VeloCity

курсова работа по ‘Софтуерни архитектури и разработка на софтуер’

Ангел Маринski 62469
Александър Стоилов 62382

1. Въведение

а) Обща информация за текущия документ

i. Предназначение на документа

Чрез този документ е представена софтуерната архитектура на системата за велосипеди под наем ‘VeloCity’.

ii. Описание на използваните структури на архитектурата

➔ Декомпозиция на модулите

Представя логическото разделение на системата на съответните модули и подмодули и връзките помежду им. Основните модули на системата са: Bike Manager, User Manager, Payment System, Map API, Mobile App, Database, Bike, Technical Support, Secured Collection, Web App. Чрез връзките помежду им се осъществява комуникацията между тях - най-често чрез така наречения ‘Communication API’ подмодул. User Manager контролира потребителските процеси, Mobile App и Web App са приложенията на системата (различните среди на използване). Database и Secured Collection са модули в областта на съхранение на данните.

➔ Структура на процесите

Чрез структурата на процесите онагледяваме цялостният процес на употреба на системата от страна на потребителско лице. Тя представя относително информативно къде в системата се намира съответното събитие по протичането на цялостният процес. Предназначена е за запознаване със системата, тъй като представлява значително по-високо ниво на абстракция от останалите структури.

➔ Структура на внедряването

Чрез тази диаграма се представя вече наистина по-детайлно разположението на системата върху софтуерните и хардуерните компоненти. Тъй като ‘VeloCity’ е изградена от множество различни компоненти, е от ключова важност представянето на диспозицията им.

iii. Структура на документа

1. Въведение

2. Декомпозиция на модулите
    a) Общ вид на декомпозицията на модули на системата
    b) Контекстна диаграма
    c) Подробно описание на всеки модул
    d) Описание на възможните вариации

3. Структура на внедряването
    a) Първично представяне
    b) Описание на елементите и връзките
    c) Описание на обкръжението
    d) Описание на възможните вариации

4. Структура на процесите
    a) Първично представяне
    b) Описание на елементите и връзките
    c) Описание на обкръжението
    d) Описание на възможните вариации

5. Архитектурна обосновка

b) Общи сведения за системата

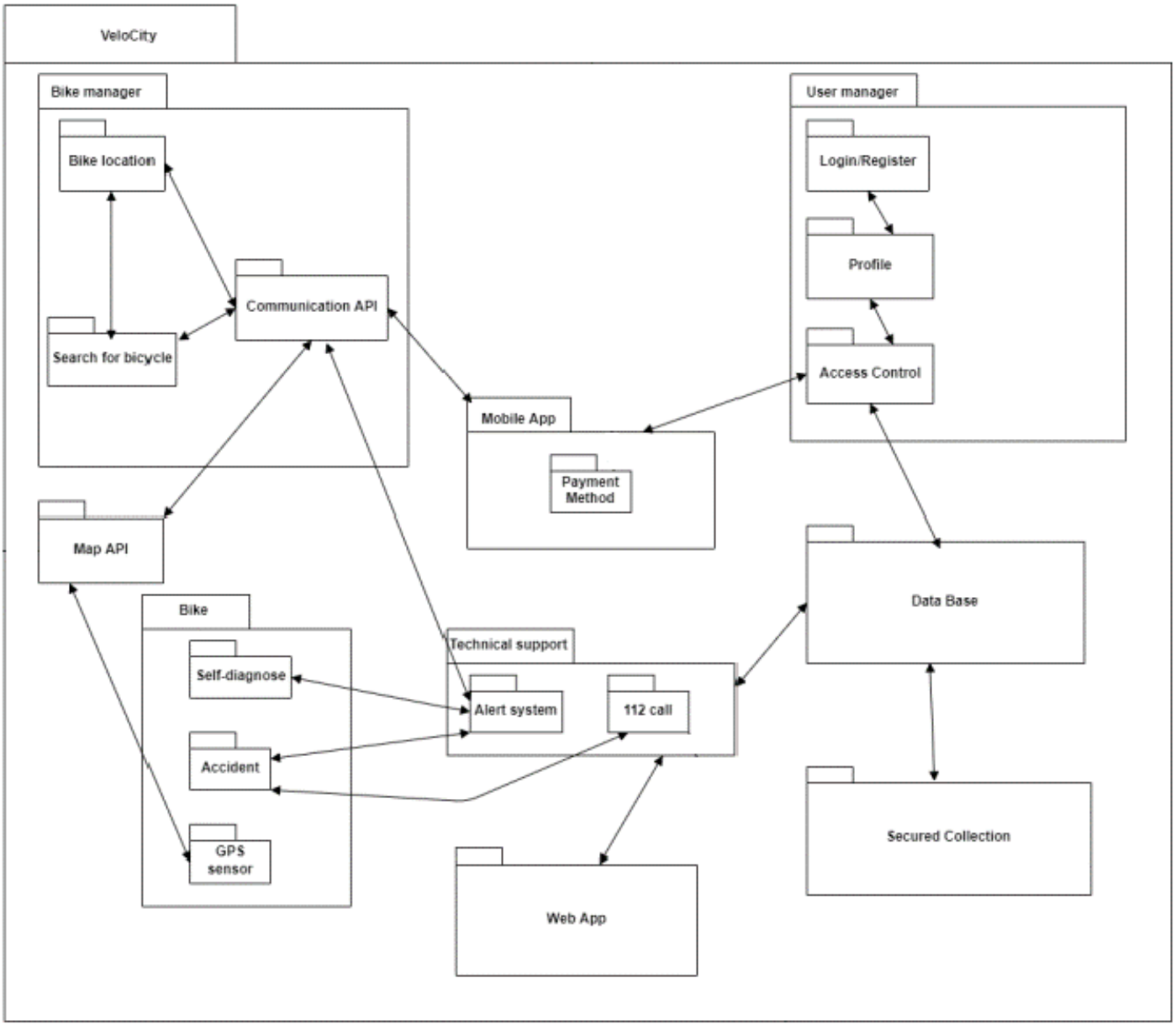
Глобалното затопляне. Проблем, за който всички сме чували, и към който непрекъснато допринасяме чрез използването на моторни превозни средства за каквито и да е маршрути и придвижвания. Решението на този проблем е едно - VeloCity. Система за отдаване на велосипеди под наем, предоставяща изцяло нов поглед върху придвижването из градската зона. Само с няколко клика, всеки може да наеме велосипед от която и да е точка на града си и да го остави където му е удобно. VeloCity е истински революционен подход за справянето с проблемите, свързани с вредата към околната среда.

c) Терминологичен речник

2. Декомпозиция на модулите

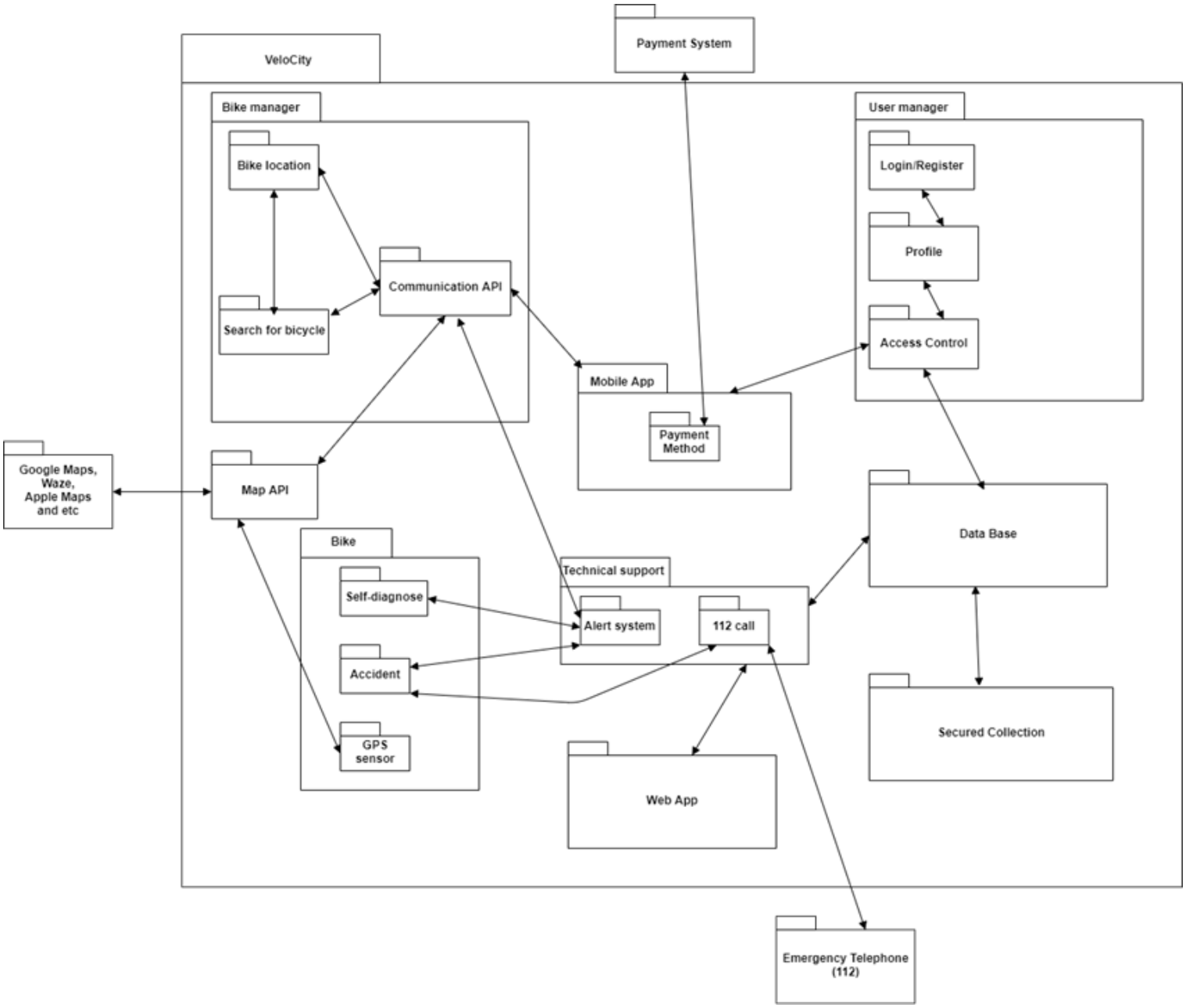
a) Общ вид на декомпозиция на модули на системата

Системата разполага със собствените модули Bike Manager, User Manager, Map API, Mobile App, Database, Bike, Technical Support, Secured Collection, Web App. Bike Manager се грижи за управлението на велосипеда и връзката на Mobile App с Bike. User Manager контролира обсега на потребителското лице, изпълнявайки връзката между Mobile App и клиентската функционалност. Map API представлява подсистема по интегриране на географски карти към



системата. Database и Secured Collection се грижат за качественото съхранение на данни. Web App е порталът за употреба на Technical Support членовете.

b) Контекстна диаграма



Mobile App и Web App са интерфейсите. От изискванията извличаме, че съществува, както мобилна, за наематели, така и уеб, за членове на техническата поддръжка, наблюдатели и системни администратори, версия на приложението.(изискване №1)

Не е добра идея Mobile App и Web App да комуникират директно с базата данни, затова създадохме сървиса User manager, който да се занимава с различните потребители и да ни дава информация за ролите на потребителите. Модулът удовлетворява изискването до приложението да се допускат само регистрирани потребители, регистрацията се извършва с множество лични данни за връзка(изискване №2), след като са регистрирани и се логнат в системата, модула им позволява непряк достъп до базата данни и връзка с мобилното приложение. По този начин гарантираме, че потребителските данни са добре скрити.(изискване №3)

Payment Method – осигурява връзката с външни разплащателни системи. Този модул се грижи за правилно обработване на заявките за плащане, които могат да бъдат, чрез ваучер, СМС, карта и други.

Bike manager следи за състоянието на велосипедите(локация,батерия). Посредством подмодулите Bike location и Search for bicycle следим местоположението на велосипедите и чрез подмодула Communication API се изпраща информация до мобилното приложение за най-близкото колело и неговото състояние.

Bike е модулът, който отговаря велосипеда самосиндикално да се диагностицира и събраната информация да се разпраща на заинтересованите лица, в нашия случай техническия отбор(Technical support), също така при наличието на инцидент модулът оповестява 112 и техническия отбор за проблема, автоматично. Също така в модула се съдържа подмодулът Gps sensor, който е отговорен да наблюдава движението на велосипедите и ни подсигурява, че те няма да напуснат предварително определените граници за ползване.(изисквания №5 №6)

Map API ни дава възможността за интеграция на всякакъв вид външни карти като Google Maps, Waze и др.(изискване №7)

Database е модулът, който застава като защитна стена към прекия достъп до нашата база данни, имаме нужда от този модул, за да осигурим по- голяма сигурност на личните данни и на данните на системата.

c) Подробно описание на всеки модул

User Manager

➔ Предназначение

Контолиране на функционалността на потребителите на системата.

➔ Отговорности

Правилно достъпване до базата данни.
Връзка до ‘задкулисните’ операции в системата.
Изпълнение на потребителските команди от фронтенда.

➔ Интерфейси

register(string name, string surname, string UCN, string phone_number, string email)

Входни данни:

име, фамилия, единен граждански номер, телефонен номер, по желание имейл за обратна връзка;

Резултат:

прозорец за въвеждане на уникално потребителско име за системата и парола-ключ, регистриране на потребителя в системата;

Грешки и заключения:

подходящо обратно съобщение при невалидни данни;

Зависимости от други елементи:
комуникация с Database;

login(string username, string password)
Входни данни:
уникално потребителско име, парола-ключ;

Изходни данни:
вход в системата от страна на потребителя;

Грешки и заключения:
подходящо обратно съобщение при невалидни данни;

Зависимости от други елементи:
комуникация с Database;

inspect_profile(<user_info>)
Входни данни:
определени въведени данни, част от възможните входни данни при регистрация на потребител;

Изходни данни:
изходна справка с всички потребители, отговарящи на съответните входни данни;

Грешки и заключения:
подходящо обратно съобщение при входни данни, неотговарящи на нито един потребител в системата, методът е предвиден за екипът на системата;

Зависимости от други елементи:
комуникация с Database;

update_profile(<user_info>, <updated_info>)
Входни данни:
определени въведени данни, част от възможните входни данни при регистрация на потребител, и новите данни, които потребителя желае да актуализира;

Резултат:
актуализирани съответните данни, които потребителя е пожелал да промени;

Грешки и заключения:
подходящо обратно съобщение при невалидни данни;

Зависимости от други елементи:

комуникация с Database, извършване на security check-ове, целящи да проверят легитимността на актуализацията (дали реално бива извършена от страна на потребителя, а не някакъв тип malware);

Database

➔ Предназначение

Осигуряване на комуникационна връзка между User Manager и Secured Collection.

➔ Отговорности

Изолиране на същинската база данни.

Контролиране на заявките (достъпа) до хранилището от страна на останалите модули в системата.

Качествено разпределение на заявките между компонентите на системата по посока на хранилището.

➔ Интерфейси

search_bike(<coordinates>)

Входни данни:

координати на потребителя, подал заявката;

Изходни данни:

най-подходящият велосипед спрямо входните данни;

Грешки и заключения:

подаване на подходяща информация към Bike Manager при нередности, свързани с намирането на подходящ велосипед;

Зависимости от други елементи:

комуникация със Secured Collection и Bike Manager;

register(<user>)

Входни данни:

съответните данни, подадени от регистриращ се потребител;

Резултат:

регистриране на потребителя;

Грешки и заключения:

подаване на подходящ exit status към User Manager при проблеми, свързани с регистрацията на потребителя;

Зависимости от други елементи:

комуникация със Secured Collection и User Manager;

login(<user>)

Входни данни:

съответните данни, подадени от влизащ в системата потребител;

Резултат:

влизане в системата от страна на потребителя;

Грешки и заключения:

подаване на подходящ exit status към User Manager при проблеми, свързани с входа в системата на потребителя;

Зависимости от други елементи:

комуникация със Secured Collection и User Manager;

inspect_profile(<user>)

Входни данни:

данни, целящи да филтрират определен/-и потребител/-и;

Изходни данни:

справка от потребителите, отговарящи на подадените входни данни;

Грешки и заключения:

подаване на подходящ exit status към User Manager при подадени данни, несъответстващи на лице в системата;

Зависимости от други елементи:

комуникация със Secured Collection и User Manager;

update_profile(<user>)

Входни данни:

данни, подадени от потребителя, целящ да актуализира определена информация;

Резултат:

променени данни на потребителя;

Грешки и заключения:

подаване на подходящ exit status към User Manager при грешка в изпълнението на актуализацията;

Зависимости от други елементи:

комуникация със Secured Collection и User Manager;

inspect_bike(<bike>)

Входни данни:

данни за велосипед;

Изходни данни:

информация от инспекцията на велосипеда;

Грешки и заключения:
съобщение при неспособност за неосъществяване на връзка с устройствата/сензорите на велосипеда, изпращано към Bike Manager;

Зависимости от други елементи:
комуникация със Secured Collection и Bike Manager;

update_bike(<bike>)

Входни данни:
данни за велосипед, които предстои да бъдат актуализирани;

Резултат:
актуализация на посочените входни данни;

Грешки и заключения:
подходящо съобщение при невъзможност да бъдат актуализирани съответните данни, посочени като входни, изпращано към Bike Manager;

Зависимости от други елементи:
комуникация със Secured Collection и Bike Manager;

Bike Manager

➔ Предназначение
Осъществява контрол върху процесите, свързани с управлението на велосипед.
Обработва резултати от- и изпраща към модула Bike изходни стойности/справки.

➔ Отговорности
Наличие на лесна комуникация както на ниво подмодули, така и на Mobile App с Map API, Technical Support и Bike.

➔ Интерфейси
search_bike(<coordinates>)

Входни данни:
координати на потребителя, подал заявката;

Изходни данни:
най-подходящият велосипед спрямо входните данни;

Грешки и заключения:
подаване на подходящ exit status към User Manager при нередности, свързани с намирането на подходящ велосипед;

Зависимости от други елементи:

комуникира с метода search_bike от Database, който работи с необходимите данни от Secured Collection;

inspect_bike(<bike>)

Входни данни:

данни за велосипед;

Изходни данни:

справка за велосипеда по подадените данни;

Грешки и заключения:

съобщение при неотговарящ на данните (неприсъстващ) велосипед в системата;

Зависимости от други елементи:

комуникация с метода inspect_bike от Database, който работи с необходимите данни от Secured Collection;

update_bike(<bike>)

Входни данни:

данни за велосипед, които предстои да бъдат актуализирани;

Изходни данни:

актуализация на посочените входни данни;

Грешки и заключения:

съобщение при невъзможност за актуализация;

Зависимости от други елементи:

комуникира с метода update_bike на Database, който работи с необходимите данни от Secured Collection;

report_accident(<bike>, <user>, <information>)

Входни данни:

данни за велосипед, данни за потребител, информация за инцидента;

Резултат:

изпращане на сигнал към Emergency Telephone;

report_bike_failure(<bike>)

Входни данни:

данни за велосипед;

Резултат:

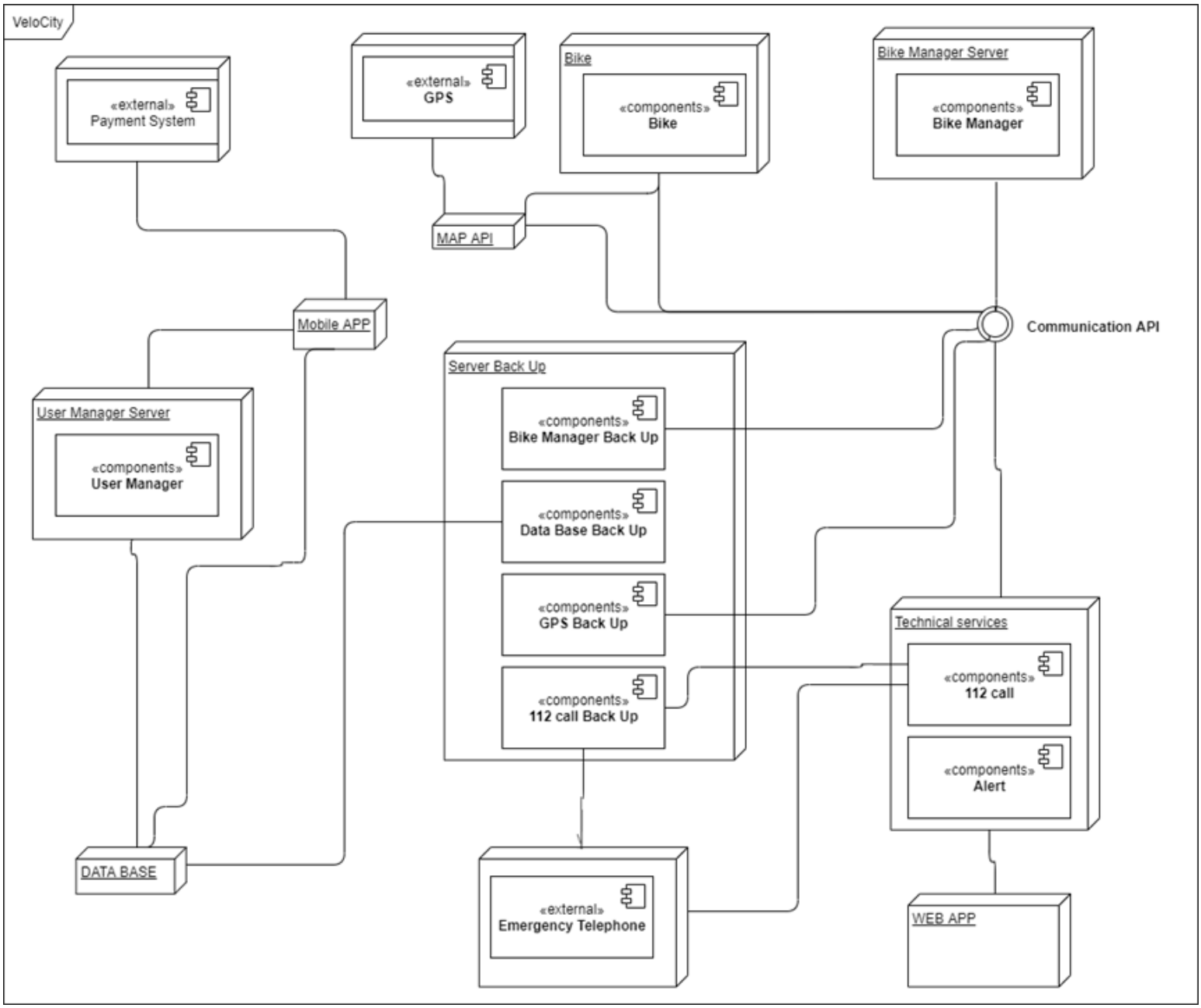
изпращане на сигнал към Database за проблем с велосипед;

Зависимости от други елементи:

комуникация с Database, който работи с необходимите данни от Secured Collection;

3. Структура на внедряването

а) Първично представяне



b) Описание на елементите и връзките

Тъй като системата Velocity е изградена от множество различни компоненти - уеб и мобилно приложение, велосипеди, стоянки, множество сървъри за обработка на голямо количество заявки и бази данни, които съхраняват данните за потребителите и за велосипедите,

за нас е от голяма необходимост тези компоненти да бъдат ясно показани в някоя структура на разположението.

От структурите на разположението избрахме тази на внедряването (deployment), представяща разположението на отделните модули върху хардуера. Изборът ни се основава на факта, че голяма част от изискванията водят до това, че един модул трябва да бъде внедрен върху няколко машини или да съществуват няколко инстанции на даден сървър. А тези тактики за производителност и наличност не могат да се представят толкова добре чрез структура на разгръщането или файлова структура.

Диаграмата представя сървърите, на които по-голямата част от модулите ще бъдат внедрени. Според изискванията системата трябва да е 99,999% налична. Следователно е добре да създадем пасивен излишък (passive redundancy) в лицето на нашия Server Back Up.

Понеже системата трябва да бъде устойчива на пикови натоварвания, както и навременно да се осведомяват съответните лица при наличието на проблем, инцидент или излизане от рамките на града, решихме да приложим тактиката за разнородност в копията на сървърите, с което увеличаваме надеждността на системата.

Communication API осъществява връзката между микросървисите, играе роля на gateway.

Mobile APP представлява всички мобилни телефони на наемателите, на които има изтеглено приложението, а WEB APP – компютрите, чрез които всички членове на екипа използват системата.

Bike Manager ще е най- натоварения микросървис, защото към него се изпращат множество заявки и запитвания за велосипеди, заради това е от изключителна важност да имаме втори сървър, при възникването на проблем.

Data Base е микросървис, който възпира директната връзка с базата ни от данни, но това не го прави по- маловажен, не можем да си позволим да се срине и да не се възстанови възможно най-бързо, защото така верификацията на профилите няма да бъде напълно адекватна и може неприятел да се добере до достъп до системата ни.

c) Описание на обкръжението

Payment System е external система, осигуряваща изпълнение на съответните операции по VeloCity.

GPS съответства на външната/-те платформа/-и за географски карти, интегрирана/-и към системата VeloCity с целите, посочени в *(т.5) Архитектурна Обосновка*.

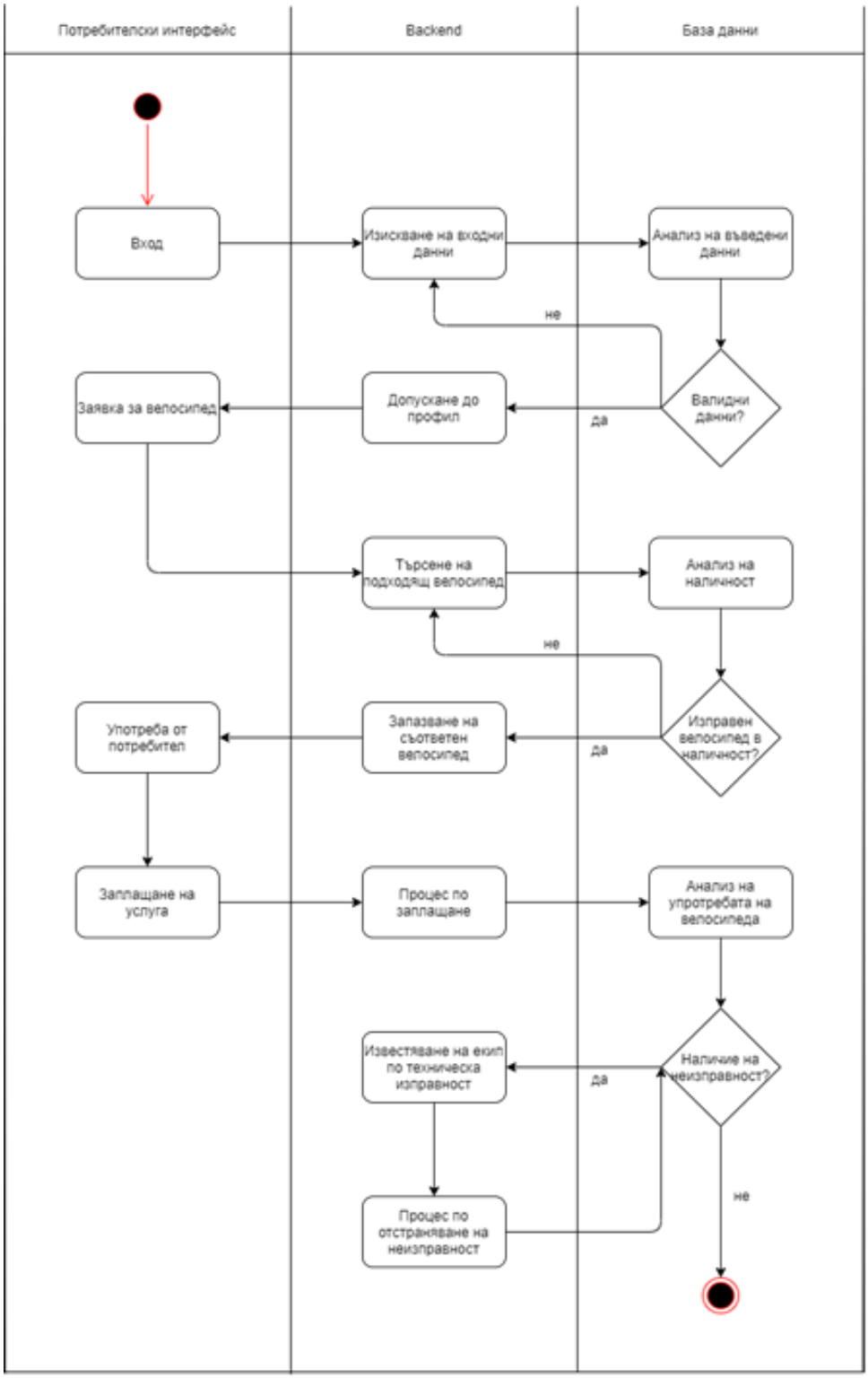
Emergency Telephone е националната линия за спешни случаи, към която се изпраща сигнал от метода report_accident на Bike Manager.

d) Описание на възможните вариации

Би могла да бъде съставена собствена за системата подсистема за географски карти, но това би отнело значително количество ресурси. По аналогичен начин може да се разгледа и Payment System.

4. Структура на процесите

a) Първично представяне



b) Описание на елементите и връзките

Прикачената по-горе структура на процесите има за цел да предостави информация по цялостния процес на употреба, разглеждан от страна на потребителско лице на системата.

Процесите са обособени в три съставни компонента - потребителски интерфейс, backend и база данни. Разглеждайки диаграмата на декомпозицията на модулите (*т.2*) можем да отбележим, че потребителският интерфейс представлява модулът User Manager - този, с предназначение контрол на функционалността на потребителите и отговарящ за правилното достъпване до базата данни от собствените си компоненти. Модулът Databases и същинското хранилище Secured Collection обособяват частта, която на структурата е изобразена като “База данни”. Модулите Payment System, Bike Manager, Map API, Mobile App, Bike, Technical Support, Web App дефинират backend-а на системата и отговарят на втората (средната) част от структурата на процесите.

Логиката на структурата е следната: Потребител осъществява заявка за вход в системата. Получава изискване на входни данни от нейна страна, които следва да попълни правилно. След анализиране на входните данни и верифицирането им, при коректни такива потребителят бива допуснат до системата. Той осъществява заявка за велосипед. Системата се ангажира с това да намери подходящ за потребителя велосипед спрямо местоположението му, характеристиките и състоянието на велосипедите в околността му. При намиране на подходящ велосипед, той бива разпределен за потребителя, като съответния получава информация за местоположението на велосипеда. След употреба от страна на потребителя на превозното средство, той инициира процес по заплащане на услугата (еднократно или автоматично от универсално разплащателно средство, в зависимост от подбраните опции на лицето). Сървърът обработва процеса по заплащане, след което прави анализ на техническата изправност на велосипеда и в случай на проблематични характеристики - причините за пораждането им. При наличието на неизправност бива известен екипът по техническа поддръжка, който има за цел да отстрани повредата. След това процесът приключва и с това завършва структурата на процесите, обхващата “полезрението” на потребителската употреба на велосипед на VeloCity.

c) Описание на обкръжението

При процеса на търсене на подходящ велосипед се използва външната система за географски карти, интегрираната към VeloCity (*вж. т.3*).

При разплащателния процес се използва външната система за финансови услуги (*вж. т.3*).

d) Описание на възможните вариации

В случай на инцидент всяка операция по плана структурата на процесите мигновено се стопира, докато бъде съобщено на необходимите власти, с цел възможно по-бързото изпращане на сигнал.

5. Архитуктурна обосновка

● *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*

Ролите в системата имат ключова важност за формирането на архитектурните структури в началото на процеса на разработка. Те оформят

скелета на платформата в смисъла на идеята за функционалността и използваемостта от крайните потребители. Системата разполага с Mobile App, предвиден за потребителите, и Web App за екипа.

● *Наемателите на велосипед се регистрират през мобилното приложение, като в профила им се включват следните лични данни: имена, ЕГН, както и данни за връзка.*

● *Личните данни на потребителите трябва да са абсолютно защитени от външна намеса. Достъпни са единствено до наблюдателя на правомерното използване на велосипедите.*

Аспектите за възможност за регистрация на потребител чрез определени лични данни и спазването на законите за тяхната защита от външния свят и/или други потребители формират цели, изискващи определени изначални решения при структурирането на потоците от данни и правата за достъп между компоненти с цел недопускането на грешки в тази област. Модулът, отговарящ на посоченият архитектурен драйвер, е Data Collection. В допълнение, чрез структурата на внедряването е демонстриран регламентирания достъп до съхранението на данните.

● *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.*

Аспектите, покриващи надеждността в навременното алармиране за технически проблеми и/или неизправности, както и пътни инциденти, са жизненоважни и биха могли да бъдат имплементирани в системата по аналогичен начин, вземайки предвид общите характеристики на необходимите елементи за реализирането им.

● *Системните администратори имат права да конфигурират системата и да следят за правилната ѝ работа.*

Абсолютна необходимост е системните администратори да имат възможност да извършват работни процеси върху системата с цел правилното ѝ функциониране. Това е осигурено чрез наличието на връзка между Technical Support и останалата част от backend-а на системата.

● *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.*

Възможността да се интегрират съществуващи в реалния свят услуги за географски карти ще лишат разработващия екип от необходимостта да се реализира собствена такава, ще улесни крайния потребител в привикването му към системата, и ще доведе нови такива поради лесния преход към системата от досега използваните от тях приложения. Свързването към външна система за географски карти е представено чрез модула Payment System.

● *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.*

Основополагащо за привличането на крайни потребители е и наличността на системата в рамките на деня, извън ясно определените за нея часове за профилактика, които следва да бъдат съобразени с използваемостта на системата по различните времеви точки спрямо денонощието и годишните периоди като празнични дни, дни със засилена употреба на превозни средства в градски условия и други.