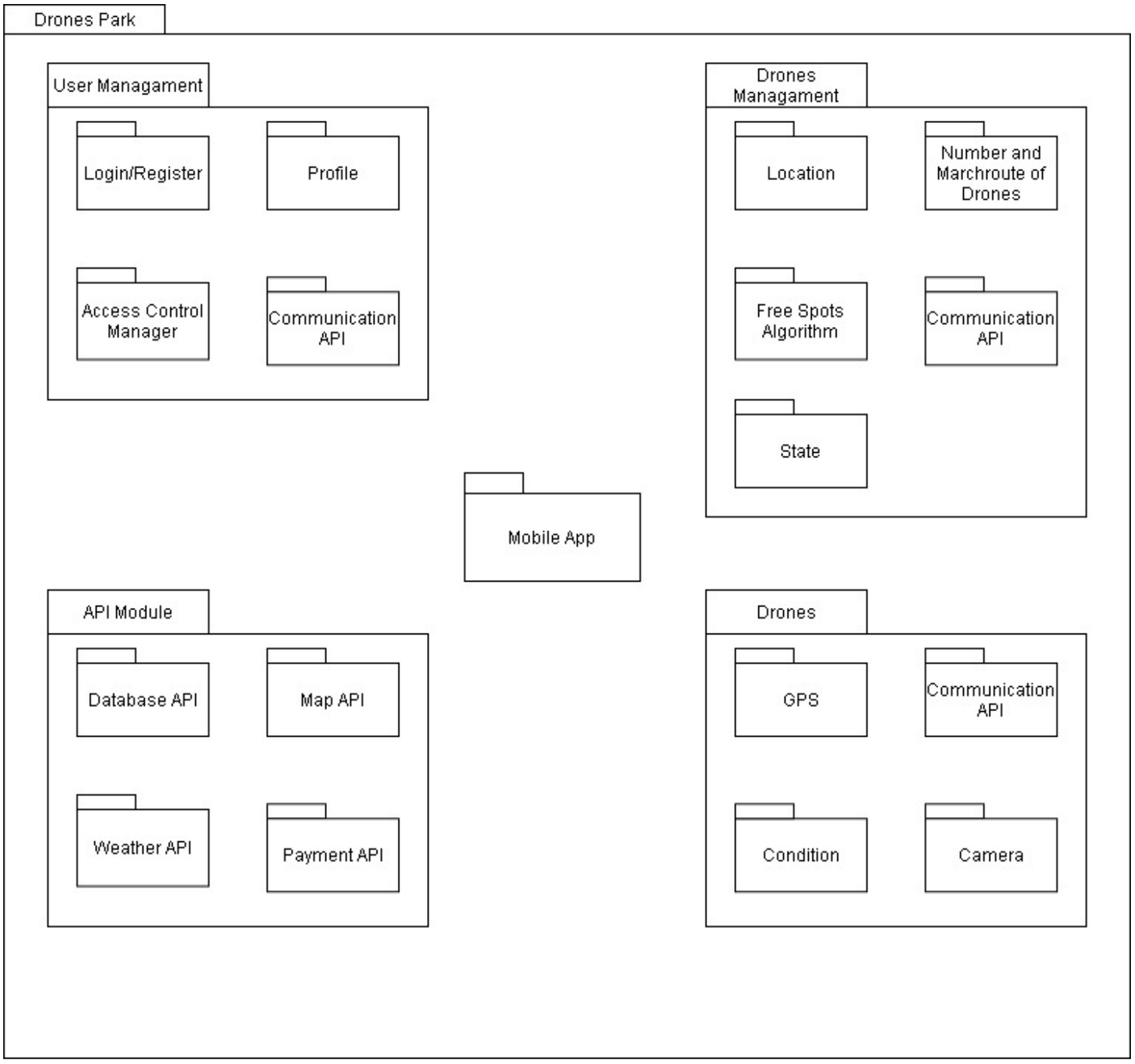
Drones Park е приложение, целящо да улесните гражданите, използващи личните си автомобили, в задачата за намиране на свободни паркоместа. Всеки шофьор е запознат с неприятната задача да търси свободно място, когато закъснява за работа или за друго важно събитие. Чрез система от летящи дронове, подаващи информация за свободните места, задачата на шофьорите се улеснява максимално, като нужно единствено да изберат свободно място, което да резервират, без да се налага да обикалят из тесните централни улици на града. Освен ползите, които извличат потребителите, приложението спомага за по екологичната градска среда, тъй като спестява работата на двигателя в излишно обикаляне на улици. Системата ще бъде налична на 100% в рамките на светлата част на работния ден, а личните данни на потребителите ще бъдат на 100% защитени от външен достъп.

Терминологичен речник

- **API** – Application Programming Interface (посредник между две системи)
- **Интерфейс** – Споделена граница,служеща за обмяна на информация между отделни компоненти
- **UML** – Unified Modeling Language
- **База данни** – съвкупност от логически свързани данни, структурирани по определен начин
- **Криптиране** – преобразуване на информация според определен алгоритъм
- **Външна система –** отдалечена софтуерна система, служеща като доставчик на информация или функционалност
- **Модул –** логически обособена софтуерна единица

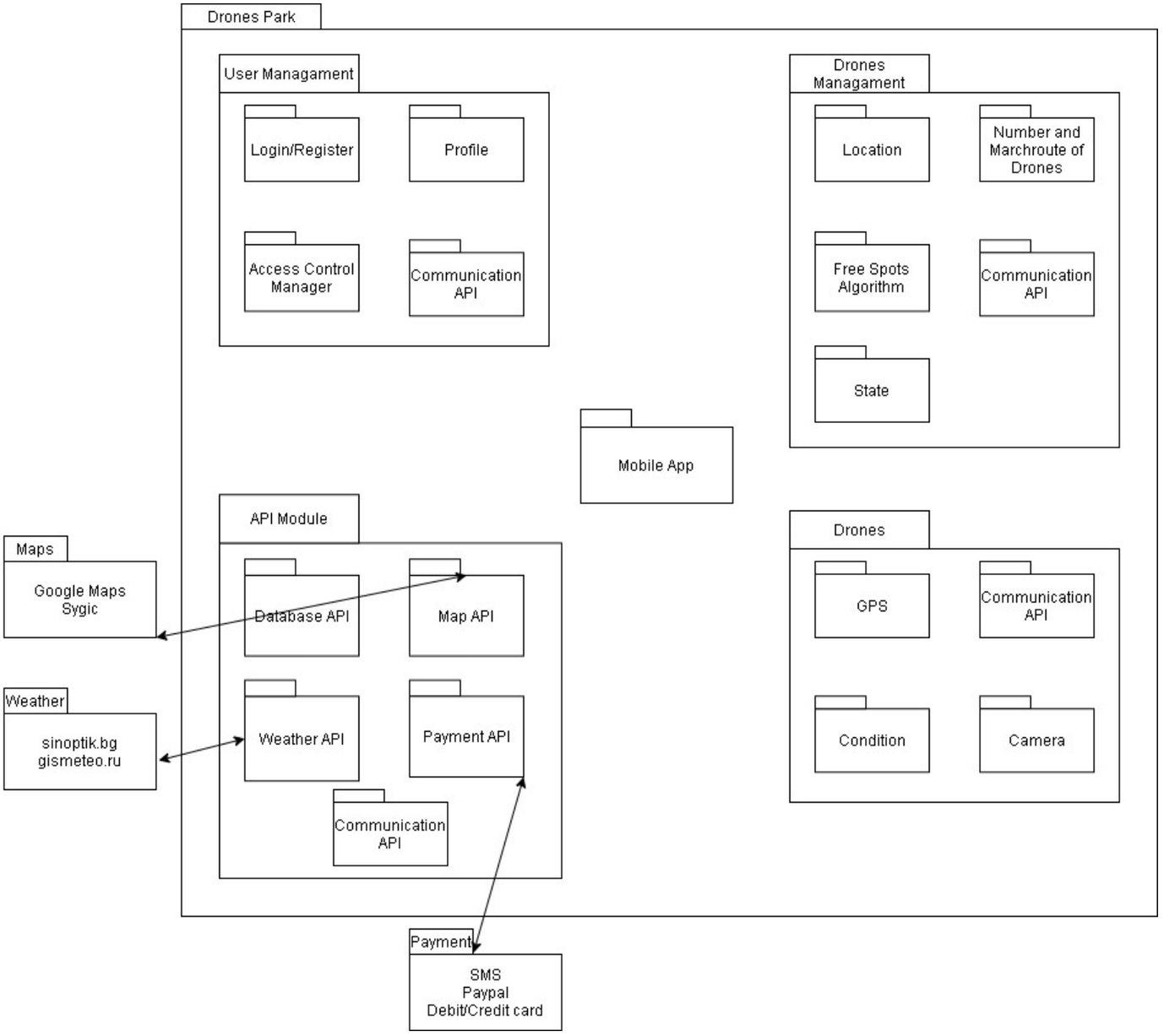
Декомпозиция на модулите

Общ вид на декомпозицията на модули за системата



Диаграмата показва модулната декомопозиция. Всеки модул отговаря за определени функционалности на системата, като подмодулите са тясно свързани или отговарят за определена задача.

Контекстна диаграма



Отделните модули на системата комуникарат помежду си чрез **Communication API**. За връзката с външни системи е обособен отделен модул, отговарящ за всички API, установяващи връзката с тях.

Payment API: Предоставя възможност за плащане чрез използването на външни системи.
Map API: Предоставя на системата възможност за използване на различни видове карти**.**
Weather API: Предоставя на системата връзка с външен източник за климатичните условия в средата на работа за дроновете.
Database API: Предоставя интерфейс за работа с базата данни на приложението, като са съобразени всички установени норми за сигурността на базата данни.

Подробно описание на всеки модул

User Managament

- Предназначение:
  Модулът предоставя на потребителите различните функционалности, до които съответния потребител има достъп
- Основни отговорности на модула в системата:
  Основните отговорности са свързани с обслужването на потребителите и защита на личните им данни
- Описание на интерфейсите на модула:
  - void Register(string username, string password)
    - Входни данни: username, e-mail, password
    - Изходни данни/резултат: Потребителя получава достъп до системата, а информацията се запазва в базата данни
  - void Login(string username, string password)
    - Входни данни: username,password
    - Изходни данни/резултат: Прави се проверка на данните, и потребителя получава достъп до системата
  - custom_ struct ViewProfile(string username)
    - Входни данни: Идентификация на потребител
    - Изходни данни/резултат: Налична информация за дадения потребител
  - void ChangeProfile(username,custom_attribute) – множество възможности
    - Входни данни: Идентификация на потребител, нови въведени данни
    - Изходни данни/резултат: Обновяват се данните в базата данни
  - void GetInfo(username,custom_attribute) – множество възможности

        - Входни данни: username на потребител
        - Изходни данни/резултат: Получава се информация за правата на дадения потребител
    - bool validateUsername(string username)
        - Входни данни: username на потребител
        - Изходни данни: 1 при успешно валидиране, 0 при неуспешно
    - bool validatePassword(string password)
        - Входни данни: password на потребител
        - Изходни данни: 1 при успешно валидиране, 0 при неуспешно

Drones Managament

- Предназначение:

  Модулът отговаря за управлението на дроновете и тяхната работа (маршрут, брой, алгоритми за разпознаване на свободни места)

- Основни отговорности на модула в системата:

  Основните отговорности са свързани с определянето на броя и разположението на дроновете, определяне на маршрутите, както и за алогоритми за обработване на информацията, която се извлича от тях. Чрез съобщения изпращани от самите дронове се получава информация за състоянието на всеки дрон.
- Описание на интерфейсите на модула:
    - Point3D getLocation(int droneNumber)
        - Входни данни: номер на дрон
        - Изходни данни: структура от тип Point, съдържаща координатите на дрона

    - bool getState(int droneNumber)
        - Входни данни: номер на дрон
        - Изходни данни: 1 при нормална работа, 0 при проблем
    - int getNumberOfDrones()
        - Входни данни: username на потребител
        - Изходни данни: 1 при успешно валидиране, 0 при неуспешно

- void changeDroneMarchroute(Point3D* points,int droneNumber)
  - Входни данни: набор от точки(маршрут) и номер на дрон
  - Изходни данни/резултат: промяна на маршрут

Drones

- Предназначение:

  Модулът отговаря за системата на дроновете и за това да подава сигнал за състоянието си

- Основни отговорности на модула в системата:

  Основните отговорности са свързани с наличието на GPS и обработването на данни получени от него, наличието на камера и съхранението на информацията, която се получава от нея. Важно е да се отбележи и подмодулът, отговарящ за периодичното изпращане на информация за състоянитето на всеки дрон, за да може да се реагира навреме от техническия екип

- Описание на интерфейсите на модула
  - sendLocation()
    - Входни данни: няма входни данни
    - Изходни данни: структура съдържаща информация за координатите на дрона
  - sendInfo
    - Входни данни: няма входни данни
    - Изходни данни: структура съдържаща информация за дрона
  - sendPhoto
    - Входни данни: няма входни данни
    - Изходни данни: снимка от камерата на съответния дрон

API Module

- Предназначение:

  Модулът отговаря за комуникацията на системата с всички външни системи, нужни за функционирането ѝ.

- Отговорности:

  Модулът е разделен на подмодули спрямо различните видове външни системи. Отговаря за наличието на информация относно атмосферните условия, връзка с външни системи за плащане, системи, наличиетно на GPS карти, както и връзка с базата данни. Всяка операция в системата изисква използването на базата данни, а за функционирането на системата е

невъзможно без външните системи. Затова и всички API са обособени в отделен модул.

Mobile App

- Предназначение:

  Модулът служи за използването на системата от потребител, като той има възможност за промяна на профила си, търсене на свободни паркоместа и съответното им заплащане.

- Отговорности:

  Модулът трябва да обединява отделните модули, като използва функционалностите им, за създаването на цялостна система

- Описание на интерфейсите на модула:
  -

Описание на допълнителните структури

Структура на внедряването

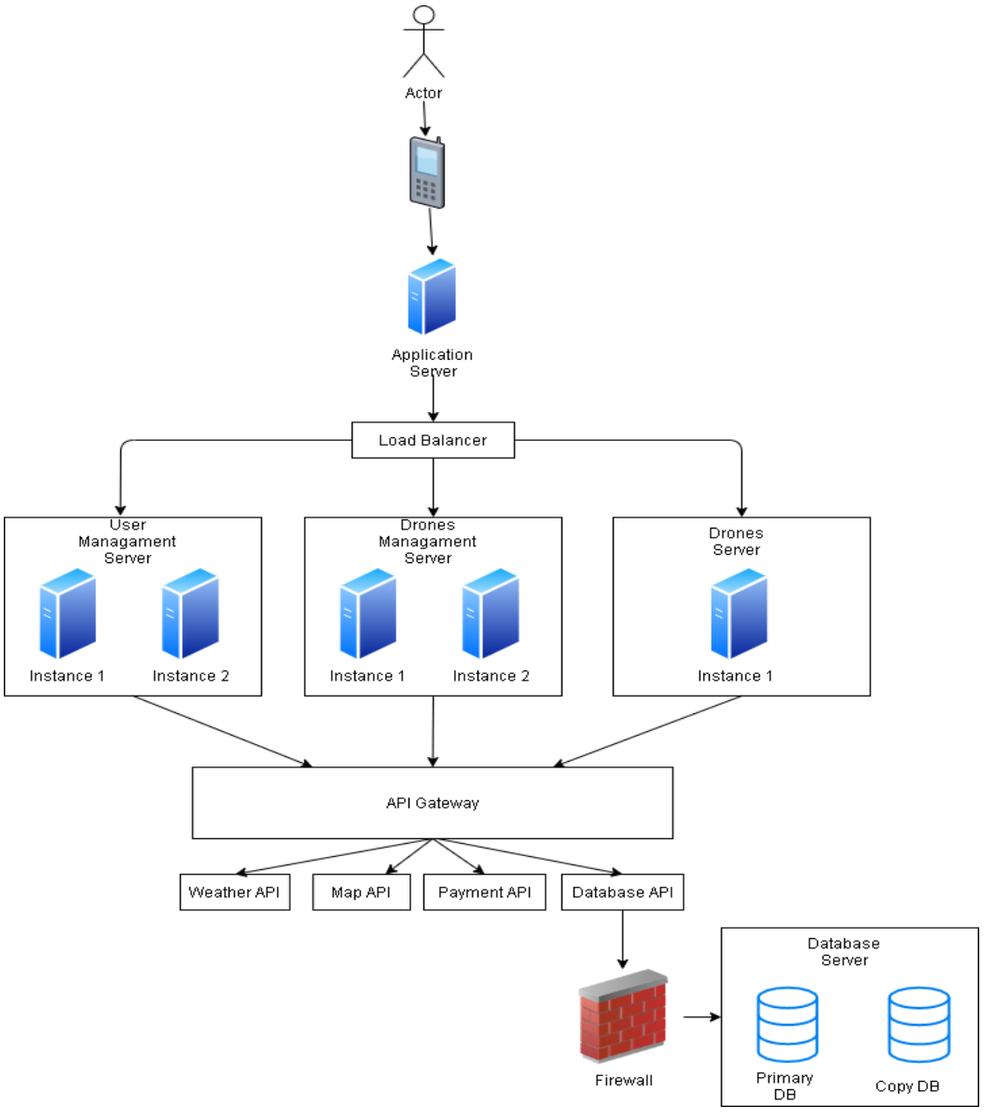
Структурата на внедряването е нужна за описването на решения, свързани с нефункционалните изисквания, тъй като тези решения няма как да бъдат описани в модулната структура. Всеки модул е разположен на отделен сървър, като по-важните имат по две инстанции с цел предодвратяването на претоварване или при евентуален отказ на даден сървър да заработи втората инстанция.

Изискването *„**Останалите потребители трябва да имат 100% защитен от външна намеса достъп до системата“*** изисква наличието на мерки взети за защита

на базата данни. Тя се намира на отделен сървър като до него има достъп само Database API. Включена е употребата на firewall, който спомага за защита от външен достъп. Има две инстанции на базата данни, в случай че някаква информация бъде погрешно сменена или изгубена поради технически причини.

Друго изискване, пораждащо нуждата от структура на внедряването е *„**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)“.*** Както вече споменахме, наличието на две (или повече инстанции) на даден сървър ще спомогне за удовлетворяването на изискването. При евентуален проблем, всеки сървър ще може да бъде заменен и системата ще продължи да работи.

Първично представяне



Описание на елементите и връзките

- **Application Server** предоставя интерфейс за работа с приложението
- **Load Balancer** разпределя заявки към съответния сървър, като спомага да се избегне претоварване и да се намали времето за отговор за заявка
- **User Managament Server** отговаря за обработката на потребителските профили
- **Drones Managament Server** отговаря за обработката на информация свързана с Drones Managament модулът
- **Drones Server** е свързан с вградената системита на дроновете
- **API Gateway** отговаря за връзката между сървърите и съответните API, които са нужни за функционирането на системата.
- **Weather API** отговаря за връзката с външна система, предоставяща информация относно атмосферните условия
- **Map API** отговаря за връзката с различни карти, с които работи системата
- **Payment API** отговаря за връзката с външни системи, с които се извършва заплащане
- **Database API** отговаря за връзката с базата данни
- **Firewall** има за цел да защити от външен достъп базата данни
- **Database Server** съдържа цялата информация за потребителите и данните за работата на системата

Структура на употребите

Структурата служи за по-задълбочено разбиране на работата на системата. Показва основната обмяна на информация между различните подмодули на главните модули, които са отразени в модулната декомпозиция. Може да служи като спомагателно средство за последователността при разработката или за улеснено добавяне на нова функционалност.

За да се избегне пренасищане със стрелки върху модулната диаграма, сме избрали да представим връзките между модулите по-малко по различен начин.

Първично представяне

- **Mobile App** извършва комуникацията с отделните модули чрез **Communication API**
- **State** използва информацията от **Condition**

- **Location** използва информацията от **GPS**
- **Free Spots Algorithm** използва информацията от **Camera**
- **Number and Marchroute of Drones** използва информация от **Weather API**
- Всички подмодули на **User Manager** изпращат или вземат информация от **Database API**
- **GPS и Location** използват ресурси, предоставени от **Map API**

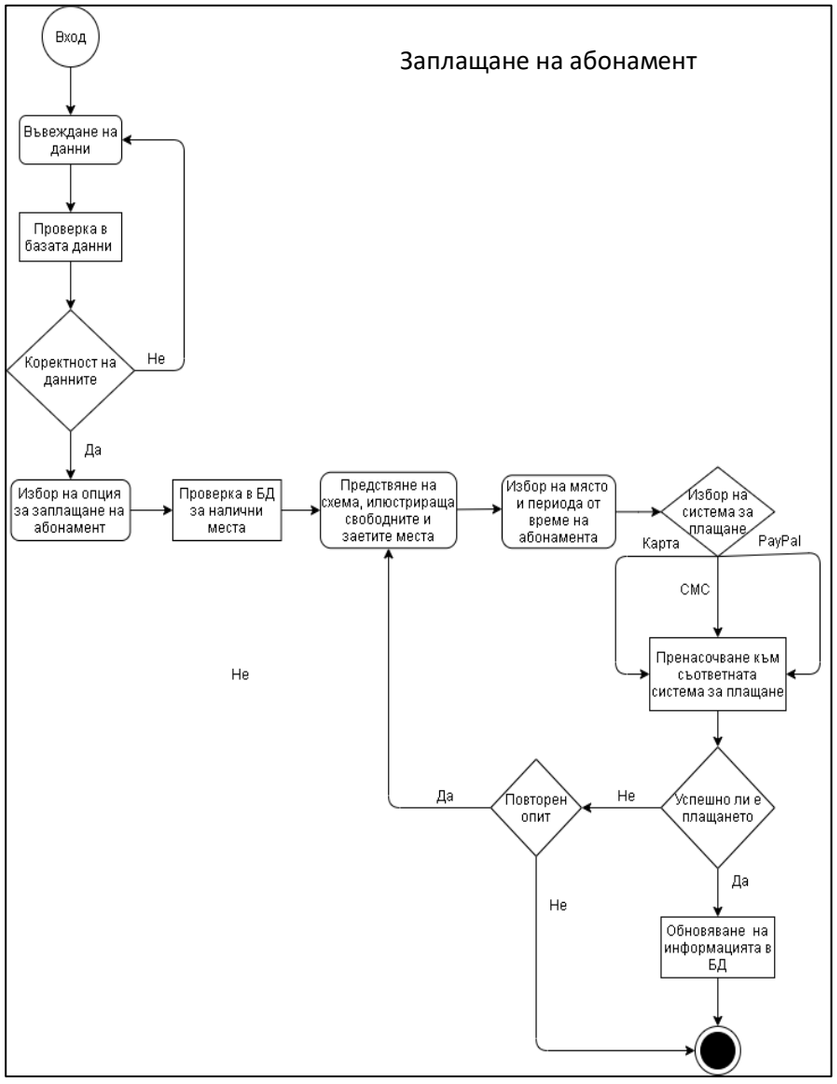
Описание на елементите и връзките

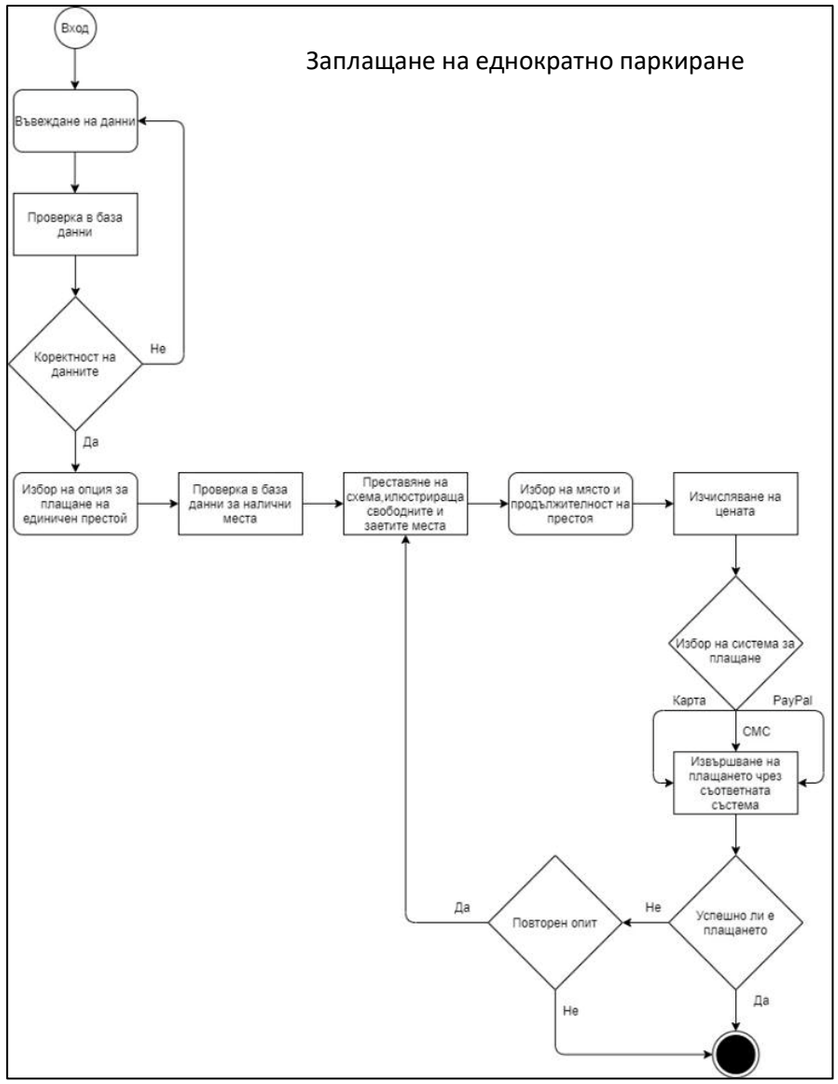
Отделните елементи са модулите на системата, които изграждат цялата система, както е показано в модулната декомпозиция. Връзките представят нуждата от обмяна на информация между отделни подмодули, като са съобразени нормите за modifiability и информацията се получава и изпраща с помощта на **Communication API**. За да работи правилно дадена функционалност, изградена с помощна на два или повече модула с изградена връзка помежду си е нужно всички те да функционират правилно.

Структура на процесите

Структурата представя основните процеси, случващи се при взаимодействието на системата с потребителя (заплащане на абонамент, еднократно ползване на паркомясто, изпращане на сигнал за нарушител). Основната цел е част от заинтересованите лица (потребители, инвеститори, маркетинг екипи, както и всички без дълбоки познания относно проекта) да придобият по-ясна представа за същността на системата. В диаграмите не са използвани сложни термини или нотации, които изискват специални знания за разчитането им. Представено е и взаимодейстие на системата с базата данни на абстрактно ниво. Изобразяването на процесите може да бъде и полезно за разработващия екип, като им помогне да се предвидят евентуални места, където могат да възникват трудности по време на създаването на системата или места където могат да се получат грешки при неочаквани действия от страна на потребителите.

Първично представяне





Описание на елементите и връзките

За представянето на елементите и връзките са използвани UML нотации, като трябва да отбележим, че когато е използван правоъгълник със заоблени върхове става въпрос за събитие извършено от потребителя, а при използването на нормален правоъгълник – събитие случващо се в системата.

Връзките представляват последователността от събития случващи се при използването на системата.

Архитектурна обосновка

Най-общо за да се удовлетворят функционалните изисквания, в модулната декомпозиция са обособени отделни модули, отговарящи за тези изисквания, а за да се удовлетворят качествените изисквания, са взети решения, описани в структурата на внедряването. По-подробно ще разгледаме изискванията,пораждащи нужда от вземане на решения относно архитектурата със съответните примери:

- **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.**

  Изискването е удовлетворено, чрез обособяването на отделни модули, описани в модулната декомпозиция (**Number and Marchroute of Drones, Weather API)**. За да съхраним информацията, която ни е нужна, за определяне на работата на дроновете имаме **Database API**.

- **За определяне на метеорологичните условия да се ползва външна услуга за прогноза за времето**

  Информацията за метеорологичните условия се приема от външен източник с помощта на **Weather API.**

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

  Обособен е подмодул **Free Spots Algorithms**, отговарящ за обработването на данните, получени от дроновете и предаване на информацията, нужна на потребителя.

- **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

Модулът **User Managament** има за цел да разграничи различните видове потребители, като главно това се изразява в правомощията, които имат и функционалностите, които системата предлага. За това спомога и **Access Control Manager**.

- **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.**

  Всеки дрон изпраща информация за състоянието си, на определен период от време (**State** модулът). Така при евентуален проблем веднага ще може да се реагира.

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

  Информацията се получава от външна система с помощта на **Weather API.**

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

  Плащането става посредством **Payment API,** осигуряващо връзка с външна система.

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

В структурата на внедряването са описани тактиките за съхранение на чувствителна информация, а именно използването на **Firewall**, достъп до базата данни единствено и само през **Database API**. Криптирането на информацията също е наложително. **Access control Manager** също служи ограничаване на достъпа до различни данни.

- **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).**

  Отново в структурата на внедряването са описани тактиките за удовлетворяването на изискването. Използването на повече от две или повече инстанции на даден сървър, отговарящ за някоя функционалност, спомага за постигането на по-голяма наличност на системата. **Load Balancer** има за задача, освен да разпредели заявките равномерно, да установи проблем с даден сървър и да спре да изпраща заявки към него.

  В контекстната диаграма се вижда и използването на повече от една външна система за определена работа, тъй като без тях нашата система няма как да функционира правилно, и при евентуален срив в някоя от тях ще имаме възможност да използваме подобна такава.