Софийски университет

„Свети Климент Охридски“

Факултет по математика и информатика

Специалност: “Софтуерно инженерство ”

Курс: 2, Група: 1

Дисциплина: “Софтуерни архитектури и разработка на софтуер”

КУРСОВ ПРОЕКТ

Изготвил:

Кристина Йотова 62247

Александър Найденов 62251

Летен семестър 2019/2020

Съдържание

1. Въведение

1.1. Обща информация за текущия документ

1.1.1. Предназначение на документа

Целта на документа е да запознае заинтересованите лица с архитектурата на софтуерната система за следене и управление на свободните паркоместа. В него са представени основните архитектурни решения, взети относно разработката на системата.

1.1.2. Описание на използваните структури на архитектурата

- **Декомпозиция на модулите**
  Декомпозицията на модули е много важна част от разработката на всяка софтуерна система. Чрез нея се придобива идея за реализацията на самата система. Използвайки декомпозиция на модулите можем по-лесно да:
  - разпределяме работата на разработчиците, защото всеки ще знае точно кой модул да разработи без да се интересува от другите
  - променяме модулите, защото знаем, че при промяната няма да счупим някой друг и да нанесем вреди на цялата система, които ще изискват много време за поправяне
  - разберем функциите на отделните модули

- **Употреба на модулите**
  За пълнота на описанието на модулите сме избрали да представим и модулната структура употреба на модулите, тъй като смятаме, че декомпозицията не предоставя информация как модулите си взаимодействат помежду си по време на изпълнение. Благодарение на структурата употреба на модули Всички заинтересовани лица могат да видят приложението на различните модули: как си взаимодействат помежду си, в кои процеси участват и колко са важни за системата.

- **Структура на внедряването**
  Структурата на внедряването показва как ще бъдат разпределени различните модули върху хардуерни/софтуерни системи, необходими за работата на приложението DronesPark. Чрез тази структура желаем да покажем специфичните архитектурни решения, които сме взели - използването на два сървъра UI и DronesPark, вместо един и комуникацията между тях. Тъй като искаме да имаме доста добра защита срещу атаки сме използвали

firewall, което сме изобразили и на нашата диаграма. Клиентът използва приложението с помощта на мобилно приложение или уеб браузър като директо си комуникира с UI server-а, който пък от своя страна си комуникира с DronesPark сървъра с помощта на HTTP протоколи. Изготвянето на тази структура е част и от определянето на какви и колко големи хардуерни ресурси трябва да бъдат на разположение за реализирането на системата. Тези ресурси за ключови за определянето на бюджета за разработка на приложението.

- **Структура на процесите**
  Чрез структурата на процесите сме се опитали да изобразим основните функционалности на DronesPark. Използвали сме 3 диаграми, в които сме изобразили различни функционалности на системата. Важно е да се отбележи, че тази структура представя по-високо ниво на абстракция с цел по-лесно разбиране на функционалностите.Структурата на процесите е насочена главно към заинтересованите лица без технически знания като потребителите. Нейната цел е да онагледи последователността на изпълнение на процесите.

1.1.3. Структура на документа

Документът е разделен на 4 секции:

- Секция 1 “Въведение” (точка 1)
  Дава описание на предназначението на документа, неговата структура, както и използваните структури и термини при изграждането на софтуерната архитектура на системата.
- Секция 2 “Архитектурни структури” (точка 2,3,4,5)
  За всяка архитектурна структура се описва защо е избрана, какви елементи включва и какви са връзките между тях.
- Секция 3 “Архитектурна обосновка” (точка 6)
  В тази секция е описано по какъв начин е изпълнено всяко едно от изискванията към системата.
- Секция 4 “Допълнителна информация” (точка 7)
  В тази точка е описана допълнителната информация относно системата, представена по разбираем начин на читателя.

1.2. Общи сведения за системата

В последните години намирането на свободни места за паркиране в големите градове се превръща в голямо предизвикателство. Системата DronesPark служи за следене и управление на свободните паркоместа в градовете чрез система от дронове. Тя е предназначена за хора, които искат бързо и лесно да

наемат свободно паркомясто. Функционалностите заложени в нея са свързани с това да може да се наема или абонира удобно за потребителя място за паркиране, заплащайки услугата чрез СМС, карта или PayPal. Системата е доста полезна за потребителите, които често паркират на определено място, защото чрез нея могат да направят абонамент за това място с цел да не бъдат неприятно изненадани когато “любимото” им място е заето.

От огромно значение за системата са именно дроновете те се следят дали са в пълна изправност и при възникнал проблем се изпраща сигнал за да може аварийният екип да успее максимално бързо да отстрани повредата.

За да се подсигури сигурността на данните на всеки потребител, се прилага криптиране на данните и при вход в системата се премива през автентикация и оторизация. Освен това, като допълнително ниво на сигурност се преминава и през защитни стени (firewall). Отказоустойчивостта на софтуерния продукт се поддържа чрез допълнителни сървъри(идентични по функционалност, но различни по начин на реализация), които да заменят главните ако има повреда в тях.

Архитектурните драйвери на системата са следните изисквания:

- ✓ Идентифициране и заснемане на свободните паркоместа от системата от дронове
- ✓ Използване на специфичен алгоритъм за разпознаване на свободните места, на база на заснетите изображения
- ✓ Системата поддържа 6 групи потребители
- ✓ При трайно намалена видимост да се извести оператора
- ✓ Регистрираните потребители могат да заплащат абонамент за определено паркомясто
- ✓ Динамично таксуване на паркоместата
- ✓ Различни методи за плащане
- ✓ Групите по контрол на паркирането следят дали няма. 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.
- ✓ Системата да работи 100% без отказ в рамките на светлата част на работния ден

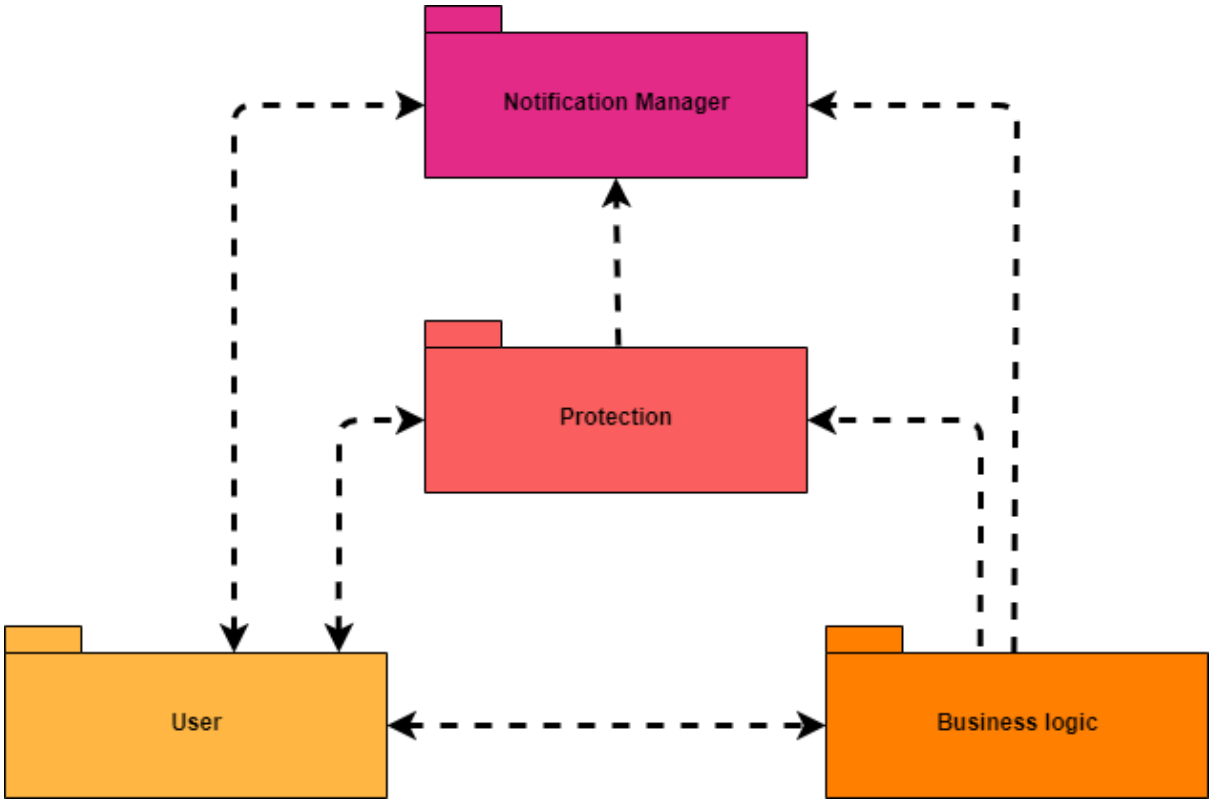
1.3. Терминологичен речник

- **Автентификация** - или удостоверяване на автентичност означава електронен процес, който позволява електронната идентификация на физическо или юридическо лице или потвърждаването на произхода и целостта на данни в електронна форма

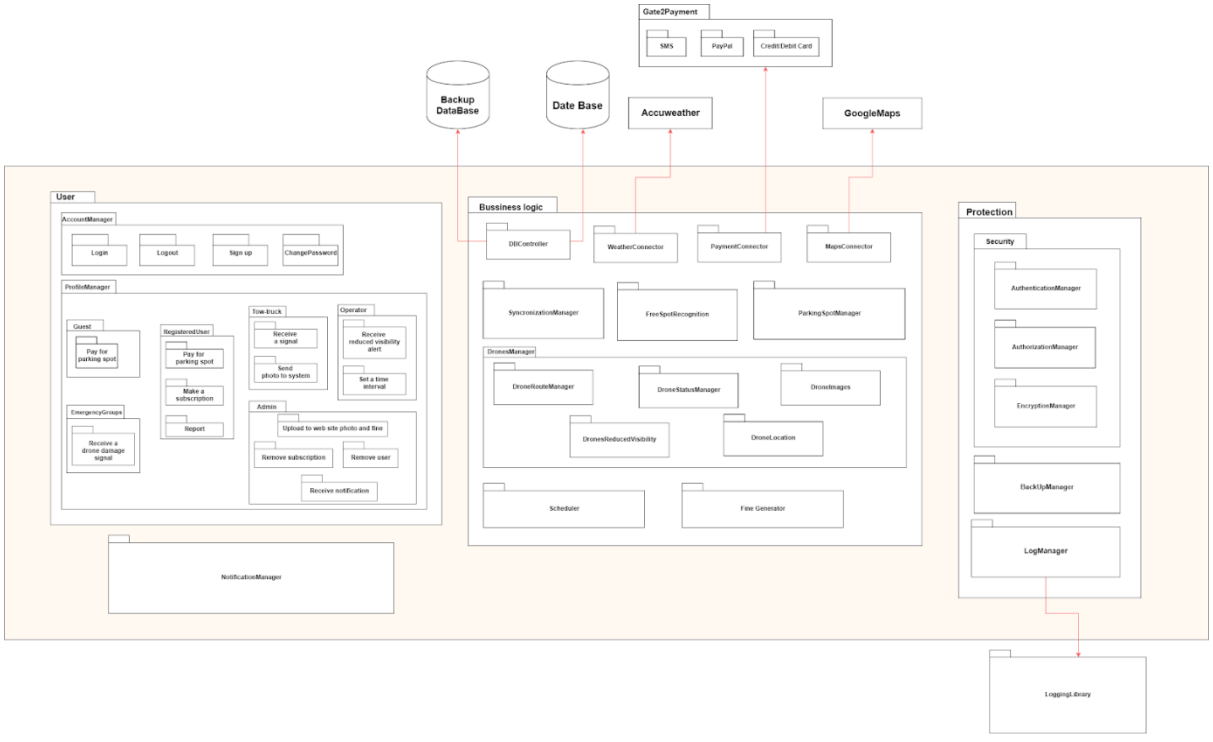
- **База от данни**- представлява колекция от логически свързани данни, структурирани по определен начин, така че компютърна програма да може да извлича информация по зададени критерии.
- **Външна система** – отдалечена софтуерна система, която е източник на данни или функционалности, които биват използвани от основната система.
- **Декомпозиция на модулите**- софтуерна структура, показваща как системата се разделя на отделни модули. Типовете елементи изграждащи тази структура са модули, а връзките между тях са от типа “X е подмодул на У”.
- **Заинтересовани лица**- всички, които имат отношение към създаването на софтуерната система – напр. собствениците, управителите, специалистите по продажби, ръководителя на проекта, разработчиците, екипа по поддръжка, различни прослойки от страна на клиента, крайните потребители и т.н.
- **Защитна стена** - или така наречената Firewall е е специализиран хардуер или софтуер, който проверява мрежовия трафик, преминаващ през него и разрешава или забранява достъпа по определени правила.
- **Криптиране**- процесът на кодиране на информацията по начин, който предотвратява достъпа на неоторизирани лица до нея.
- **Модул**- подсистема, част от компютърна програма, която изпълнява конкретни действия.
- **Оторизация**- функция за определяне на права / права на достъп до ресурси, която е свързана с сигурността на информацията и компютърната сигурност като цяло и по-специално с контрола на достъпа.
- **Софтуер** - е информация, обработена от компютърни системи, програми и данни. Той включва компютърни програми, библиотеки и свързани неизпълними данни, като онлайн документация или цифрови носители.
- **Софтуерна система** - система, която е в състояние да вземат набор от входни данни, да ги обработи и да се създадат набор от изходни данни.
- **Структура на внедряването**– Показва как софтуера се разполага върху хардуера и комуникационното оборудване. Елементите са процеси, хардуерни устройства и комуникационни канали. Връзките са напр. “внедрен върху” или “мигрира върху”
- **Структура на процесите**- дава информация за това каква е последователността на изпълнение, кои процеси могат да протичат паралелно,какво действие може да стартира или прекрати един процес, както и какви са възможните резултати от изпълнението му.
- **Употреба на модулите**- показва какви са връзките между отделните модули. Връзките между модулите са от вида “X използва Y”.

2. Декомпозиция на модулите

2.1. Общ вид на декомпозицията на модули за системата



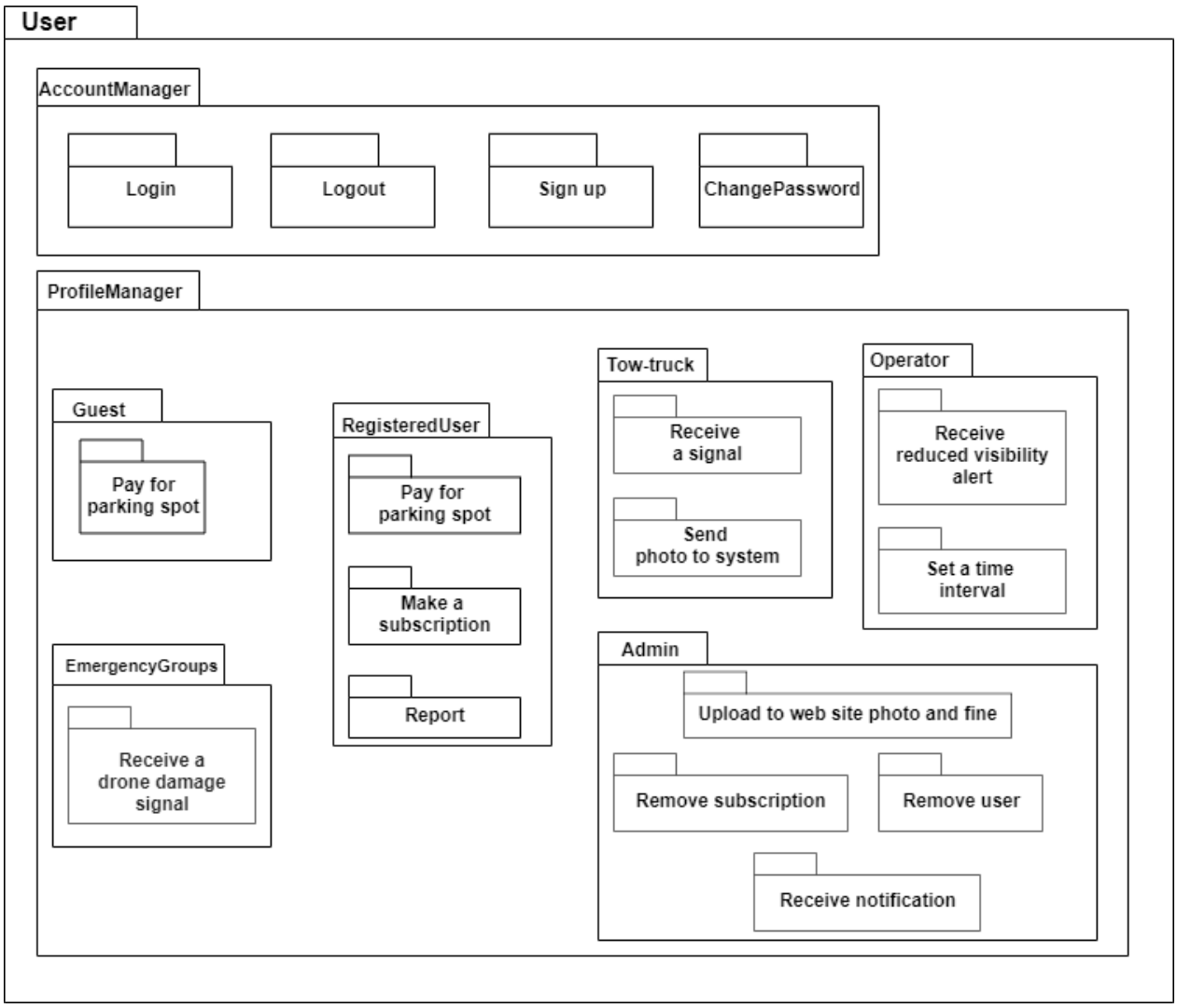
2.2.Контекстна диаграма



2.3. Подробно описание на всеки модул

Декомпозицията на модулите се състои от 4 главни модула:

- **User**



Модулът има за цел да контролира достъпа до различни функционалности на различните видове потребители на системата (Администратор, Оператор, Аварийни групи, Групи по контрол на паркирането (т.нар. „паяци“), Регистрирани потребители и Обикновени потребители). Този модул се състои от два подмодула:

- **AccountManager-** Модулът съдържа всички общи функционалности за отделните потребители.
  - **Login-**Този модул съдържа функционалността за вход на user-ите, които вече имат създаден профил, който се пази в базата данни.

Входни данни: потребителско име и парола

Изходни данни: дали логването е успешно

Съобщения за грешка: При неправилно въвеждане на потребителско име и/или парола да се оповестява NotificationManager.

Връзка с други компоненти: При влизане в системата модулът Login изпраща паролата към EncryptionManager и след това се изпраща заявка към AutenticationManager за верифициране на въведените данни.

- **Logout-** Този модул съдържа функционалността за изход от системата, когато потребителят реши, че не желае да я използва.

  Съобщения за грешка: При възникнал проблем относно излизането от системата да се оповестява Notification Manager.

- **Sign up-** Този модул съдържа функционалността за регистрация в системата на нови потребители.

  Входни данни: потребителско име, имейл адрес, парола

  Изходни данни: дали е успешна регистрацията

  Съобщения за грешки: При вече съществуващо потребителско име или имейл в базата данни или при неуспешно създаване на акаунт се оповестява NotificationManager.

  Връзка с други компоненти: При регистриране модулът Sign up изпраща заявка към EncryptionManager за криптиране на паролата и към DBController за верифициране на данните в БД.

- **ChangePassword-** Модулът изпълнява функционалността за смяна на парола.

  Входни данни: стара парола, нова парола

  Изходни данни: дали е успешна смяната на паролата

  Съобщения за грешка: При неправилно въвеждане на старата парола или неуспешна промяна на паролата се оповестява NotificationManager.

  Връзка с други компоненти: При опит за смяна на паролата модулът ChangePassword се свързва с EncryptionManager и също така изпраща

заявка към DBManager за проверка за коректността на въведената стара парола и записа на новата.

- **ProfileManager-** Модулът става активен след успешен вход в системата. Той служи за разграничаване на различните потребители на системата, тъй като те имат достъп до различни функционалности.
  - **Guest**
    - **Pay for parking spot-** Този модул отговаря за единствената функционалност, която системата предоставя на нерегистрирания потребител, а именно наемането на паркомясто.

      Входни данни: номер на паркомясто[1] (описано е в точка 7)

      Изходни данни: успешно ли е запазено

      Съобщения за грешка: При неуспешно наемане на паркомясто се уведомява NotificationManager.

      Връзка с други компоненти:При намиране на място модулът се свърза с ParkingSpotManager, откъдето се наема паркомястото и също така с PaymentConnector, чрез който се осъществява плащането.

  - **Emergency groups**
    - **Receive a drone damage signal-** Този модул отговаря за единствената функционалност на аварийните групи- да получават сигнали за повредени дронове.

      Входни данни: сигнал

      Изходни данни: успешно ли е приет сигналът

      Връзка с други компоненти: Получава постъпили сигнали от DroneStatusManager чрез NotificationManager.

  - **Registered user**
    - **Pay for parking spot-** Този модул служи за наемане на паркомясто от регистриран потребител.

      Входни данни: номер на паркомясто[1] (описано е в точка 7)

      Изходни данни: успешно ли е нает паркомястото

Съобщения за грешка: : При неуспешно наемане на паркомясто се уведомява NotificationManager.

Връзка с други компоненти: При намиране на място модулът се свърза с ParkingSpotManager, откъдето се наема паркомястото и също така с PaymentConnector, чрез който се осъществява плащането.

- **Make a subscription-** Този модул служи за абониране за определено паркомясто.

  Входни данни: номер на паркомясто[1] (описано е в точка 7)

  Изходни данни: успешно ли е направен абонамент за паркомястото

  Съобщения за грешка: При неуспешно абониране за паркомястото се уведомява NotificationManager.

  Връзка с други компоненти: При намиране на място модулът се свърза с ParkingSpotManager, откъдето се наема паркомястото и също така с PaymentConnector, чрез който се осъществява плащането.

- **Report-** Модулът служи за осигуряване на функционалността за докладване на неправомерно заето място до групите по контроли докладване на администе за нередности относно сайта.

  Входни данни: потребителски сигнал

  Изходни данни: дали успешно е изпратен сигналът

  Съобщения за грешка: При неуспешно изпращане на сигнал към NotificationManager грешката се изпраща отново към NotificationManager.

  Връзка с други компоненти: Изпращане на сигналите става чрез NotificationManager, който от своя страна разпределя сигналите.

- **Tow-truck**

- **Receive a signal -** Модулът служи за получаване на сигнал от регистрираните потребители за неправомерно заето паркомясто паркиране, както и за сигнал за неплатено място, но заето.

  Входни данни: сигнал

  Изходни данни: дали сигналът е получен успешно

  Връзка с други компоненти: Сигналите се получават посредством NotificationManager- от регистриран потребител посредством модулът Report, а за неплатено място чрез модулът FreeSpotRecognition.

- **Send photo to system -** Модулът служи за качване на снимка на неправомерно заел място автомобил във система.

  Входни данни: снимка

  Изходни данни: дали успешно е качена снимката

  Съобщения за грешка: При неуспешно качване на снимката в базата данни се оповестява NotificationManager.

  Връзка с други компоненти: Качването на снимката в базата данни се осъществява чрез DBController.

- **Admin**
  - **Upload to website photo and fine -** Модулът служи за качване на снимка и фиш на публично достъпния уеб сайт.

    Входни данни: фиш и снимка

    Изходни данни: дали успешно са качени снимката и фишът в уеб сайтът.

    Съобщения за грешка: При неуспешно качване на снимката и/или фиша на сайта да се уведомява NotificationManager.

    Връзка с други компоненти: Чрез DBController снимката и фишът се извличат от Базата от данни.

- **Remove subscription-** Модулът служи за премахване на абонамент на потребител, който по някакъв начин е нарушил правилата.

  Входни данни: потребител

  Изходни данни: дали успешно е премахнат абонамента

  Съобщения за грешка: При възникване на грешка по време на премахването на абонамент да се оповестява NotificationManager.

  Връзка с други компоненти: Чрез DBController се променя информацията в БД.

- **Remove user-**Модулът служи за премахване на потребител, който по някакъв начин е нарушил правилата.

  Входни данни: потребител

  Изходни данни: дали успешно е премахнат акаунта на потребителя

  Съобщения за грешка: При възникване на грешка по време на премахването на акаунта на потребителя
  да се оповестява NotificationManager.

  Връзка с други компоненти: Чрез DBController се изтрива акаунта от БД.

- **Receive notification-** Модулът служи за получаване на сигнал от потребителите за нередности в сайта.

  Входни данни: сигнал

  Изходни данни: успешно ли е приет сигналът

  Връзка с други компоненти: Чрез NotificationManager се получава сигналът от потребителя.

▪ **Operator**

- **Receive reduced visibility alert-** Модулът служи за получаване на сигнал за намалена видимост.

Входни данни: сигнал за намалена видимост

Изходни данни: успешно ли е приет сигналът

Връзка с други компоненти: Сигналът, изпратен от DronesReducedVisibility се получава посредством NotificationManager.

- **Set a time interval-** Модулът осъществява функционалността свързана с задаване на интервал от време за обновяване на информацията за свободните паркоместа.

  Входни данни: интервал от време

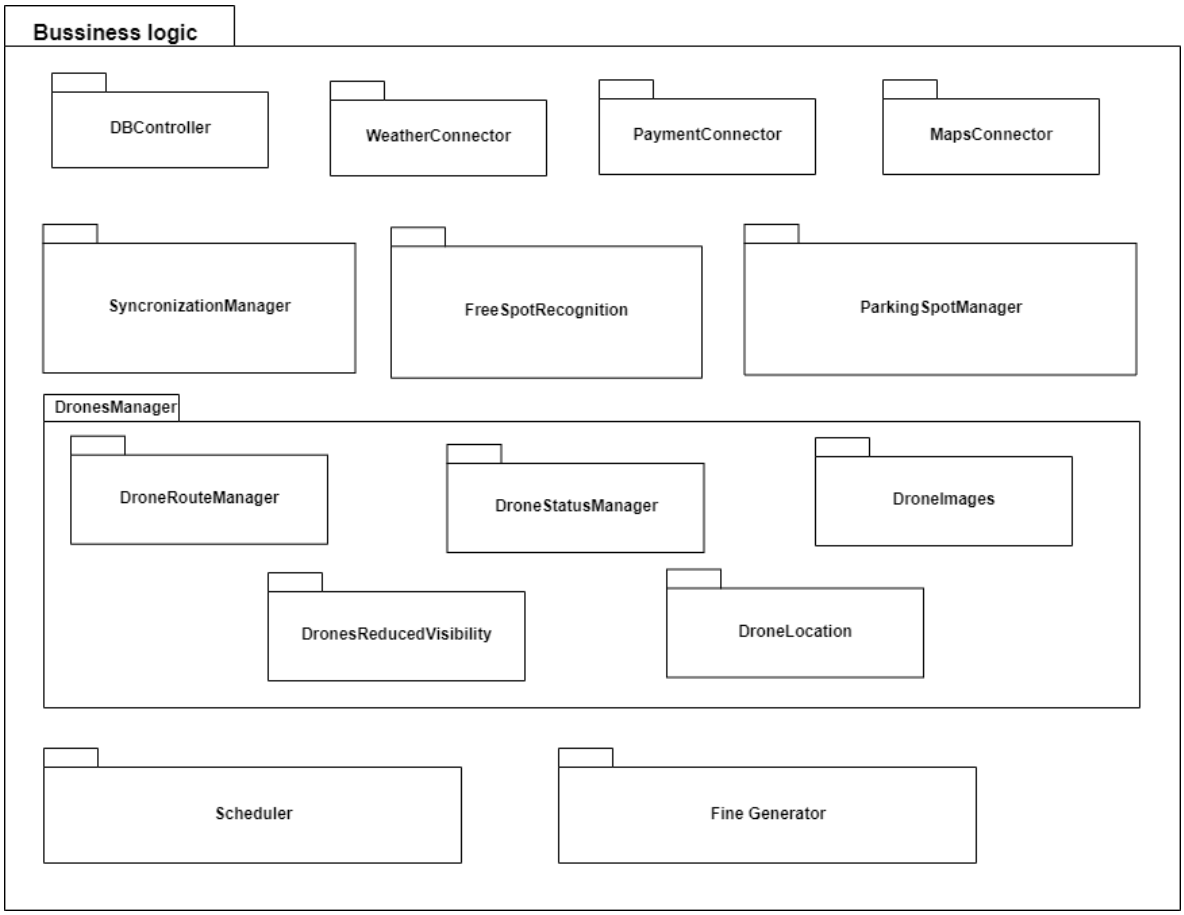
  Изходни данни: успешно ли е зададен интервалът

  Съобщения за грешка: При възникнала грешка със задаването на интервал се оповестява NotificationManager.

  Връзка с други компоненти: Интервалът от време се задава във DroneImages.

- **Business logic**

  В този модул се съдържа бизнес логиката на системата. Също така осъществява и връзката с външните системи.



- **DBController-** Модулът има за цел да предостави по-лесен достъп до базата данни Също така DBManager може да записва и променя информацията в базата данни. Също така той служи и за синхронизиране на двете бази данни.
  Входни данни: заявки

  Изходни данни: дали успешно е изпълнена заявката

  Съобщения за грешка: При липса на връзка с базата данни или неуспешна заявка ще изпраща подходящо съобщение на модулът NotificationManager.

  Връзка с други модули: Получаване на заявки от останалите модули (описани са в описанието на съответните модули).

  Връзка с външни системи: Черпи и записва информация от Основната и резервната база от данни.

- **WeatherConnector-** Модулът служи за връзка между системата и външната система Accuweather.

  Входни данни : район

Изходни данни : метрологично състояние на района

Съобщения за грешка: При невъзможно свързване с външната система Accuweather се оповестява NotificationManager.

Връзка с други модули: Изпраща информация за времето на ParkingSpotManager(описано е в описанието на съответния модул).

Връзка с външни системи: Извлича информация за метрологичното състояние от Accuweather.

- **PaymentConnector-** Модулът служи за връзка между системата и външната система Gate2Payments,чрез която се осъществяват плащанията.

  Входни данни : потребител

  Изходни данни : дали е успешно осъществено плащането

  Съобщения за грешка: При невъзможно свързване с външната система Gate2Payments се оповестява NotificationManager.

  Връзка с други модули: Има връзка с всички модули, отговарящи за наемане на паркомясто (описано е в описанието на съответните модули).

  Връзка с външни системи: Той има способността да си комуникира с външната система, отговаряща за различните видове плащания Gate2Payments(PayPal, SMS, кредитна или дебитна карта).

- **MapsConnector -** Модулът служи за връзка между системата и външната система Google Maps.

  Входни данни : район

  Изходни данни : карта на района

  Съобщения за грешка: При невъзможно свързване с външната система Google Maps се оповестява NotificationManager.

  Връзка с други модули: Има връзка с DroneRouteManager (описано е в описанието на съответния модул).

  Връзка с външни системи: Черпи информация за картата на района от GoogleMaps.

- **SyncronizationManager-** Модулът се използва за осъществяване на синхронизацията на данните, които получава клиента с настоящата версия на базата данни.

Входни данни: заявка за получаване на информация

Изходни данни: съответната информация от базата данни

Съобщения за грешка: При неуспех с извличането на информация оповестява NotificationManager.

Връзка с други модули: SynchronizationManager черпи информация от Базата данни чрез DBController.

- **FreeSpotRecognition** - Модулът има за цел да разпознава свободните места на база на заснетите изображения и да изпраща сигнали за неправомерно заети места.

  Входни данни: изображения, заснети от дронове

  Изходни данни: дали дадено паркомясто е свободно

  Съобщения за грешка: При възникнал проблем оповестява NotificationManager.

  Връзка с други модули: Използва заснетите и форманитарани изображения от модула DroneImages. Когато разпознае свободно място го изпраща до DBController, който от своя страна променя информацията в БД. Също така модулът изпраща сигнал посредством NotificationManager на Tow truck ако дадено паркомясто е заето, но не е платено в рамките на 5 минути.

- **ParkingSpotManager -** Чрез този модул се осъществява запазване на абонамент, както и еднократното наемане на паркомясто. Могат да се заемат само паркоместата, за които няма вече съществуващ абонамент. Също така този модул е отговорен чрез специален алгоритъм да определи динамично цената на абонамента.

  Входни данни: номер на паркомясто[1] (описано е в точка 7)

  Изходни данни: успешно ли е наемането

  Съобщения за грешка: При възникнал проблем оповестява Notification Manager

  Връзки с други модули: Чрез DBController-а се свързва с базата данни, откъдето черпи информация за паркоместата. От друга страна за определяне на цената използва информация от WeatherConnector.

Модулът получава заявки за запазване на абонамент, както и еднократното наемане на паркомясто от модулите Pay for Parking Spot съответно в Guest и Registered user, Make a subscription.

- **DronesManager-** В този модул се съдържат всички функционалности, свързани с управлението и извличане на информация от дроновете.
    - **DroneRouteManager-**Чрез специален алгоритъм изчислява и задава оптимален маршрут на дроновете спрямо картата на района, динамиката на паркиране в съответния ден и час от седмицата, както и спрямо метеорологичните условия.

      Входни данни : карта на района, динамика на паркиране, метеорологичните условия и местоположението на дроновете

      Изходни данни : задава оптимална траектория на дроновете

      Съобщения за грешка: При проблем в модула се изпраща подходящо съобщение към NotificationManager.

      Връзка с други модули : Получава данни за картата на района от MapConnector, метерологичните условия от WeatherConnector и местоположението на дроновете от DroneLocation.

    - **DroneStatusManager-**Анализира моментното състоянието на дроновете (в екплоатация, не е в експлоатация).

      Входни данни : номер на дрона² (описано е в точка 7)

      Изходни данни : статус

      Съобщения за грешка: При невъможно получаване на моментното състоянието на дрона, известява NotificationManager

      Връзка с други модул: Изпраща съобщение за повреда към NotificationManager, при извънредни стойности за състоянието на дроновете, който от своя страна има за цел да уведоми аварийните групи.

    - **DroneImages-**Модулът се използва за заснемане и обработване в удобен формат на снимките заснети от дроновете. Този модул служи и за задаване на интервал от време от оператора. 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. за заснемане на снимките на зоните за паркиране. Това време трябва да е съгласувано с изискването за “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.” като се има предвид, че FreeSpotRecognition може да разпознае свободните паркоместа за 5сек.

Входни данни : снимки от дроновете

Изходни данни : форматирана снимка

Съобщения за грешка: При възникнал проблем оповестява NotificationManager.

Връзка с други модули: Снимките се изпращат на FreeSpotRecognition в удобен за анализ формат.

- **DronesReducedVisibility -** Чрез специален алгоритъм върху заснетите изображения модулът служи за идентифициране на намалена видимост.

  Входни данни: заснети изображение

  Изходни данни: информация за видимостта (true/false)

  Съобщения за грешка: При възникнал проблем с получаване на изображения оповестява NotificationManager.

  Връзка с други модули: Използва заснетите снимки от DroneImages. При разпознаване на намалена видимост оповестява NotificationManager.

- **DroneLocation -** Показва точното местоположение на съответния дрон.

  Входни данни: номер на дрона² (описано е в точка 7)

  Изходни данни: локацията на дрона

  Съобщения за грешка: При проблем с определяне на местоположението, остава запазено последното коректно местоположение и оповестява NotificationManager.

  Връзката с други модули: За определяне на локация се използва MapsConnector. Ако няма промяна в местоположението на дрона в рамките на 30сек. променя статуса на дрона в DroneStatusManager на “не е в експлоатация”.

- **Scheduler-** Този модул служи за даване на по-висок приоритет на определени заявки (приоритетът се определя от важността на заявката за системата).

  Входни данни: неизпълнени заявки

  Изходни данни: приоритизирани заявки

  Съобщения за грешка: При възникнала грешка се оповестява NotificationManager.

- **Fine Generator-** Този модул служи за автоматично генериране на фиш за нарушение на правилата.

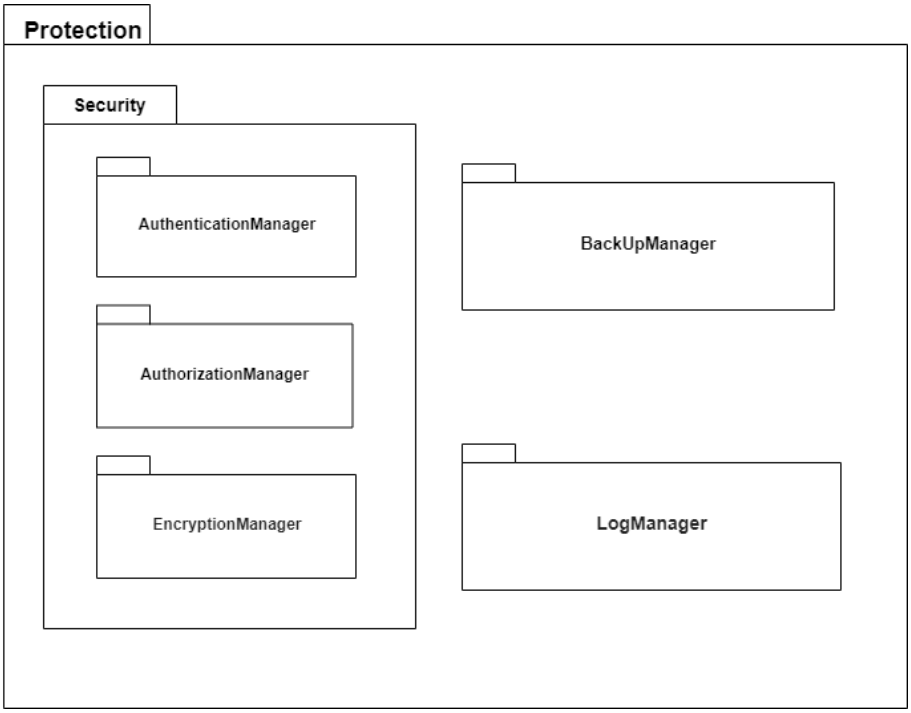
  Входни данни: данни за нарушителя и нарушението

  Изходни данни: фиш

  Съобщения за грешка: При възникнала грешка се оповестява NotificationManager.

  Връзка с други модули: Генерираният фиш се запазва в БД чрез DBController.

- **Protection**



Модулът се грижи за защита на данните в системата.

- **Security-**Този модул е свързан с постигането на висока защита на системата от атаки.
    - **AuthenticationManager-** Потребителят трябва да въведе потребителско име и парола (използва се SSL протокол). След като получи от потребителя въведеното потребителско име и парола, компютърът ги сравнява със стойностите, които се съхраняват в базата от данни и ако съвпадат, допуска потребителя в системата. При успешна автентикация ще се издава автентикационен номер.

      Входни данни: потребителско име и парола

      Изходни данни: автентикационен номер

      Съобщения за грешка:При неуспешно свързване с DBController се изпраща подходящо известие на NotificationManager.

      Връзка с други компоненти: При логване в системата модулът Login се свързва с AuthenticationManager, а той от своя страна с DBController.

    - **AuthorizationManager -** След успешна автентикация при всяка заявка от страна на потребителя ще се прави оторизация за правата, с които разполага, чрез дадения автентикационен номер. Така ще се знае с какви права разполага определения потребител.

      Входни данни: автентикационен номер

      Изходни данни: с какви права разполага съответния потребител

      Съобщения за грешка:При неуспешно свързване с AuthenticationManager се оповестява NotificationManager.

      Връзка с други компоненти: AuthotizationManager получава автентикационения номер от AuthenticationManager.

    - **EncryptionManager-**Модулът се използва криптиране на информацията- кодиране на информацията по начин, който предотвратява достъпа на неоторизирани лица до нея.

      Входни данни: информация

      Изходни данни: криптирана информация

      Съобщения за грешка: При неуспешно криптиране на информацията се изпраща подходящо съобщение към NotificationManager.

Връзка с други модули: Модулът приема информация от други модули, криптира я посредством специален алгоритъм и я изпраща към DBController, който от своя страна я запазва в Базата данни.

- **BackUpManager -** Този модул ще служи за синхронизиране на основната база данни с резервната база, така че всички данни в системата да бъдат защитени от загуба вследствие на срив или злонамерена промяна в базата от данни. Резервното копие ще се бъде създавано при всяка промяна в базата данни, тъй като не се очаква голям наплив в системата.

  Входни данни: нова информация в базата от данни

  Съобщения за грешка: При невъзможно създаване на резервното копие се оповестява NotificationManager.

  Връзка с други модули: Резервното копие се създава посредством DBController-а.

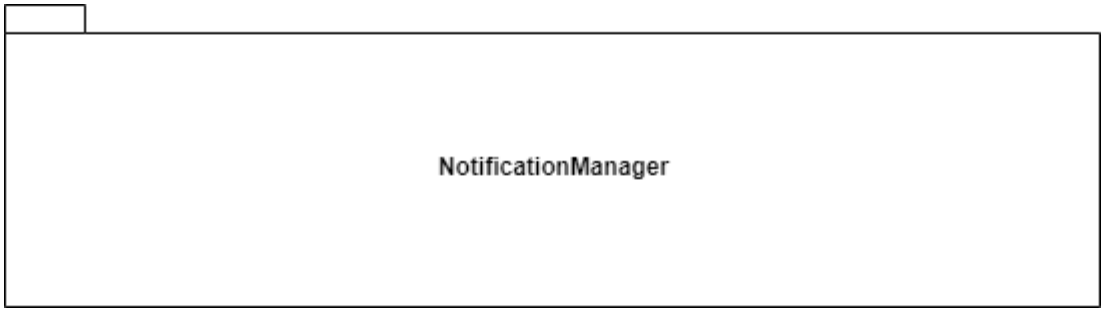
- **LogManager-** Модулът служи за записване (логване) на всички дейности в LoggingLibrary. Цялата информация по време на изпълнението се записва и това би подпомогнало до по-бързо възстановяване на системата ако възникне някаква грешка.

  Входни данни: информация за процесите протичащи в системата

  Съобщения за грешка: При невъзможно записване на лог се оповестява NotificationManager.

  Връзка с други модули: Логовете се записват в LoggingLibrary.

- **Notification Manager**

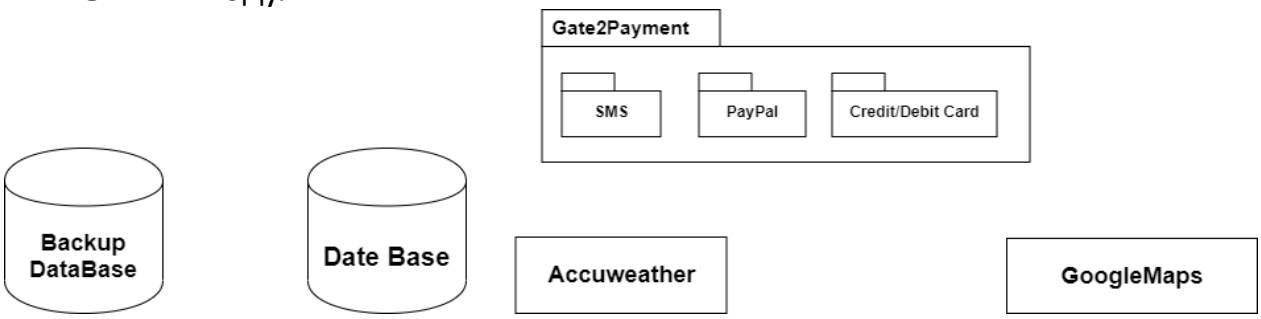


Модулът се занимава с разпределянето на всякакъв вид известия независимо дали те са грешки при свързването на отделни компоненти или нотификации към даден потребител на системата.

Входни данни: известия от други модули

Връзка с други модули: Този модул е от изключителна важност за системата и има връзки с всеки друг модул (връзките са описани при разглеждането на останалите модули).

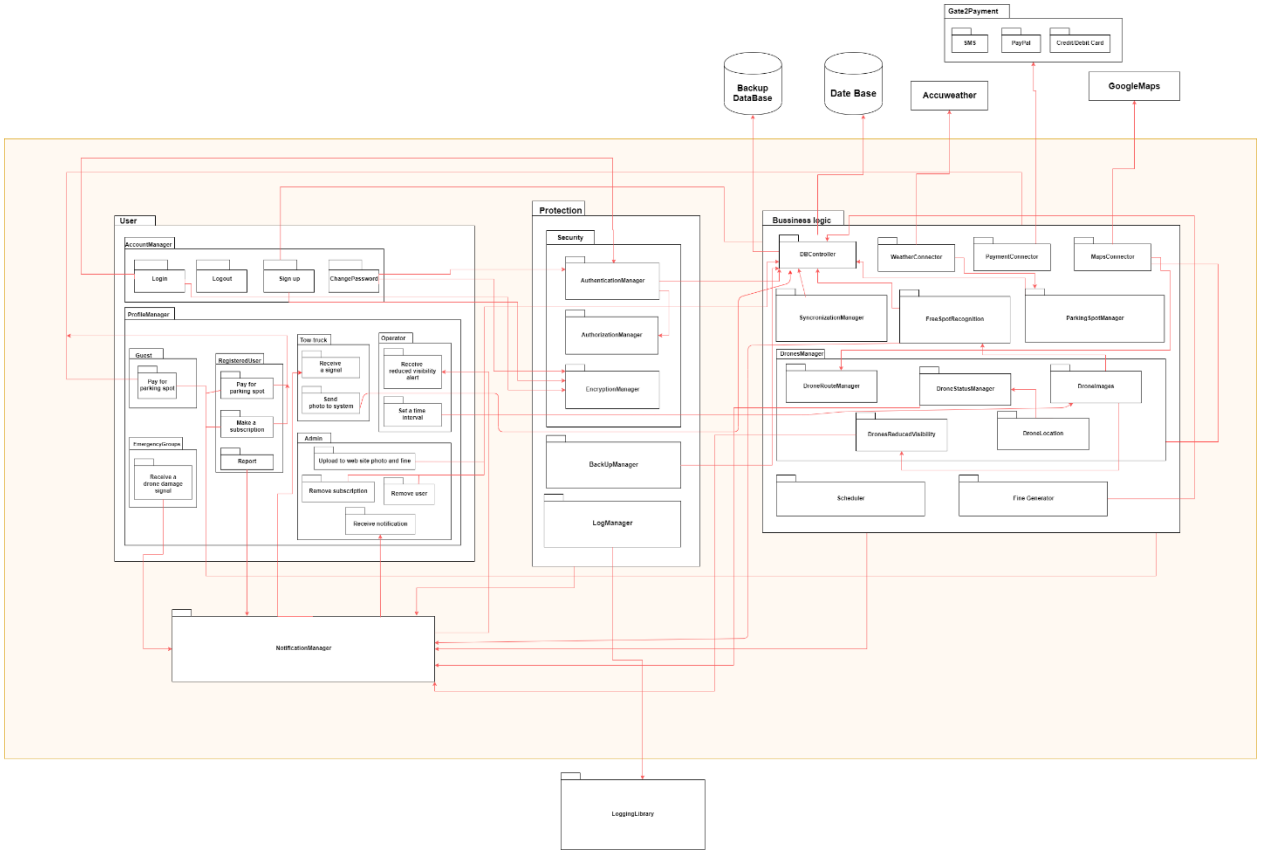
2.4.Външни модули



- **DataBase**
  Този модул представлява самата бази от данни, която се намира на отделен сървър, за по-голяма защита на данните. В нея се съхраняват както general information за потребителите, така и съдържа 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.
- **Backup Database**
  Този модул съдържа идентично копие на базата от данни, за да може при срив да я подмени без загуба на информация.
- **Accuweather**
  Този модул е външен за системата и служи за предоставяне на информация относно метеорологичните условия. Информацията става достъпна за системата чрез вътрешния модул WeatherConnector.
- **Gate2Payment**
  Този модул е външен за системата и служи за осъществяване на плащания от потребителите към системата. Плащането се осъществява чрез вътрешния модул PaymentConnector.
- **GoogleMaps**
  Този модул е външен за системата и служи за предоставяне на карти на районите, които системата обслужва. Информацията става достъпна за системата чрез вътрешния модул MapsConnector.
- **LoggingLibrary**
  Този модул е външен за системата и представлява система, в която се записват логовете.

3. Употреба на модулите

3.1. Първично представяне



Избрахме да използваме и диаграма за употреба на модулите, за да онагледим връзките, които сме описали в декомпозиция на модулите. Тя ясно представя взаимодействието между отделните модули като по този начин се изгражда по-добра цялостна представа за взаимодействието между отделните модули.

3.2. Описание на елементите и връзките

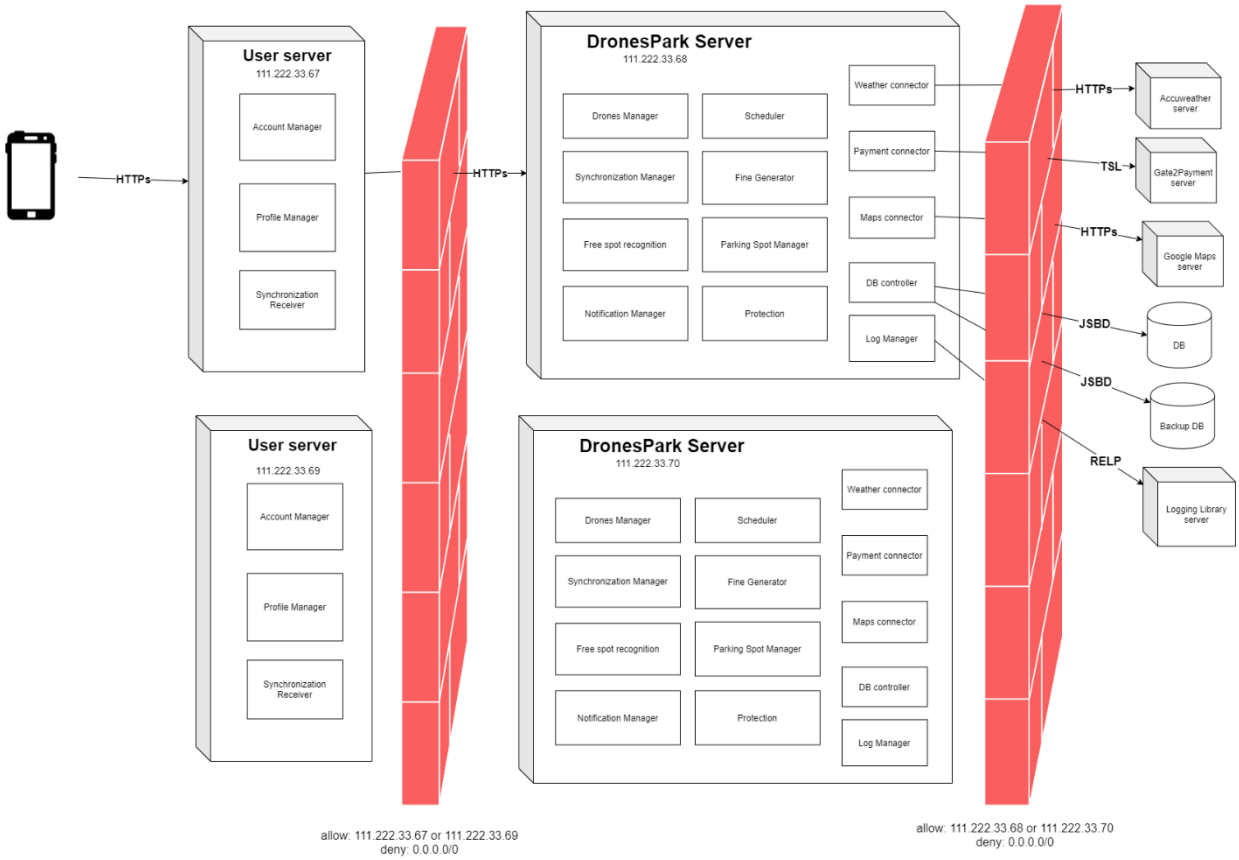
Структурата се състои от елементи (модули) и връзки между тях (стрелки, които указват посоката на взаимодействие). Елементите са модулите, които разглеждахме в декомпозицията на модулите.

3.3. Описание на обкръжението

DBController, WeatherConnector, PaymentConnector и MapsConnector, LogManager са модулите, чрез които системата си комуникира с външните системи.

4. Структура на внедряването

4.1. Първично представяне



4.2. Описание на елементите и връзките

Структурата на внедряването ясно показва специфичните архитектурни решения, които са използвани за реализацията на система. Всеки елемент на диаграмата е отделно хардуерно устройство. Тъй като не се очаква системата да бъде използвана от милиони потребители едновременно, тя няма да бъде подложена на огромни натоварвания и затова не е необходимо от разделяне на модулите на голям брой отделни сървъри. Избрали сме да имаме 2 основни сървъра:

- User server- на него са разположени модулите, отговарящи за потребителските функционалности.
- DronesPark server- на него са разположени модулите, отговарящи за бизнес логиката, защитата на данните и модулът за разпределяне на известията.

За да осигурим архитектурния драйвер “ Системата да работи 100% без отказ в рамките на светлата част на работния ден” сме използвали тактиката за отстраняване на откази чрез дублиране на сървърите на системата. Модулите на тези резервни сървъри са написани на различен програмен

език, използвани са различни алгоритми, дори и са писани от различен екип от програмисти. По този начин ако има срив в някой от основните сървъри, то “бъгът” няма да бъде дублиран и при резервните.

За допълнителна защита на данните са използвани защитни стени (firewall). Всички заявки постъпващи от User server преминават през firewall преди да достигнат основния сървър. По този начин възможността за сриване на системата от злонамерени атаки се свежда до минимум.

4.3. Описание на обкръжението

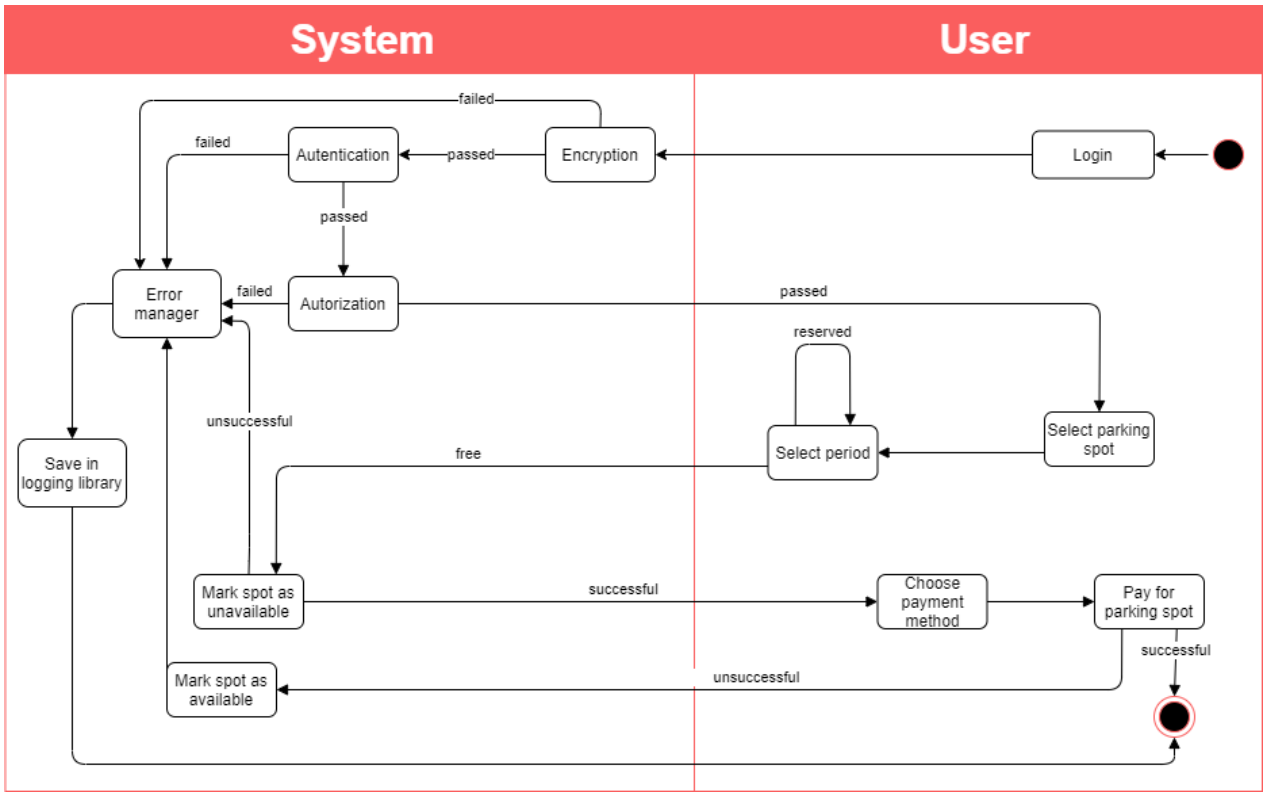
Външните модули за системата са разположени и на отделни сървъри, от които системата черпи информация. Тази информация също преминава през защитна стена, за да избегнем получаване на злонамерена информация с цел сриване на сървъра.

4.4. Описание за възможните вариации

Тъй като използваме универсални модули за комуникация с външните системи това прави много лесна подмяна на дадена външна система при необходимост. Ако системата се разрасне в бъдеще има вариант чрез дублирането на сървърите при пикови натоварвания двата сървъра да работят успоредно.

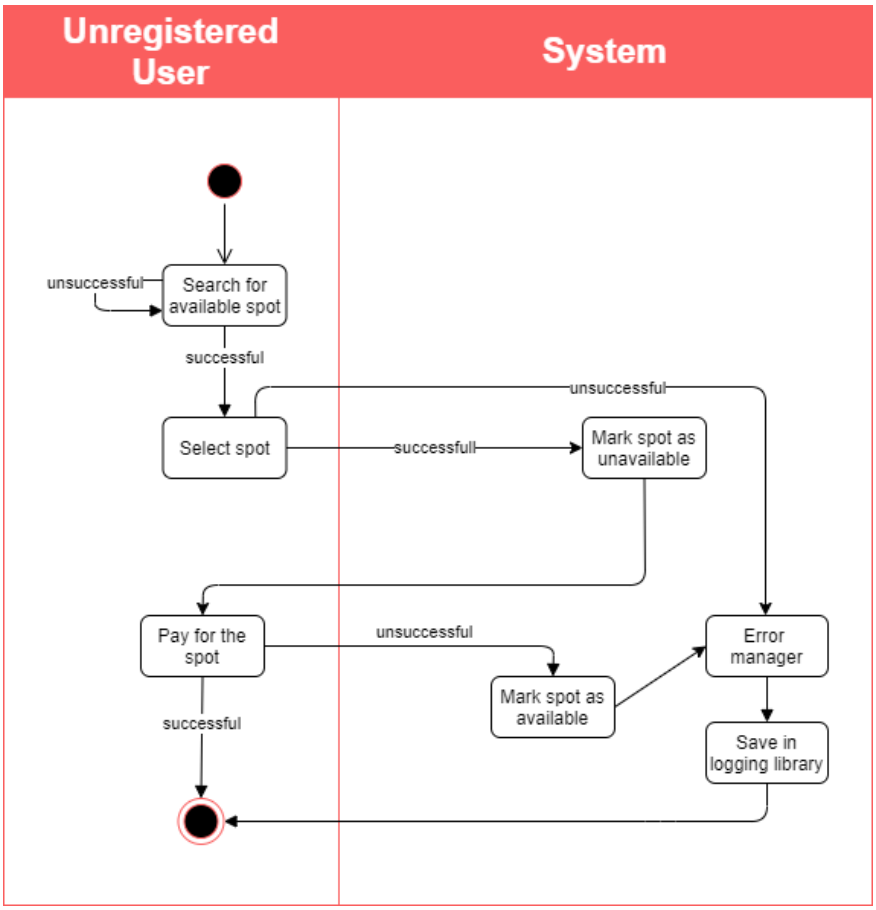
5. Структура на процесите

5.1. Структура на процеса “Заплащане на абонамент за паркомясто от регистриран потребител”



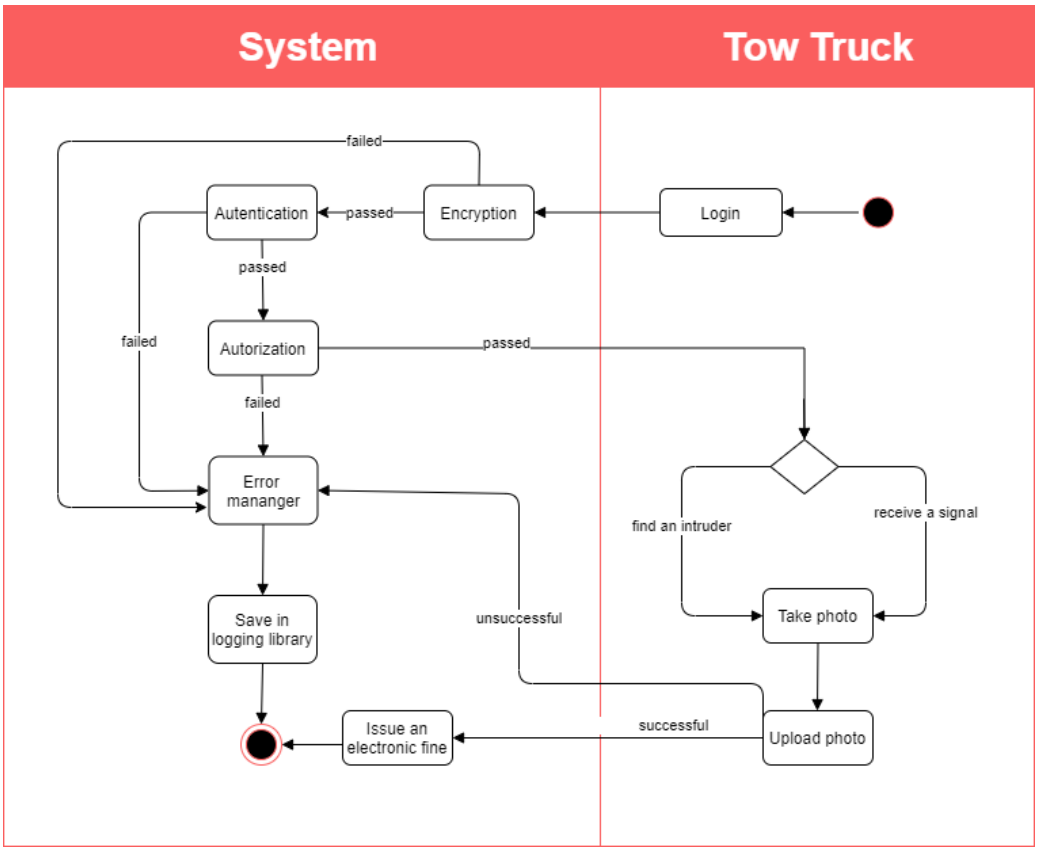
Опцията за “заплащане на абонамент” е валидна само за регистрираните потребители. В системата има няколко групи потребители затова след логване в системата и преминаване през криптирането на информация се преминава към автентикация и авторизация на потребителя. Ако данните на потребителя не са коректни се изпраща подходящо съобщение и се записва в logging library. След като се установи, че данните на потребителя са коректни и той се идентифицира със статус регистриран потребител може да премине към същинското заплащане на абонамент. Първо нещо, което трябва да избере потребителя това е номерът на желаното от него паркомясто. След това е необходимо да зададе желан период за абониране. Ако мястото вече е заето за този период потребителят се уведомява и бива подканен да избере нов период. Ако мястото не е заето системата от своя страна маркира мястото като заето, а потребителят преминава към следващата стъпка-избиране на начин за плащане. Ако плащането е успешно абонирането за паркомястото е финализирано, а ако е неуспешно системата маркира мястото като свободно и потребителя получава подходящо съобщение за грешка.

5.2. Структура на процеса “Намиране на свободно място и заплащане на съответното паркомясто от Гост”



Функционалността за заплащане на паркомясто е достъпна дори и за нерегистриран потребител. Заплащането започва като потребителя търси желано от него паркомясто и ако то е свободно въвежда номера на мястото в системата. Ако мястото е свободно системата автоматично го записва като заето, а потребителя отива към плащане. Ако плащането е успешно потребителя се финализира наемането, а ако не е системата променя състоянието на свободно, изпраща нотификация на потребителя и записва грешката в logging library.

5.3. Структура на процеса “Засичане на нарушител и издаването на елекронен фиш”



Функционалността “засичане на нарушител и издаване на електронен фиш” е валидна само за групите по контрол. В системата има няколко групи потребители затова след влизане в системата и преминаване през криптирането на информация се преминава към автентикация и оторизация на потребителя. Ако данните на потребителя не са коректни се изпраща подходящо съобщение и се записва в logging library. След като се установи, че данните на потребителя са коректни и той се идентифицира със група по контрол може да премине към същинската част на тази функционалност. Нарушителите могат да бъдат открити по 2 начина- или директно от групите по контрол, докато патрулират или да е получен сигнал за нарушител. Групите по контрол трябва да заснемат и качат снимката в системата, а тя от своя страна генерира автоматично фиш за нарушението. Ако качването на снимката е неуспешно се изпраща съобщение на групите по контрол и се записва в logging library.

6. Архитектурна обосновка

Системата е реализирана на базата на следните изисквания:

- **Свободните паркоместа се идентифицират от система от дронове, които обикалят града и заснемат зоните за паркиране (отгоре).**

Това изискване се явява като фундаментално за системата за следене и управление на свободните паркоместа. От огромно значение е използването на дроновете, защото в противен случай би било необходимо наемането на много хора, които да обикалят града, за да идентифицират свободните места, което би било доста по-неефективно и би коствало повече ресурси. То бива осигурено чрез модулът DronesManager, който се намира в Business logic. Чрез него се осъществява контролирането на дроновете и извличането на информация от тях.

- **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.**

  Това изискване се осигурява чрез използването на модулът DroneRouteManager.

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

  За осигуряване на това изискване е реализиран модулът WeatherConnector, който се свързва с външна услуга за получаване на актуална прогноза за времето. Фактът, че сме използвали универсален модул за извличане на тези данни обуславя лесната замяна на външната система при нужда.

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

  Това изискване е от огромно значение за системата, тъй като функционалността за намиране на свободни места е основополагаща за системата. Свободните места трябва да се заснемат дистанционно и да се автоматизира разпознаването на свободните места.Това изискване се осигурява от модулът FreeSoptRecognition, който черни заснетите изображения съответно от модулът DroneImages.

- **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

От голяма важност за системата е да се разделят отделните потребители на групи, съответстващи на тяхната функция. Всяка група потребители има нужда от различни неща в системата и следователно самите функционалности, които ще са им предложени трябва да се различават. Това е осигурено от главният модул User, който осигурява наличието на различни функционалности на отделните групи потребители.

- **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.**

  Това изискване е осигурено от модулът DroneStatusManager, който следи за актуалното състояние на дроновете. При извънредни стойности за състоянието на дроновете, изпраща съобщение за повреда към NotificationManager, който от своя страна изпраща съобщение към модулът Receive Reduced Visibility Alert.

- **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.**

  Операторът на системата притежава функционалност за задаване на интервал за обновяване на свободните места, осигурена чрез модулът Set a time interval. Този интервал се препраща към DroneImages, а той от своя го задава на дроновете за заснемането на снимка, като се има в предвид, , че FreeSpotRecognition може да разпознае свободните паркоместа за 5сек.

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

  Системата трябва да е налична във всяка една ситуация, за да може да се следи реалната обстановка, а трайно намалена видимост ще доведе до невъзможност за актуализация на паркоместата. Затова при невъзможност да се заснемат паркоместата от дроновете, системата трябва бързо да извести оператора, за да се вземат мерки за актуализиране на информацията без използване на дронове. Това известява не случва чрез модулът Receive reduced visibility alert, който се получава посредством NotificationManager от модулът DronesReducedVisibility.

- **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.

Всеки регистриран потребител ще има допълнителна функционалност да заплаща абонамент за определено паркомясто. Заплащането на абонамент дава сигурност на потребителите на системата, че тяхното паркинг място няма да бъде заето, когато им е нужно и това го определя като задължително функционално изискване. То се осъществява чрез модулът Make a subscription.

- **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.**

  За удобство на потребителите системата предлага таксуване динамично. Например в ден с лоши метеорологични условия цената на час на паркомястото ще бъде по-ниска в сравнение с хубав слънчев ден. Също така през нощта цената за час също ще бъде по-ниска. Това изискване е осигурено от системата чрез модулът ParkingSpotManager, който от своя страна черпи информация за метеорологичните условия от WeatherConnector.

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

  Всяка система, която е свързана с някакъв тип плащания трябва да разполага с различни методи за плащане за максимално удобство на клиентите. Това дава достъпност на повече хора и не ги задължава да използват конкретен начин за плащане. Това изискване е осъществено чрез модулът PaymentConnector, който се свърза с външната система Gate2Payments. Бъдещето добавяне на начини за плащане е осигурено също от този модул, тъй като то е универсален и лесно може да бъде заменена външната система.

- **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.**

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

  Това изискване се осъществява чрез модулите Authentication и Authorization, през които се преминава при всяко логване в системата.

- **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.**

  Групите по контрол са от съществено значение, защото ако ги няма, то тогава всеки ще може да паркира, където си пожелае без никакви последствия и никой няма да плаща за паркомясто. За качване на заснетата снимка на автомобила се използва модулът Send photo to system, достъпен само за групите по контрол. Фишът се генерира автоматично чрез модулът Fine Generator. След като е издаден фиш за глоба и снимка, те трябва да са максимално леснодостъпни за всеки потребител, затова те биват качвани от адмна директно на уеб сайта, използвайки модулът Upload to web site photo and fine.

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

  Всеки регистриран потребител има възможност да подава сигнал за неправомерно заето място чрез модулът Report, който изпраща подходящо съобщение до NotificationManager, който от своя страна уведомява групи по контрол посредством модулът Receive a signal.

- **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).**

  Системата реализира идея, която след добиване на популярност ще бъде използвана от много хора и ще бъде натоварена в определен интервал от време. Това качествено изискване е значимо за системата, тъй като във светлата част на работният ден напливът от хора е най-голям и тогава системата е изключително важна за потребителите.Изпълняването на това изискване се вижда в deployment диаграмата, където са изобразени

резервни на основните сървъри. По този начин ако има срив в системата за минимално време, тя ще бъде възстановена. Същата тактика е използвана и за базата от данни, за да може при злонамерена атака към нея и загуба на някакво количество информация, то да не бъде напълно изгубено.

- **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.**

  Използваме външна база от данни, която ни позволява да съхраняваме каквото количество информация искаме срещу заплащане от наша страна. Смятаме, че това е добър избор, защото по този начин бихме могли да доставим едно допълнително ниво на защита, тъй като самият сървър, на който е разположена базата от данни се поддържа и от други лица, не само от екипът на DronesPark.

Тактиките, които са използвани за реализацията на системата са следните:

- **Тактика за отстраняване на откази**
  За отстраняване на откази сме използвали добре познатата тактика за активен излишък. В нашата системата тя е реализирана в два аспекта- от една страна имаме копие на базата от данни, а от друга имаме резервни сървъри на основните- написани са с различни алгоритми, различни програмни езици и т.н.т. Наясно сме с фактът, че това внася допълнителна сложност, изисква повече ресурси и също така имаме и второ място където да бъдем атакувани. Въпреки това смятаме, че тази тактика е доста полезна за нашата система и би спестила редица проблеми при срив в системата, тъй като до няколко секунди можем да върнем системата отново в експлоатация.
  Друга тактика, която използваме е следене на процесите. Тя се осъществява чрез модулът NotificationManager, който стриктно следи процесите в системата. Ако даден процес откаже например не връща никакъв резултат, то мониторинга има за цел да информира съответните отговорни лица, за да се премахне или преинициализира процеса.

- **Тактика за производителност**
  За да подобрим производителността на нашата система използваме тактиката за арбитраж на ресурсите, реализирано чрез модулът Schedualer. Всяко едно събитие се приоритизира в зависимост от неговата важност. Това събитие, което има по-голям приоритет, то ще получи достъп до ресурсрите. Тук обаче може да възникне нов проблем- високо приоритетни заявки постоянно да избутват най-ниско приоритетната. Този проблем е решен чрез

допълнителен параметър наречен времева граница- крайно време, за което дадена заявка трябва да се изпълни. Така малко преди да се достигне до тази времева граница се вдига приоритета на нископриоритетния процес, за да се изпълни той.

- Тактика за сигурност

За сигурността използваме тактиките:
Автентикация на потребителите – проверка за това, дали потребителя е този, за който се представя (пароли, сертификати, биометрика и т.н.). Също така използваме скриване на фунционалностите зад потребителско име и парола.
Оторизация на потребителите – проверка за това дали потребителят има достъп до ресурсите, които иска да достъпи. Както стана ясно в системата различни потребители имат различни права.
Конфиденциалност на данните – посредством криптиране (на комуникационните канали и на постоянната памет). Криптирането е начин за скриване на информация при пренасянето ѝ. Неоторизираните хора не могат да черпят данни.
Ограничаване на експозицията- това е реализирано при базата от данни. В системата никой компонент не може да черпи информация от базата от данни директно, а само посредством DBContoroller.
Ограничаване на достъпа- използвани са защитни стени, което е показана при структурата на внедряването.
Също така използваме и тактика за откриване на атаките чрез log-ване. . Чрез него можем да проследим действията на извършителя. Много често тези логове биват атакувани затова сме взели мерки за достъп до тях само при определени ситуации. Важно е да разберем какъв е типът на потребителя, какво е направил той, че да успее да пробие системата, за да успеем да се поправим в бъдеще. Наясно сме, че в 21 век почти не съществува система, която да не може да бъде хакната. Ето затова имаме модул BackUpManager, който се грижи за застраховане на системата от загуба на данни.

- **Тактика за изпитаемост/тестване**
  Тук отново имаме тактиката за log-ването. В LoggingLibrary се записват всички процеси, които протичат в модулите.

7. Допълнителна информация

номер на паркомясто[1] - Системата възприема паркоместата с уникален номер-комбинация от букви и цифри като буквите определят района, а цифрите номера на конкретното паркомясто.

номер на дрона[2] - Системата разпознава дроновете по съответните им уникални номера, които са записани при регистриране на дрона в системата.