Курсов проект

по

Софтуерни архитектури и разработка на софтуер

на тема

VeloCity

система за отдаване на велосипеди под наем

Изработили:

Ивана Драгнева, 62480

Весела Петрова, 62494

Софтуерно инженерство, 2-ри курс

Съдържание:

1. Въведение

1.1. Обща информация за текущия документ

1.1.1. Предназначение на документа

Документът служи за представянето на софтуерната архитектура на системата **VeloCity** - система за отдаване на велосипеди под наем. Написана е по достъпен начин и цели да спомогне работата по системата на всички заинтересовани лица.

1.1.2. Описание на използваните структури

1.1.2.1. **Декомпозиция на модулите** - Целта на тази структура е да покаже разпределението на функционалността на системата по модули, като основни такива са - **Server, Client, Velo, Payment, Security, Database, UserInterface, SystemUsersManager** и **Map**. Към повечето от тях има прилежащи подмодули, които са подробно описани в **т. 2.2**. Декомпозицията на модулите е полезен изглед на системата, защото когато знаем в кой модул се намира определена функционалност, се предполага лесно да можем да променим имплементацията, ако се налага.

1.1.2.2. **Структура на процесите** - Структурата на процесите изобразява подробно всички стъпки, през които потребител, желаещ да наеме велосипед, преминава до получаване на местоположение на дадено превозно средство. Подробна информация - **т. 3.1**.

1.1.2.3. **Контекстна диаграма** - Тази структура показва потока данни, който се обменя между системата и различните участници в нея - външни системи или различни групи потребители. Подробната информация се намира в **т. 3.2**.

1.1.3. Структура на документа

Документът се състои от четири основни секции:

Секция 1: Въведение - тя служи за обобщаване на съставните части на системата и помага за лесно навигиране из документа. Тук се съдържа и информация за архитектурните драйвери, подкрепени с обосновка за техния избор.

Секция 2: Декомпозиция на модулите - тази секция описва всички модули, които съставят системата, заедно с техните отговорности. Обобщена информация за тази структура може да се намери на т. **1.1.2.1.**

Секция 3: Описание на допълнителните структури - тук са изобразени графично допълнително избраните архитектурни структури и техните описания, както следва – **„Структура на процесите" т.3.1** и **„Контекстна диаграма" т. 3.2**. Обобщена информация за двете структури може да се намери на **т. 1.1.2.2** и **т. 1.1.2.3.**

Секция 4: Архитектурна обосновка - тази секция посочва изискванията, според които се разработва системата и е подкрепено с аргументи как всяко едно от тях удовлетворява софтуерната архитектура на VeloCity.

1.2. Общи сведения за системата

1.2.1. Функционални драйвери

I. **Регистрация на потребител** - Личните данни, които потребителите въвеждат при регистрация са напълно защитени и са достъпни единствено от наблюдателите. Те имат право да ги използват, като се свържат чрез оставените в профила имена, ЕГН и друга информация за връзка, ако възникне нередност с наетия от потребителя велосипед.

II. **Правомощия на системните наблюдатели** - Входът им в системата се осъществява чрез уеб приложението. Те имат право да наблюдават местоположенията на велосипедите и да сигнализират на определените групи потребители със специални функции при възникнала нередност.

III. **Възможности на системните администратори** - Входът в системата за тази група потребители е както при наблюдателите - чрез уеб приложение. Администраторите могат да конфигурират системата, за да удовлетворят желанията на всички заинтересовани лица, както и да следят дали работи според изискванията.

IV. **Взаимодействие с онлайн услуги за географски карти** - Системата се нуждае от точно местоположение на потребителя наемател,

който е подал заявка за намиране на най-близък велосипед, както и от начин да следи за това дали превозното средство се използва в обхвата на града и дали се оставя на правилните стоянки. За тази цел ще е необходимо системата да е интегрирана с онлайн географски карти като Google maps, BG maps, Open Street maps и ще има възможност за добавяне на нови в бъдеще.

V. **Функционалности, свързани с велосипедите** - Наемателите си плащат определено време, за което им е необходим велосипед. Това време се регистрира на сървъра на системата и когато изтече, се изпраща напомнящо съобщение до потребителя, който трябва да го остави на най-близката стоянка или при необходимост да го замени с друг. Докато не е нает, неговата батерия се зарежда на тази станция. Един велосипед може да се вземе само с определено количество минимално заредена батерия, установено от системата.

VI. **Таксуване на услугите** - Велосипедът може да се използва свободно в обхвата на града. Заплащането за тази услуга от обикновените потребители става посредством кредитна карта (Credit/Debit), чрез съобщение, което се изпраща до личен телефонен номер или чрез предварително закупени талони. Ако се използва последният начин, трябва да се въведе в системата ръчно или автоматично (QR-code) кода, който се намира върху талона.

1.2.2. Нефункционални драйвери

I. **Системата трябва да поддържа следните групи потребители:**
   - Наемател на велосипед (обикновен потребител)
   - Член на група по техническа поддръжка на велосипедите
   - Системен администратор (техническа софтуерна поддръжка)
   - Наблюдател/отговорник по използването на велосипедите

   Всяка група потребители си има определено предназначение, за да може системата да функционира правилно.

II. **Изправност** - За да се осигури изправността на системата, се използват наличните на велосипеда GPS устройство и смарт-сензори за самодиагностика. Състоянието и безопасността на потребителите е важно, затова при установяване на неизправност, причинена от ПТП или друг инцидент, се изпраща автоматично сигнал до спешна помощ в рамките на 1 секунда след засичане на инцидента. В рамките на 5 секунди се известява и

наблюдателя на системата. Ако проблемът е технически, се изпраща сигнал до екипа по техническа поддръжка, който в рамките на половин час трябва да диагностицира повредата и да вземе мерки за отстраняването ѝ. Изправността е важна качествена функционалност, защото при чести сривове доверието на потребителите би намаляло и дори може да ги откаже да използват системата, колкото и полезна да е през останалото време.

III. **Сигурност** - Възможно е, докато потребителят (наемател) използва велосипеда, да се разсее и да напусне обхвата, в който е разрешено да се намира. Когато съзнателно се напуснат рамките на града, трябва да се сигнализира наблюдателя в рамките на 1 минута, както и ако се загуби връзка с даден велосипед - в рамките на 10 секунди, като и за двата сигнала се изпращат данни за движението на велосипеда в последните 30 минути. При втория вид сигнал се изпраща и най-вероятната позиция, на която се е намирал велосипеда в момента на изгубване на връзката. За изчисляването на приблизителното местоположение се използва изкуствен интелект. Сигурността на велосипедите и системата също не трябва да се пренебрегва, за да се възпрепятства неразрешената им употреба и да може всички съвестни потребители да се възползват от услугите.

IV. **Наличност** - Системата трябва да бъде 99.999% налична в по-заетата част от денонощието, когато велосипедите се използват най-много и могат да са полезна част от ежедневието на потребителите (наематели). През останалото време, от 2:30 - 5:30 часа, се допуска ремонт и профилактика.

V. **Надеждност** - В часовете на денонощието, когато велосипедите се използват най-много, системата трябва да бъде устойчива на претоварване от потребителски заявки, за да може да се предоставят услуги на колкото може, по-голямо количество потребители.

VI. **Изменяемост** - VeloCity ще бъде лесно изменяема система с оглед на това да могат да се интегрират нови видове онлайн карти, както и да има възможност да се добавят други функционалности, възникнали в бъдеще.

VII. **Системата трябва да е достъпна през мобилно приложение и през уеб приложение за специфични групи потребители** – Архитектурата на системата зависи от това през какви устройства ще бъде налично приложението. От обикновените потребители VeloCity ще се достъпва през мобилно приложение, което може да се изтегли на всякакви видове мобилни устройства, които имат достъп до мобилен интернет или WIFI. Две групи, които се занимават с поддръжката на системата и проследяването на велосипедите - администратори и наблюдатели, ще достъпват VeloCity през уеб браузър.

1.3. Терминологичен речник

1.3.1. **Софтуер** – съвкупността от цялата информация от инструкции и данни, необходими за работата на всяка електронноизчислителна машина.

1.3.2. **Софтуерна система** - система от взаимодействащи компоненти, базирани на софтуер, представляващ част от компютърна система.

1.3.3. **Външна система –** отдалечена софтуерна система, която е източник на данни или предоставя функционалности, които настоящата система използва

1.3.4. **База от данни (Database) -** колекция от логически свързани данни в конкретна предметна област, които са структурирани по определен начин.

1.3.5. **Декомпозиция -** софтуерна структура, показваща как системата се разделя на отделни модули. Типовете елементи изграждащи тази структура са модули, като е възможно даден модул да има подмодули.

1.3.6. **Модул –** логически обособена софтуерна единица. Част от една програма.

1.3.7. **Структура на процесите -** софтуерна структура, показваща даден процес през какви условия и действия преминава.

1.3.8. **Контекстна диаграма -** диаграма, която дефинира границата между системата или част от системата и нейната среда, показваща обектите, които взаимодействат с нея.

1.3.9. **Структура на внедряване -** софтуерна структура, която показва как софтуера се разполага върху хардуера и комуникационното оборудване

1.3.10. **Процес –** съвкупност от стъпки, която изгражда логическо действие и стига определена цел

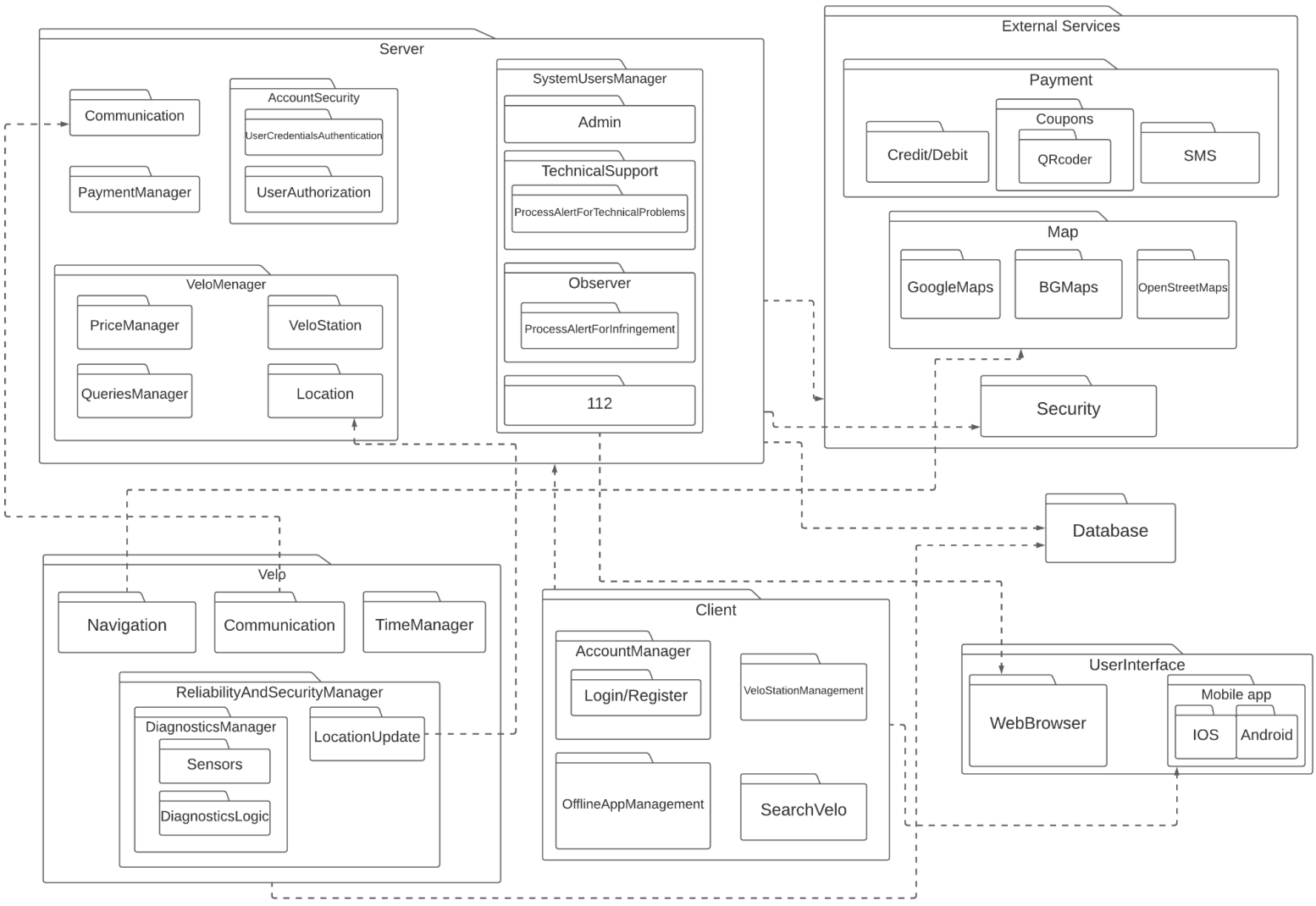
1.3.11. **Потребител –** човек, който използва софтуерна услуга

1.3.12. **Интерфейс (interface)** - абстрактна структура, която предоставя дефиниция на определени свойства които дадено приложение трябва да имплементира.

1.3.13. **API (Application programming interface) -** Приложно-програмният интерфейс е интерфейсът на изходния код, който операционната система или нейните библиотеки от ниско ниво предлагат за поддръжката на заявките от приложния софтуер или компютърните програми.

2. Декомпозиция на модулите

2.1. Общ вид на декомпозицията на модули за системата



2.2. Подробно описание на всеки модул

2.2.1. Client – Този модул се състои от всички функционалности, на които имат право обикновените потребители след регистрация или вход.

2.2.1.1. AccountManager – Модулът предоставя възможност за регистрация и вход на хората, желаещи да си наемат велосипед от системата.

2.2.1.1.1. Login/Register – В модулът ще бъде конкретизирана логиката, необходима за успешна регистрация и правилен вход в системата от даден потребител, тоест ще се наложи използването на модула **AccountSecurity**. В **Register**

Interface е добавен следния метод: *public void* ***register****(String name, String egn, String mobileNumber, String password)*, а в **Login Interface** - *public void* ***login****(String name, String password)*.

2.2.1.2. VeloStationManagement - Модулът представлява възможност на потребителите да подават заявки за информация относно стоянките из града. По този начин те научават за най-близките свободни места, където могат да оставят наетия от тях велосипед. VeloStationManagement си комуникира с VeloStation пакета на сървъра. **VeloStationManagement Interface** съдържа следния метод:

- *public void* ***findVeloStations****(Location userLocation)*

2.2.1.3. OfflineAppManagement - Този модул позволява на системата да следи времето, за което потребителят използва велосипеда, когато той е с изключен интернет или е излязъл от приложението.

2.2.1.4. SearchVelo - Модулът се използва за търсене на подходящ велосипед възможно най-близо до местоположението на потребителя. За да попадне велосипед сред възможните за наемане, трябва батерията му да е заредена на поне **X%**. Затова този модул се свързва с модула QueriesManager на Server компонента и там се обработва потребителската заявка. Тоест **SearchVelo Interface** има следната функционалност:

- *public void* ***makeSearchVeloQuery****(Location userLocation)*

2.2.2. Server - Модулът съдържа бизнес логиката на системата, тоест основните функционалности, свързани с управлението на велосипедите, както и проверката и защита на данните на потребителите. В сървър модула се намира и конфигурирания параметър **T**, който представлява максималното време за използване на определен велосипед.

2.2.2.1. SystemUsersManager - Предоставя функционалностите, използвани от потребителите, които поддържат системата и се грижат за изправността и сигурността ѝ. Всяка група такива потребители се разглежда като отделен модул, в който се намира специфичната за всеки бизнес логика, описана в документа с изискванията към системата. Модулът се свързва с

базата данни, от където ще взима необходимата информация, както и ще получава сигнали и заявки от други компоненти на Server модула.

2.2.2.1.1. **Admin -** Според изискванията на VeloCity администраторите ще могат да конфигурират системата и да следят за правилната ѝ работа. Следователно тук ще се намира кода, който ще реализира тази функционалност.

2.2.2.1.2. **TechnicalSupport -** Техническият екип има за задача да следи анализите, които са записани в базата данни за всички велосипеди и най-вече се занимават с връщането в изправност на повредени превозни средства. Този модул ще трябва да има достъп до базата данни и по-специфично до диагностиките на велосипедите и тяхната локация.

2.2.2.1.3. **Observer -** Тази група потребители ще се грижи за сигурността на превозните средства, тоест при напускане обхвата на града от даден велосипед, те трябва да се отзоват възможно най-бързо до последното местоположение, записано в базите данни. Също така при случай на катастрофа, те ще бъдат сигнализирани от системата, за да могат да реагират на случилото се и отново ще им трябва точна локация, изпратена от велосипеда. Observer модула се нуждае от достъп до Database модула, за да получава информация относно местоположение на велосипедите.

2.2.2.2. **AccountSecurity** - Това е подмодулът, в който е логиката, свързана с автентикацията и ауторизацията на потребителските данни. Чрез автентикацията се потвърждава, че въведените данни са на правилния потребител, а чрез ауторизацията се дават правата за достъп до ресурсите на системата.

2.2.2.2.1. **UserCredentialsAuthentication** - Модулът проверява дали въведената информация присъства в базата данни на системата. **Authentication Interface** включва метода:

- *private void* ***authenticate****(String name, String egn, String mobileNumber, String password)*

2.2.2.2.2. **UserAuthorization** - Модулът позволява на потребителя да използва определените за него функционалности на системата. Този модул е свързан и с front-end частта, където

ще се изобразяват само детайлите, свързани с функционалностите на потребителя. **Authorization Interface**:

- *private void **authorize**(User user)* - Методът ще проверява до кои функционалности и ресурси на системата има достъп дадения потребител и спрямо тези проверки ще се изобразява front-end частта, за да няма неправилно показана функционалност в user interface-а или пък да се даде неправомерен достъп до данни на системата.

2.2.2.3. VeloManager - В този модул се помещава най-важната/основна логика, а именно тази свързана с велосипедите (центъра на цялата система). Негови подмодули са:

2.2.2.3.1. QueriesManager - Този модул ще се занимава с обработката на изпратените към сървъра заявки от потребители за наемане на велосипед. Ще се използва локацията на наемателя, за да се установи най-близката стоянка с велосипед, чиято батерия е заредена на поне **X%**, и ще се върне отговор на заявката. За осъществяване на заявките от този тип се използва VeloStation модула, който има достъп до по-подробна информация. **QueriesManager Interface** следва да има методите:

- *private Location **getLocation**(User user)*
- *private Location **searchVelo**(Location userLocation)* - този метод приема локация на потребител, която може да се вземе чрез горния getLocation(); ще се върне отново обект от тип Location, който ще съдържа координати до най-близкия велосипед.

2.2.2.3.2. PriceManager - Модулът ще обновява дължимата сума според изминатото време **T**, в което велосипедът се използва от наемателя. PriceManager ще комуникира с базата данни и с прилежащи на сървъра модули, както и с Velo модула. **PriceManager Interface** ще има следния метод:

- *private void **updatePrice**(PriceUpdate update).*

Промените на цената се записват в базата данни, като и сървъра се сигнализира за тях. Екипът по поддръжка на системата има достъп до тези данни, за да може да проверява дали промените са правилни. Когато свърши

предвиденото време за използване на дадения велосипед, ще се изпрати и известие до велосипеда с крайната цена, която трябва да заплати наемателя.

2.2.2.3.3. **VeloStation** - Модулът ще обработва информацията, която ще се изпраща от всички стоянки. Например, дали е свободна или пък е заета и на колко процента е зареден велосипеда, прикачен към нея. VeloStation ще има връзка с по-горе споменатите QueriesManager от VeloManager и VeloStationManager от Client модула, както и свободно ще може да общува с всички останали вътрешни модули за сървъра. В този компонент ще се обработват заявки от QueriesManager, свързани с това дали има подходящи велосипеди в близост до подаденото като входен параметър местоположение. Следните методи ще бъдат достъпни в интерфейса на модула - **VeloStation Interface**:

- *private double* ***checkBattery****(Station st)*
- *private bool* ***isFreeStation****(Station station)*
- *private Location* ***sendLocationOfFreeStation****(Location userLocation)* - този метод се възползва от горния, за да направи проверка за най-близката свободна стоянка, спрямо полученото местоположение на потребителя като входен параметър на метода. В този метод се намира приложение и за първия **checkBattery**, защото даденият велосипед трябва да е зареден на поне **X%** (според изискванията за системата).

2.2.2.3.4. **Location -** Модулът служи за динамична и своевременна проверка за това дали даден велосипед се намира в обхвата на града, или го е напуснал съзнателно или не. Има постоянна връзка с модула LocationUpdate, който е част от Velo пакета. Също така се налага постоянно записване на текущо изпратените координати от превозното средство в базата данни на системата, тоест Location общува с Database модула. Ако велосипед е напуснал разрешения за използване обхват, Location пакета изпраща сигнал до сървъра, който се предава към лицата по поддръжка на системата или по-конкретно наблюдателите. Затова имаме

връзка със SystemUsersManager модула. **Location Interface** притежава следните методи:

- *private bool* ***isVeloInCity****(Velo velo)*
- *private void* ***updateVeloLocationInfoToTheDatabase****(Location veloLocation)*
- *private void* ***sendSignalToObserver****(String message)*

2.2.2.4. PaymentManager - Отговаря за управлението на цената, която трябва да заплати наемателя, след като е използвал услугите на системата. Системата предоставя няколко начина за заплащане, следователно логиката за всеки начин се намира в този модул. PaymentManager си комуникира с външния модул Payment, както и модула Client. От клиентския пакет, както и от Payment пакета се избира метод за плащане и се изпраща известие до сървъра. Заявката се обработва на сървъра от PaymentManager и ако плащането става чрез кредитна карта, информацията ще се получава чрез поток от данни от клиента и след това ще се изпраща до Payment. PaymentManager има допълнителна функционалност за обработката на СМС съобщения. Номерът, от който се изпраща съобщението е обвързан с профила на наемателя и така ще бъде сигурно плащането. Ако се избере третия подход за плащане чрез закупен талон, кодът ще се изпраща отново от клиентския модул и ще бъде обвързан с определен профил, така че ще се избегнат всички обърквания за данните и заплащането ще бъде отчетено правилно. Модулът има и специална логика за разчитане на QR-code. **PaymentManager Interface** методи:

- *private void* ***processPayment****(User user, PaymentMethod pMethod, Payment payment)*
- *private void* ***processCreditCardPayment****(User user, Payment payment)*
- *private void* ***processSMSPayment****(String mobileNumber, User user, Payment payment)*
- *private void* ***processCodePayment****(String code, User user, Payment payment)*
- *private void* ***processQRcodePayment****(QR code, User user, Payment payment)*

2.2.2.5. **Communication -** Модулът отговаря за изпращането и получаването на съобщения (данни) от и за всички останали модули (с изключение на Maps). Чрез Communication пакета се създава общ формат на съобщенията и по този начин се улеснява модифицирането на системата чрез добавяне на нови модули на по-късен етап. За да се подсигури общуването между различните компоненти на системата се използва модула Security, който ще обработва всички заявки/съобщения.

2.2.3. UI - Този модул предоставя специфична имплементация на потребителския интерфейс, който се визуализира по различен начин в зависимост от какво устройство потребителят е влязъл в системата. Всяка група потребители имат специфични функционалности, които ще бъдат показани и лесно използваеми чрез UI модула. VeloCity има потребителски интерфейс за следните платформи:

2.2.3.1. **WebBrowser** - В този вариант системата е налична само за екипите, които поддържат системата и се грижат за правилното й използване, сигурността и изправността. Всяка група потребители ще вижда различен потребителски интерфейс. Например администраторите могат да конфигурират системата и ще имат специални бутони и прозорци, чрез които да си изпълняват задълженията. Наблюдателите пък ще имат възможност да следят всички велосипеди на картата и ще получават сигнали за нередности. Екипът по техническа поддръжка ще може да достъпва анализите на определени велосипеди чрез потребителския интерфейс и също ще получава сигнали за неизправност във велосипедите.

2.2.3.2. **MobileApp -** Мобилната версия на системата е направена за потребителите, които искат да си наемат велосипед, и също ще имат достъп до специални функционалности през потребителския интерфейс, чрез които да намерят подходящо превозно средство в тяхна близост. За да покрие нуждите на всички потребители, системата има мобилно приложение и за двете най-разпространени операционни системи:

2.2.3.2.1. **IOS**

2.2.3.2.2. **Android**

2.2.4. Database - В този модул се управляват базите от данни, които съхраняват информацията. Видовете информация в базата включват данните на потребителя, въведени при регистрация, както и тези на велосипеда, получени при анализирането му за технически неизправности или някакви повреди при катастрофа. Също така в базата данни се обновява периодично крайната цена, която трябва да заплати потребителя според изминалото време, в което е използвал даден велосипед. В базата данни ще се записва и обновява своевременно информацията, свързана с локацията на велосипеда, за да може при нарушения да се вземат мерки. Това означава, че съдържанието на базата данни трябва да може да се модифицира лесно и сравнително бързо да се търси информация в нея. За да бъде това възможно, ще използваме външно за системата API, което ще се грижи за тези нужди на VeloCity системата. Модулът осъществява връзка със Server модула и също така с Velo модула, за да се знае текущото местоположение и състояние на велосипеда, както и крайната цена, която трябва да се заплати.

2.2.5. Velo - Модулът отговаря за управлението на велосипедите и анализирането им за повреди, както и за изпращане на данни към сървъра.

2.2.5.1. Navigation - Модулът има за задача динамично да определя маршрута, по който се движи даден велосипед. Всички маршрути, които покриват разрешения за използване обхват, ще са предварително инсталирани. Модулът ще комуникира с LocationUpdate и Communication пакетите от Velo модула, както и с Maps пакета. Чрез Maps ще се визуализира картата на велосипеда и ще се изчислява точното местоположение за по-лесно проследяване на превозното средство при напускане на града. Местоположението ще се изпраща до LocationUpdate пакета. Navigation Interface ще има следните методи:

- *public Location* ***getVeloLocation****(Velo velo)*
- *public void* ***sendLocationInfo****(Location veloLocation)*
- *public void* ***showMap****(Map map, Location veloLocation)*

2.2.5.2. **ReliabilityAndSecurityManager** - Модулът съдържа логиката и функционаностите, имплементирани за велосипеда, които осигуряват неговата изправност и сигурност.

2.2.5.2.1. **DiagnosticsManager -** В модула се намира функционалността, свързана с анализирането на състоянието на велосипеда чрез специалните сензори по него и изпращането на сигнали към техническите екипи, ако има техническа повреда. Ако повредата е причинена от участие на велосипеда в катастрофа се известяват незабавно 112 и наблюдателите на системата. Това означава, че модулът е свързан с Communication модула във Velo и Server, както и със SystemUsersManager модула. Сигналите за проблем се изпращат до сървъра, който от своя страна ги изпраща към подходящите подмодули на SystemUsersManager.

2.2.5.2.1.1. **Sensors -** Този модул е полезен за лесно добавяне на нови сензори в бъдеще или премахване на такива, които не са необходими. В **Sensors Interface** се съдържат споменатите функционалности:

- *private void* ***addSensor****(Sensor sensor)*
- *private void* ***removeSensor****(Sensor sensor)*

2.2.5.2.1.2. **DiagnosticsLogic** - Тук се намира имплементацията на функционалностите, свързани с анализирането на велосипеда чрез наличните по него сензори.

- *public Signal* ***sendSignalToTheServer****(Velo velo, ProblemType type)* - Този метод получава като входен параметър велосипеда, който има проблем. В зависимост от това какъв е type параметъра, дали е проблем заради пътно транспортно произшествие, или е технически, се изпраща определен сигнал (Signal) до сървъра, който съдържа необходимата информация (кой е велосипеда/наемателя и къде се намира в текущия момент превозното средство).
- *public void* ***analyzeVelo****(Velo velo)*

- *public void* ***updateDiagnosticsInfoToTheDatabase****(Diagnostics diagnostics)*

2.2.5.2.2. LocationUpdate - Този подмодул се занимава със своевременното обновяване на текущото местоположение на велосипеда, като изпраща данните към сървъра, затова е свързан с Location в Server модула.

- *public Location* ***sendLocationToTheServer****(Location veloLocation)*

2.2.5.3. TimeManager - Модулът се използва за известяване на наемателя на даден велосипед, че позволеното време за използване на превозното средство е изтекло

- *public void* ***showMessage****(String message, Location location)* - методът получава като входни параметри, съобщението, което трябва да се покаже на наемателя, както и локацията на най-близката стоянка, на която трябва да се остави велосипеда.

2.2.5.4. Communication - Модулът служи за комуникация със сървъра и за изпращане на заявки към него. Communication обменя информация с всеки един от подмодулите на Velo компонента.

- *public void* ***sendMessageToTheServer****(String message)*

2.2.6. ExternalServices -

В този модул са поместени компонентите, които използват външни за системата услуги като географски карти или различни начини за плащане. В долните точки са разгледани по-подробно съответните модули и техните подмодули.

2.2.6.1. Payment - Този модул отговаря за различните форми на плащане, когато потребителят желае да наеме велосипед. Системата има задължение да предложи три начина на заплащане, като за всеки един от тях съществува подмодул и външно API, с което комуникират, за да се извърши превода на парите.

2.2.6.1.1. CMC - Модулът позволява на потребителите да заплатят наем на велосипед чрез изпращане на съобщение от мобилния им телефон, като СМС-ът трябва да бъде изписан по определен шаблон, даден от системата, за да бъде валиден.

2.2.6.1.2. **Credit/Debit** - Този подмодул отговаря за трансфериране на пари от банковата сметка на потребителя по повод наем на велосипед.

2.2.6.1.3. **Coupons** - Подмодулът служи за заплащане за определено време с велосипед чрез предварително закупени талони.

2.2.6.1.3.1. **QR-coder** - Модулът служи за въвеждане на QR код в системата, който отговаря на закупен от потребителя талон.

2.2.6.2. **Maps** - Този модул осъществява връзка с външна услуга (API), която изобразява карта на района, издаващ локациите на велосипедите. Картата е нужна и за да се покажат местоположенията на стоянките, където може да се остави наетото превозно средство. Също така е важно и да се следи географското положение на потребителя от наблюдателите и техническия екип в случай на поява на проблем или неизправност. Към този момент системата работи със следните API-та за осигуряване на онлайн услуги за географски карти:

2.2.6.2.1. **Google maps**

2.2.6.2.2. **BG maps**

2.2.6.2.3. **Open Street maps**

2.2.6.3. **Security -** Модулът използва външни за системата услуги, които осигуряват 100% защита на VeloCity - тоест не само на уеб и мобилните приложения, а и на велосипедите. Тъй като цялата информация и данни, необходими за работата на системата, се предават по мрежата, е възможно да станат обект на неправомерен достъп и неправомерно използване, което поставя под риск данните на потребителите и не само. Всеки модул, който изпраща или получава чувствителни данни, най-вече сървъра, разчита на Security пакета за анализиране на заявките и предотвратяване на хакерски атаки.

3. Описание на допълнителните архитектури

3.1. Структура на процесите

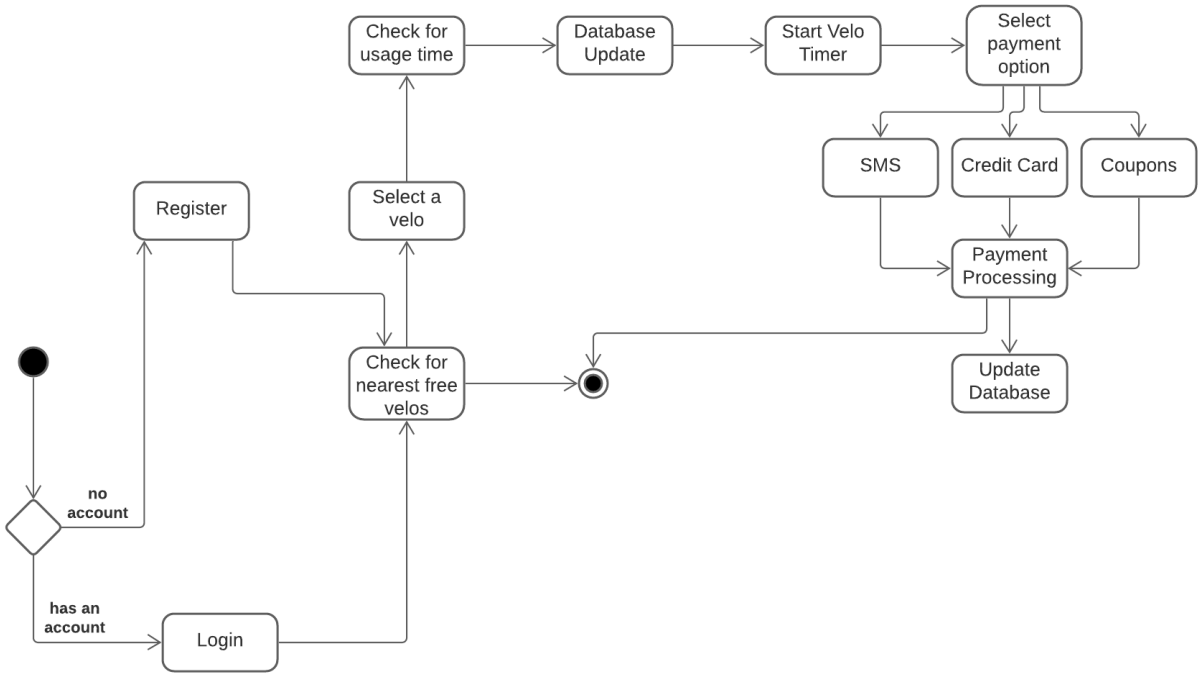
3.1.1. Мотивация за избор

Тази структура е подходяща за изобразяване на най-важната функционалност на системата VeloCity, а именно наемането на велосипед за ползване из града. Целта на структурата е да представи по лесен и достъпен начин, дори и за обикновените потребители, как се извършва наемането на превозно средство. В диаграмата са включени и две важни за системата функции:

Database Update - в базата данни се запазва информацията, която е извлечена от потребителя до него момент, тоест избрания велосипед и времето, за което ще бъде използван, за да се избегнат проблеми, свързани с това повече от един потребител да иска да наеме конкретния велосипед.

Start Velo Timer - стартира се таймера на велосипеда, който отчита времето, за което може да се използва избран велосипед от определен потребител.

3.1.2. Първично представяне



3.1.3. Описание на елементите

За да се наеме велосипед, е необходимо потребителят да има регистрация в системата. Процесът може да започне от една начална позиция, като се прави проверка дали потребителят има

вече създаден акаунт или няма. Ако няма такъв, то той се регистрира, а иначе влиза в системата. След което потребителят подава заявка за намиране на подходящи велосипеди, тоест такива, които се намират най-близо до него. Наемателят посочва избрания велосипед, а системата проверява за какво време може да се използва точно това превозно средство и връща отговор. На следващата стъпка се запазва информацията в базата данни за определено време (времето, за което ще бъде зает велосипеда). Стартира се таймера на велосипеда, който ще показва оставащото време за използване. Когато свърши определеното време, потребителят трябва да плати дължимата сума, като избира един от трите начина - SMS, талони или кредитна карта. Тези стъпки описват целия процес по наемане на велосипед.

3.1.4. Описание на обкръжението

Използват се външни услуги за изобразяването на карта, по която се движат велосипедите, както и за заплащането на услугите.

3.1.5. Описание на възможните вариации

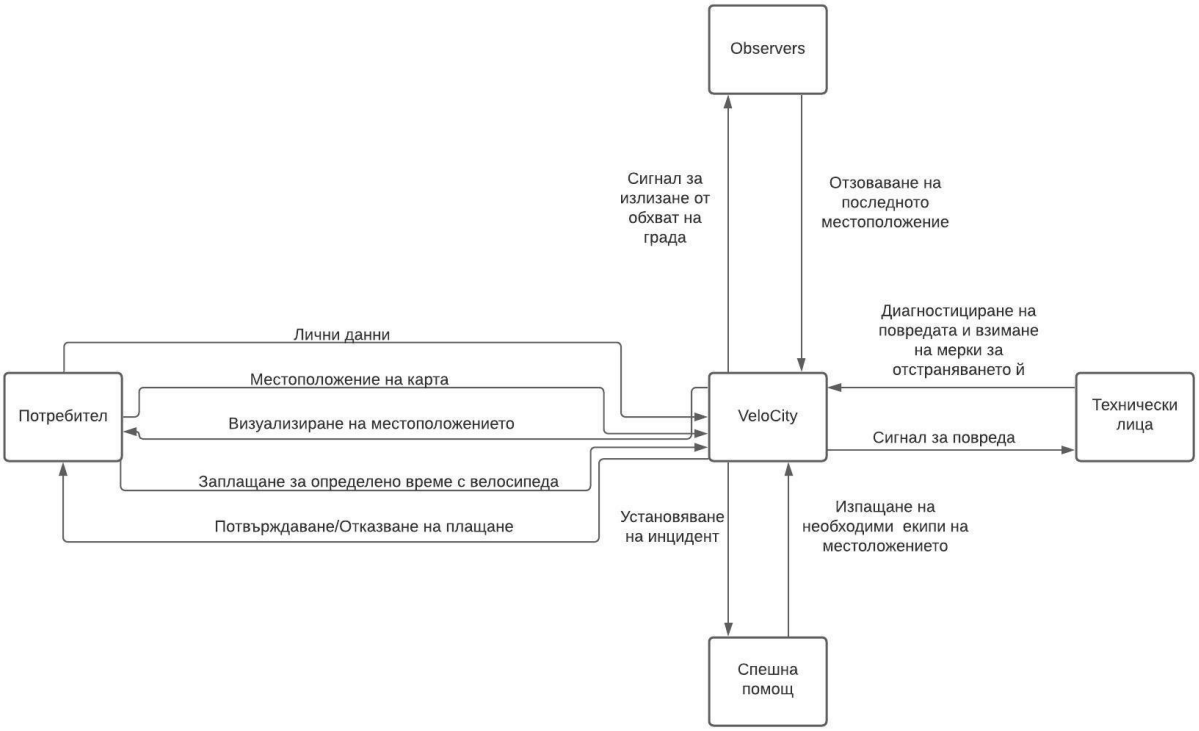
При промяна във функционалностите на системата, добавяне на нови или премахване на съществуващи, следва да бъде обновена и диаграмата на процесите.

3.2. Контекстна диаграма

3.2.1. Мотивация за избор

Системата VeloCity взаимодейства с множество външни източници, които непрестанно обменят информация при изпълнение на различните функционалности. Поради тази причина, най-добър вариант е да се изгради контекстна диаграма. Тя насочва вниманието върху данните, които влизат и излизат от системата и по този начин се следи лесно потока на информация.

3.2.2. Първично представяне



3.2.3. Описание на елементите и връзките

Основните елементи в диаграмата са системата - VeloCity и различните типове потребители, които я използват - потребител, спешна помощ, технически лица и Observers. Връзките между потребителите и системата показват информацията, която се обменя при изпълняване на различните основни функционалности. Стелките, които сочат към VeloCity показват информация, идваща от потребителите към системата, а стрелките от системата към потребителя, обратното.

3.2.4. Описание на обкръжението

Както е дадено в диаграмата по т. 3.1, за местоположението и заплащането се използват външни услуги.

3.2.5. Описание на възможните вариации

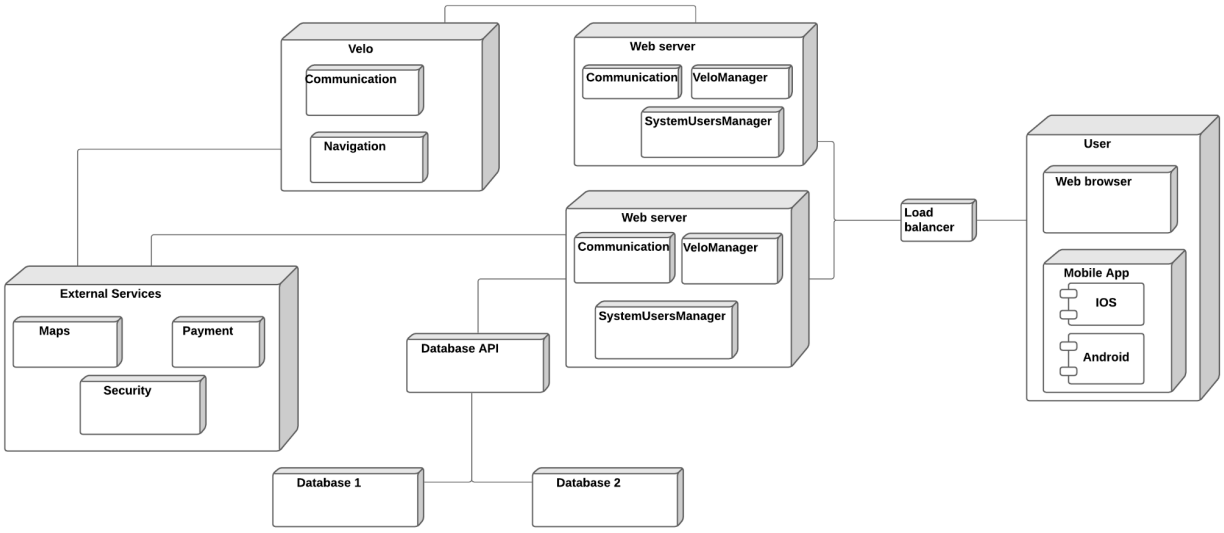
При промяна в системата следва съответно да се поднови и контекстната диаграма на системата VeloCity.

3.3. Структура на внедряването

3.3.1. Мотивация за избор

Структурата на внедряването позволява да се разберат особеностите относно бързодействието, надеждността и сигурността в една система като по този начин отлично би се описала VeloCity, понеже това са едни от нефункционалните й изисквания. Също така отлично се показва как софтуера се разполага върху хардуера и комуникационното оборудване.

3.3.2. Първично представяне



3.3.3. Описание на елементите

3.3.3.1. **Database API** сървърът има за цел контрол на заявките и разпределянето им към отделните инстанции на базата данни, без да натоварва тези сървъри.

3.3.3.2. **Database 1** и **Database 2** са сървъри с копия на базата данни. Обработват заявките от **Database API** и служат за разпределяне, за да се избегне претоварване.

3.3.3.3. **User** сървърът служи за изпращане на данни и заявки от профилите на всички видове потребители - клиентите, които достъпват VeloCity през мобилен телефон и администраторите, които влизат в системата през уеб браузър. Понеже напливът на заявки е голям, се използва модула **Load Balancer**, през който преминават потоците от данни и се разпределят в два **Web server**-а.

3.3.3.4. **Web server** е сървърът, който получава заявките и данните от **User** и ги обработва спрямо необходимата операция. За да се избегне претоварване на системата, са създадени две

инстанции на **Web server**, които балансират с работата и обработката на информацията, която преминава през тях.

3.3.3.5. **Velo** сървърът съдържа всичката необходима информация и свойства, които притежава всеки велосипед, за да бъде в изправност и готов за използване. Той си комуникира с другите модули, като изпраща информация за актуализираното си местоположение и текущо състояние.

3.3.3.6. **External services** поддържа външните услуги, с които системата си комуникира, а именно услуга за плащане, услуга за навигация и визуализиране на географската карта и услуга за защита на данните и велосипедите. Сървърът получава от **Velo** информация за местоположението и състоянието му, която после препраща на **Web Server** за обработка и допълнително трансфериране. Още една дейност бива изпращането на заявки за плащане за използване на велосипеда.

3.3.4. Описание на обкръжението

3.3.4.1. **Map** подмодулът си комуникира с различните видове географски карти, които могат да визуализират велосипеда и неговия маршрут.

3.3.4.2. **Payment** подмодулът е свързан с външни системи за извършване на съответната парична транзакция.

3.3.5. Описание на възможните вариации

3.3.5.1. Възможно е **Velo** да си комуникира само с **External services**, които да изпращат неговата обновена информация на **Web server**, вместо **Velo** да я изпраща пряко.

4. Архитектурна обосновка. Удовлетворение на изискванията.

Тази точка от документа е предвидена за представянето на всички функционалности, които са дефинирани в първоначалните изисквания на системата. След всяка функционалност е описан начинът, по който избраната архитектура го удовлетворява.

Удовлетворение на изискванията:

1. **Системата поддържа следните групи потребители: Наемател на велосипед, Системен администратор, Член на група по техническа**

поддръжка на велосипедите, Наблюдател/отговорник по използването на велосипедите. - Това изискване е изпълнено чрез изграждане на отделен компонент за ролите на потребителите, които поддържат системата, както и специален компонент за потребителите наематели на велосипеди. Също така VeloCity поддържа необходимата проверка и валидация на потребителите, за да показва само функционалностите, свързани с определената група, към която принадлежат. За регистрацията на потребители се грижи **AcountManager** модула, който е част от Client, а за даване на правилния достъп до системните функционалности **UserAuthorization** модула на Server. Имплементацията на функционалностите за различните групи потребители се намира в отделен модул, наречен **SystemUsersManager**.

2. **Наемателите на велосипед се регистрират през мобилното приложение, като в профила им се включват следните данни: имена, ЕГН, както и данни за връзка. (Потребителите наематели използват системата САМО през мобилно приложение!) Личните данни на потребителите трябва да са абсолютно защитени от външна намеса. Достъпни са единствено до наблюдателя на правомерното използване на велосипедите.** - Имплементацията на функционалността с регистрирането на нов потребител, както е упоменато и в предходната точка се намира в модула **Login/Register** на AccountManager в Client. Данните на всеки потребител са абсолютно защитени, като за постигане на тази цел се използва външно API, наречено Security. То ще бъде интегрирано към системата и всяка операция, в която участват потребителски данни ще бъде оторизирана от него. За да имат достъп до системата, наемателите използват мобилно приложение, чиято имплементация е в модула **MobileApp** в UserInterface, като за удобство на всички потребители са налични две версии според операционната система - за IOS и за Android.
3. **Системните администратори и наблюдателите използват системата през Уеб приложение.** - Изискването е спазено, като към модула UserInterface е добавен подмодул **WebBrowser,** в който се намира реализацията на начина, по който ще изглежда системата през очите на администратор или наблюдател.
4. **Велосипедите се намират на стоянки, разположени на предварително определени позиции в рамките на града. Батериите на велосипедите**

се зареждат по време на престоя им на стоянката. При заявка от потребителя за търсене на велосипед, приложението му показва най-близката стоянка, където има свободен велосипед с поне X% заредена батерия. X е конфигурационен параметър на системата. - При направена подобна заявка от желаещ да наеме велосипед се търсят подходящи велосипеди чрез функционалностите имплементирани в **QueriesManager** и **VeloStation** модулите. VeloStation модула получава постоянна информация от стоянките дали има оставен велосипед на определеното местоположение, както и колко е зарядът на батерията. Ако са удовлетворени условията - **X%** заредена батерия и наличен велосипед на определената стоянка, се изпраща отговор към потребителя с точно местоположение, който има възможност да наеме превозното средство или да се откаже (процесът по наемане на велосипед е изобразен по-подробно чрез структурата на процесите в т. 3.1).

5. **Потребителите може да заплащат услугата чрез кредитна карта, СМС или чрез предварително закупени талони, които съдържат уникален код. Кодът може да се въвежда ръчно или автоматично (QR-code).** - Цялата функционалност по заплащането е поместена в модулите **PaymentManager** в Server и отделния **Payment** с прилежащи подмодули трите начина за извършване на това действие - Credit/Debit, SMS, Coupons. PaymentManager обработва информацията при заплащане, но тъй като това е сложен процес се налага използването на външно за системата API - Payment, което е интегрирано към нея. Двата модула си комуникират за успешно извършване на плащане от наемателите на велосипеди.
6. **Всеки велосипед има уникален идентификационен номер в системата и е снабден с GPS устройство, както и със смарт-сензори за самодиагностика. При наличие на технически проблем по велосипеда (спукана/спаднала гума, повреда, и т.н.) да се изпраща известие до групите по техническа поддръжка, които в рамките на половин час трябва да диагностицират повредата и да вземат мерки за отстраняването ѝ. При засичане на пътен или друг инцидент с велосипеда, се изпраща автоматично сигнал до спешна помощ (112), в рамките на 1 сек след засичане на инцидента. В рамките на 5 сек се известява и наблюдателя на системата.** - Първото изискване е

реализирано с помощта на модулите **ReliabilityAndSecurityManager** във Velo и **SystemUsersManager**. В първия, или по-конкретно в неговия подмодул - DiagnosticsLogic, се намира функционалността, за диагностициране на повреди чрез използване на прилежащите към велосипеда сензори. Тъй като е възможно да се добавят или заменят/премахнат определени сензори, а това трябва да се случва лесно, без да се предизвикват промени по модули, които не са пряко свързани, е обособен отделен подмодул за тях, наречен Sensors. При откриване на технически проблем или повреда в превозното средство се изпраща сигнал до SystemUsersManager, където се обработва от правилния подмодул - TechnicalSupport. Ако техническият екип се нуждае от точно местоположение на повредения велосипед, то той го достъпва от базата данни, където постоянно се обновява локацията, благодарение на LocationUpdate модула във Velo и Location модула в Server. Второто изискване за засичане на пътен инцидент е изпълнено отново от функционалностите на модула DiagnosticsManager, който при засечена повреда изпраща сигнал до сървъра и се предава до SystemUsersManager, където чрез подмодулите в него се известяват потребителите наблюдатели на системата и спешна помощ - 112 за възникналата повреда.

7. **Максималното време T за използване на един велосипед е конфигурационен параметър на сървъра на системата. След изтичане на максималното време се изпраща съобщение на наемателя и той трябва да остави велосипеда на най-близката стоянка и да го замени с друг ако му е необходимо. -** Това изискване е изпълнено с помощта на модулите **VeloStationManagement** и **TimeManager**. TimeManager модула известява наемателя на велосипеда, че времето му за ползване е вече изтекло. С помощта на първия модул VeloStationManagement потребителите могат да разберат местоположението на стоянката, намираща се най-близо до тях, където може да бъде оставено превозното средство. VeloStationManagement получава информация от VeloStation пакета на сървъра.
8. **При излизане на велосипед от рамките на града, трябва да се сигнализира наблюдателя в рамките на 1 мин, като се изпратят данни за движението на велосипеда в последните 30 мин. При или загуба на връзка с даден велосипед, трябва да се сигнализира наблюдателя в**

рамките на 10 сек, като се изпратят данни за движението на велосипеда в последните 30 мин, заедно с най-вероятната му позиция на която се е намирал в момента на изгубване на връзката. Т.нар. най-вероятна позиция се определя със специален алгоритъм (напр. чрез изкуствен интелект). - Модулите **Location**, **SystemUsersManager** и **LocationUpdate** спомагат за изпълнението на тези изисквания. Първият проверява дали велосипедът е напуснал града или не. Ако това се е случило, се изпраща сигнал до сървъра и се предава до втория модул, където потребителите, поддържащи системата използват своите функционалности, за да известят обикновения потребител относно излизането извън града. При изгубване на връзка с велосипеда, се повтаря същата процедура, като се известяват потребителите от SystemUsersManager, които добиват представа за последните му местоположения от LocationUpdate модула и изчисляват по определен начин къде е най-вероятно да се намира.

9. **Системните администратори имат права да конфигурират системата и да следят за правилната ѝ работа. Допуска се ремонт и профилактика в интервала от 2:30 до 5:30 ч. В останалата част на деня, системата трябва да е 99,999% налична.** - Подмодулът **Admin** на **SystemUsersManager** предоставя функционалностите на администраторите да се грижат за системата и правилната ѝ работа. За да се постигне максимална наличност на системата са имплементирани някои тактики. Velo модула постоянно изпраща сигнали към сървъра и базата данни, което може да се каже, че е heartbeat метод за откриване на неизправност. При евентуална повреда по някой велосипед, то той ще спре изпращането на сигнали и ще следва да се отстранят неизправностите от техническия екип. С това е изпълнено горното изискване.
10. **Системата трябва да може да се интегрира с всички познати онлайн услуги за географски карти (Google maps, BG maps, Open Street maps и т.н.), като има възможност за бъдещо добавяне на нови карти.** - Това изискване е разработено от модула **ExternalServices**, който съдържа подмодул, осъществяващ връзка с API-та за осигуряване на географски карти, като в бъдеще могат да се добавят още външни услуги, както е описано в секция **1.2.1.**