RidePal

КУРСОВА РАБОТА
ПО СОФТУЕРНИ АРХИТЕКТУРИ
И РАЗРАБОТКА НА СОФТУЕР

Изготвили:
Женя Георгиева 61777
Мария Димитрова 61818

Софийски университет „Св. Климент Охридски“,
Софтуерно инженерство 2014/2018

Април - Юни 2016

Съдържание

1. Въведение

1.1 Организация на текущия документ

1.1.1 Предназначение на документа

Целта на документа е да представи софтуерната архитектура на системата RidePal.

1.1.2 Списък на структурите

Пояснение на структурите: какви са елементите, връзките между тях и какъв аспект на системата показват.

Декомпозиция на модулите

В същността си декомпозицията на модули обуславя в голяма степен възможността за лесна промяна като обособява логически свързани функционалности на едно място. Тя показва разделението на отделни модули (логически обособени единици) в рамките на една система. Елементите са модулите, а връзките между модулите са от вида „Х е под-модул на У“.
Системата Ridepal е разделена на модулите RPServer, RPSite, RPApp, Database и Platform Adapter. В RPServer-а се съхранява бизнес логиката на системата и се свързва с Database.
Модулът RPSite е уеб сайт, чрез който потребителите използват системата(регистрация, управление на профила, регистрация на МПС). RPApp е мобилно и десктоп приложение, чрез което крайните потребители следят МПС-то с което ще пътуват и общуват с другите участници в споделеното пътуване. Platform Adapter осигурява платформена независимост.

Допълнителни структури

1) **Употреба на модули :**
   Елементите са модули, а връзките между тях са от вида „Х използва У“. В тази структура се разглеждат връзките между модулите, обособени в декомпозицията.
2) **Структура на внедряването :**
   Елементите са процеси, хардуерни устройства и комуникационни канали. Връзките са например „внедрен върху“ или „мигрира върху“ . Самата структура показва как софтуерът се разполага върху хардуера и комуникационното обурудване. Структурата показва връзките между софтуерните елементи със средата, в която ще бъде използвана системата. Елементите са приложения, сървъри и външни системи.
3) **Структура на процесите:**
   Елементите са процеси, изпълнявани в системата(компоненти) и комуникационни, синхронизационни или блокираци операции между тях (конектори), а връзките между тях показват как компонентите и конекторите се отнасят помежду си. Разглеждат се някои от процесите в системата.

1.1.3 Структура на документа

• Въведение – заема секция 1(текущата секция) от настоящия документ. Въведението включва предназначението на документа, списък на структурите, структура на документа, общи сведения на системата и терминологичен речник

• Декомпозиция на модулите – заема секция 2. от текущия документ. Описва в детайли декомпозицията на системата RidePal и ясно описва всеки от нейните модули – RPServer, RPSite, RPApp, Database, Platform Adapter

• Допълнителни структури – заема секция 3. на настоящия документ и разглежда следните структури:

о Употреба на модули – Разглеждат се връзките между модулите, обособени в Декомпозицията. Това са връзки от “тип модул А използва модул Б”.
о Структура на внедряването – показва връзките между софтуерните елементи със средата, в която ще бъде използвана системата. Елементите са приложения, сървъри и външни системи.
о Структура на процесите – Разглеждат се някои от по-основните процеси в системата като регистрация, влизане и излизане в системата, реализирани в главния сървър RPServer.

• Архитектурна обосновка – секция 4. от текущия документ; обосновка на това как архитектурата на системата удовлетворява архитектурните драйвери, описано за всеки поотделно.
• Допълнителна информация - секция 5. от текущия документ, насоки за бъдещо развитие на системата.

1.2 Общи сведения за системата

RidePal е онлайн система за споделяне на транспорт по определен маршрут. На практика Ridepal представлява вид социална мрежа за споделяне между служителите в рамките на дадена организация за транспорт с лични автомобили до и от работа. Допълнителни плюсове на системата са оптимизирането на трафика, намаляването на вредните емисии и задръствания и не на последно място създаването на нови познанства. Потребителите, използващи системата, трябва да си създадат акаунт като се регистрират в системата. Тези от тях, които притежават МПС и искат да участват в споделено пътуване с него имат възможността да го регистрират в системата, за да го свържат с потребителския си профил. Всеки потребител на системата може да създава маршрут за споделено пътуване и да търси такъв по начална и крайна точка. Всеки пътник в определеното пътуване е способен да следи къде се намира автомобила по време на пътуването. Потребителите комуникират помежду си посредством чат, който е част от системата. Плюс на системата е, че може да се свързва с различни услуги за карти, както и че работи върху различни операционни системи.

1.3 Разширен терминологичен речник

1.3.1 Списък на софтуерните елементи

Номерът пред името на елемента показва секцията, в която той се намира.

2.2 **Database**

2.3. **RPServer**

2.3.3.1. **Database Manager**

2.3.3.2. **Car Manager**

- Register car
- Connect to user

2.3.3.3. **Travel Manager**

- Create route
  - Set starting point
  - Set final point
  - Set intermediate point
- Search route

2.3.3.4. **Account Manager**

- Register
- Login
- Logout

2.3.3.5. **Integration with maps**

2.4. **RPSite**

2.4.3.1 **Account Manager**

- Register
  - User
  - Car
- Login
- Logout

2.4.3.2. **Profile Manager**

- **User**
  - Create route
  - Search route
  - Pick route
    - Comment
- **Car**
- Connect to user

2.5. **RPApp**

2.5.3.1. **Account Manager**

- Login
- Logout

2.5.3.2. **Chat Manager**

- Search user
- Send message
- Receive message

2.5.3.3. **Travel Manager**

- See location

Time to pick up
Remaining time

2.6. **Platform Adapter**

2.6.2.1. **Desktop OS**

Windows
Windows XP,
Windows Vista
Windows 7/8/10
Linux
Ubuntu, Debian
MacOS X

2.6.2.2. **Mobile OS**

Android
iOS
Windows 7/8

1.3.2 Всички други специфични термини

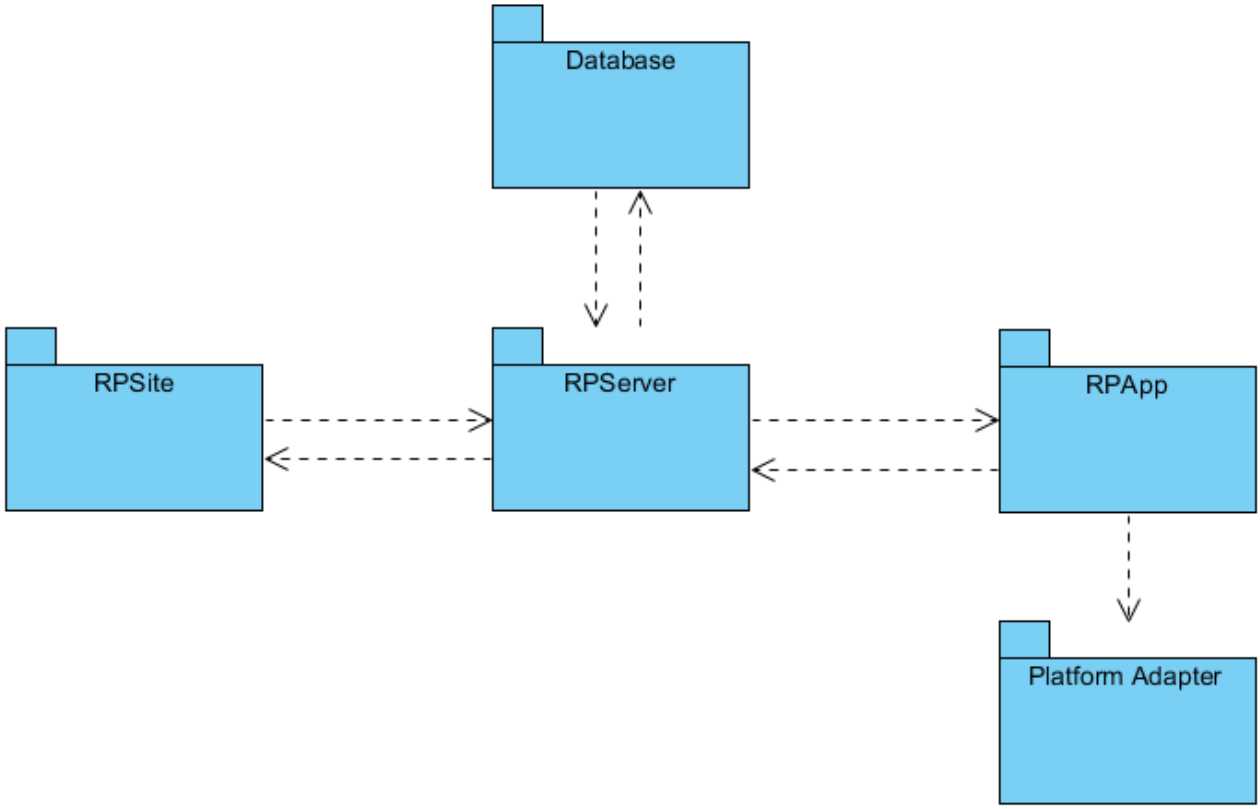
- **Адаптер** - инструмент, който прави възможна работата на два несъвместими интерфейса
- **Архитектурни драйвери** – най-важните изисквания, които определят софтуерната архитектура
- **Архитектурен стил** – дефинира група от системи, от гледна точка на модел на структурната организация
- **База от данни** - представлява колекция от логически свързани данни в конкретна предметна област, които са структурирани по определен начин. В първоначалния смисъл на понятието, използван в компютърната индустрия, базата от данни се състои от записи, подредени систематично, 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 се използват от настоящата система
- **Десктоп приложение** – приложение, което е направено за компютър и/или лаптоп
- **Интерфейс** – споделена граница, чрез която два отделни компонента на компютърната система си обменят информация
- **Клиент** – част от компютърна или софтуерна система, която достъпва услуга предоставена от сървър
- **Компонент** – изчислителна единица, която има специална функционалност, която е достъпна чрез добре дефинирани интерфейси
- **Комуникационен канал** – канал за комуникация, канал за предаване на сигнали с информация между изпращач и получател
- **Конектор** – представя механизма на комуникацията(протокола) между компонентите
- **Мобилно приложение** – приложение, кооето е направено за мобилни устройства
- **Модул** – логически обособена софтуерна единица
- **Операционна система** (**ОС**) - основна част от компютърния системен софтуер, която управлява и координира ресурсите на хардуера и софтуера и обслужва изпълняваните компютърни програми. Приложният софтуер обикновено има нужда от ОС, за да работи.
- **Платформена независимост** – възможност за работа под различни видове операционни системи
- **Потребител** – човек, който използва компютърна или мрежова услуга
- **Процес** – съвкупност от стъпки, която изгражда логическото действие и стига определена цел
- **Приложение** – софтуер, чието предназначение е да помогне на потребителя да извърши определена задача
- **Софтуерна архитектура** – съвкупност от структури, показващи различните софтуерни елементи на системата, външно видимите им свойства и връзките между тях

- **Структура** - съвкупност от софтуерни елементи, техните външно видими свойства и връзките между тях
- **Сървър** – термин, който има две тясно свързани значения:
  - Сървърът е компютърна програма, която предоставя услуги на други програми, наречени клиентски софтуер (Client).
  - Сървърът е също така компютър, стартиращ сървърен софтуер и предоставящ една или повече услуги (като например хост) на други компютри в същата мрежа.
- **GPS** (**Global Positioning System**) –спътникова радионавигационна система за определяне на положението, скоростта и времето с точност до 1 наносекунда във всяка точка на земното кълбо и около земна орбита в реално време.
- **RPApp** (RidePal Application)
- **RPServer** (RidePal Server)
- **RPSite** (RidePal Site)
- **WebServer** – сървър, който приема и връща отговор на заявки, направени от клиент.

2 Декомпозиция на модулите

2.1 Общ вид на Декомпозицията на модули за системата

Архитектурният стил, който сме избрали за най-подходящ за нашата система – RidePal, е клиент-сървър. Това може да се забележи и на посочената package диаграма. Подходяща е заради централизацията на данни, която предоставя и лесното имплементиране на back-up на системата, което подобрява Performance – да бъде устойчива в пикови часове(един от архитектурните драйвери). Следва подробно описание на всеки от модулите на системата:



2.2 Database

2.2.1 Предназначение на модула

Модулът Database представлява базата от данни, в която ще се пази информацията за различните потребители, МПС-та и споделени пътувания.

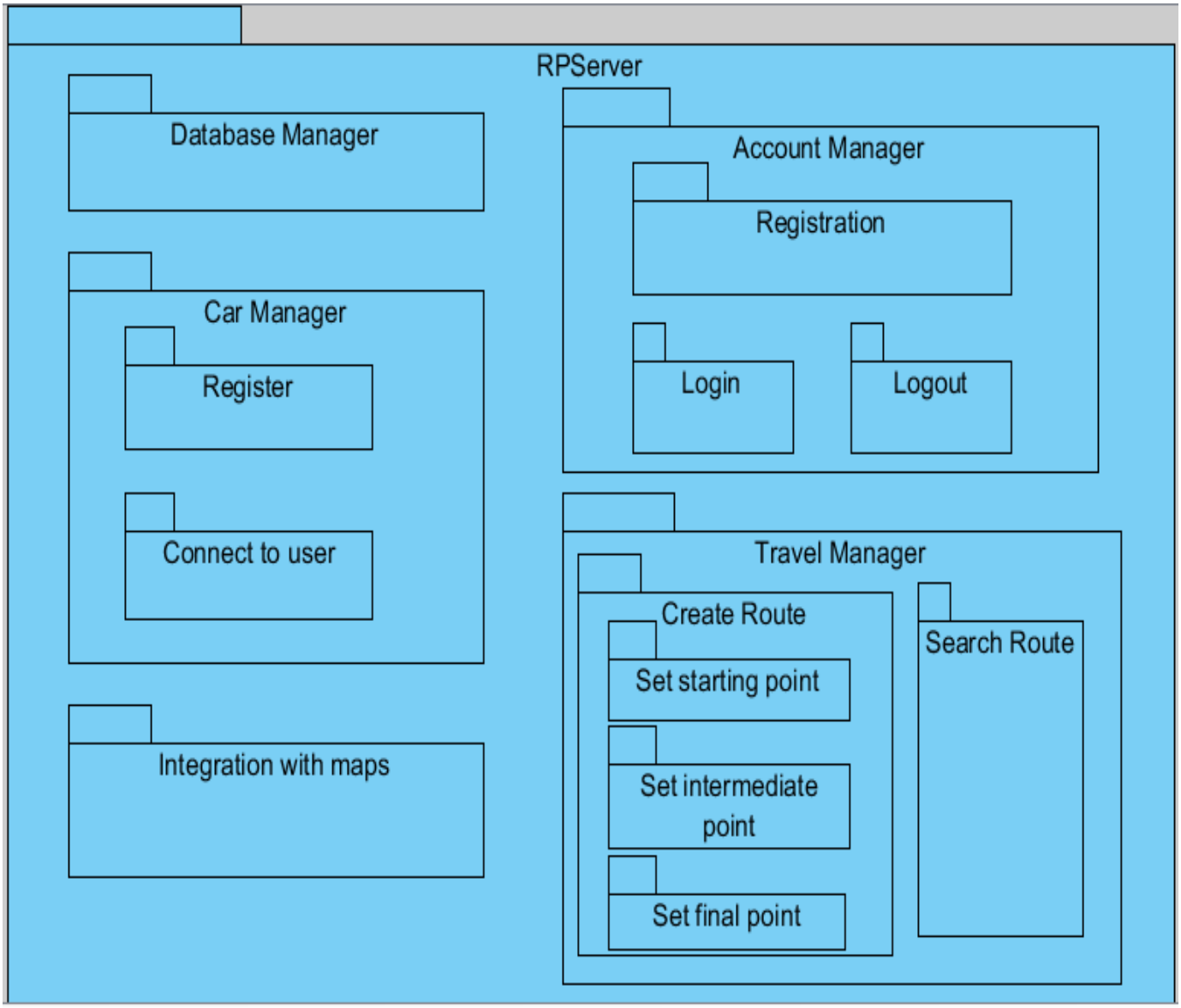
2.2.2 Основни отговорности в системата

Задачата е да съхранява данните от RPSite и RPApp, които ще й бъдат предоставени от RPServer. Те трябва да се пазят в реално време по подходящ начин, така че да бъдат консистентни и атомарни.

2.2.3 Зависимости от други елементи

Модулът зависи от всички други модули в системата.

2.3 RPSERVER



2.3.1 Предназначение на модула

В RPServer (RidePalServer) се съхранява основната бизнес логика на системата. Този модул играе и ролята на свързващо звено между базата от данни и модулите, които го използват.

2.3.2 Основни отговорности в системата

Основната отговорност на RPServer е да приема заявки от потребителските приложения, да ги обработва и да връща на съответното приложение подходящ отговор. Също така при необходимост да се обръща към Базата от данни и да взима или записва информация в нея.

2.3.3 Описание на интерфейсите на модулите

2.3.3.1 **Database Manager** – отговаря за връзката на останалите модули от RPServer с базата от данни.

2.3.3.2 **Travel Manager** – отговаря за действията, свързани със създаване на маршрут и търсене на маршрут

2.3.3.2.1

Search route – търси за определен маршрут в базата от данни и връща информация за него.

public Route[] GetByStartingPoint(string startingPoint) - търси в базата от данни маршрути,чиято начална точка е зададената

public Route[] GetByFinalPoint(string finalPoint) - търси в базата от данни маршрути, чиято крайна точка е зададената

public Route[] GetByStartingAndFinalPoint(string startingPoint, string finalPoint) – търси в базата от данни маршрути, чиито начална и крайна точка съвпадат със зададените

public Route[] GetAll() - търси всички маршрути

2.3.3.2.2 **Create Route** – създава маршрут по дадена информация:

public SetPrice(decimal price, string route) – задава на пътуването оказаната цена

public SetNumberOfPeople(int numberOfPeople, string route) – задаване на брой хора, които могат да се присъединят към пътуването

public SetDateAndHour(int year, int month, int day, string hour) – задаване на дата и час на пътуването

public Route[] SetStartingPoint(string startingPoint) – задаване на начална точка на маршрута

public Route[] SetIntermediatePoint(string intermediatePoint) – задаване на междинна точка на маршрута

public Route[] SetFinalPoint(string finalPoint)- задаване на крайна точка на маршрута

2.3.3.3 **Car Manager -** регистриране на МПС и вкарването на информацията в базата от данни, както и асоциирането на МПС-то с потребителския профил

Register – регистриране на МПС за участване с него в споделеното пътуване

public Car[] SetType(string carType) – задаване на тип на превозното средство и запис в базата от данни

public Car[] SetModel (string carModel) – задаване на модел на превозното средство и запис в базата от данни

public Car[] SetBrand (string carBrand) – задаване на марка на превозното средство и запис в базата от данни

public Car[] SetImage (byte[] carImage) – задаване на снимка на превозното средство и запис в базата от данни

Connect to user – профилът на потребителя се свързва с МПС-то, което е регистрирал и се записва в базата от данни

public ConnectUserToCar(string userid, string carid) - свързва потребителя с МПС-то, което е регистрирал

2.3.3.4 **Account Manager**

2.3.3.4.1 **Register** - проверява дали дадения потребител не се е регистрирал вече (зает email или потребителско име) и дали въведените данни са валидни(достатъчно дълго име, парола, дали има въведени задължително име, телефон и снимка). Ако потребителят не се е регистрирал и въведените данни са валидни запазва данните в базата от данни и връща положителен отговор. Ако потребителят се е регистрирал или има грешка в данните връща съобщение с причина за грешката.

public Register (string username, string password, string email, string phonenumber, byte[] image) – регистрира потребителя и вкарва данните му в базата от данни

private boolean ValidateUsername (string username) - проверява дали е въведено валидно потребителско име

private boolean ValidatePassword (string password) – проверява дали е въведена валидна парола

private boolean ValidateEmail (string email) – проверява дали е въведен валиден email

private boolean ValidatImage(byte[] image) – проверява за валидна снимка

private boolean ValidatePhoneNumber(string phonenumber) – проверява дали е въведен валиден телефонен номер

private boolean CheckIfUsernameAvailable (string username) – проверява дали въведеното потребителско име е свободно

private boolean CheckIfEmailAvailable (string email) – проверява дали въведеният email адрес е сободен

2.3.3.4.2 **Login** - проверява дали данните са валидни (съществуващ потребител, съответстваща парола). Ако данните са валидни логва потребителя, а ако не са валидни връща на потребителя съобщение за грешка.

public ResultLogin (string username, string password) – логва потребителя, ако данните са валидни или връща съобщение за грешка при невалидни данни

private boolean CheckIfUsernamePasswordMatch(string username, string password) – проверява дали потребителското име и паролата съответстват

private boolean ValidateUsername (string username) – проверява дали е въведено валидно потребителско име

private boolean ValidatePassword (string password) – проверява дали е въведена валидна парола

2.3.3.4.3. **Logout** – отнема правата за достъп на потребителя

public ResultLogout ()

2.3.3.5 **Integration with maps –** позволява интеграция с различни онлайн услуги за карти

2.3.4 Други

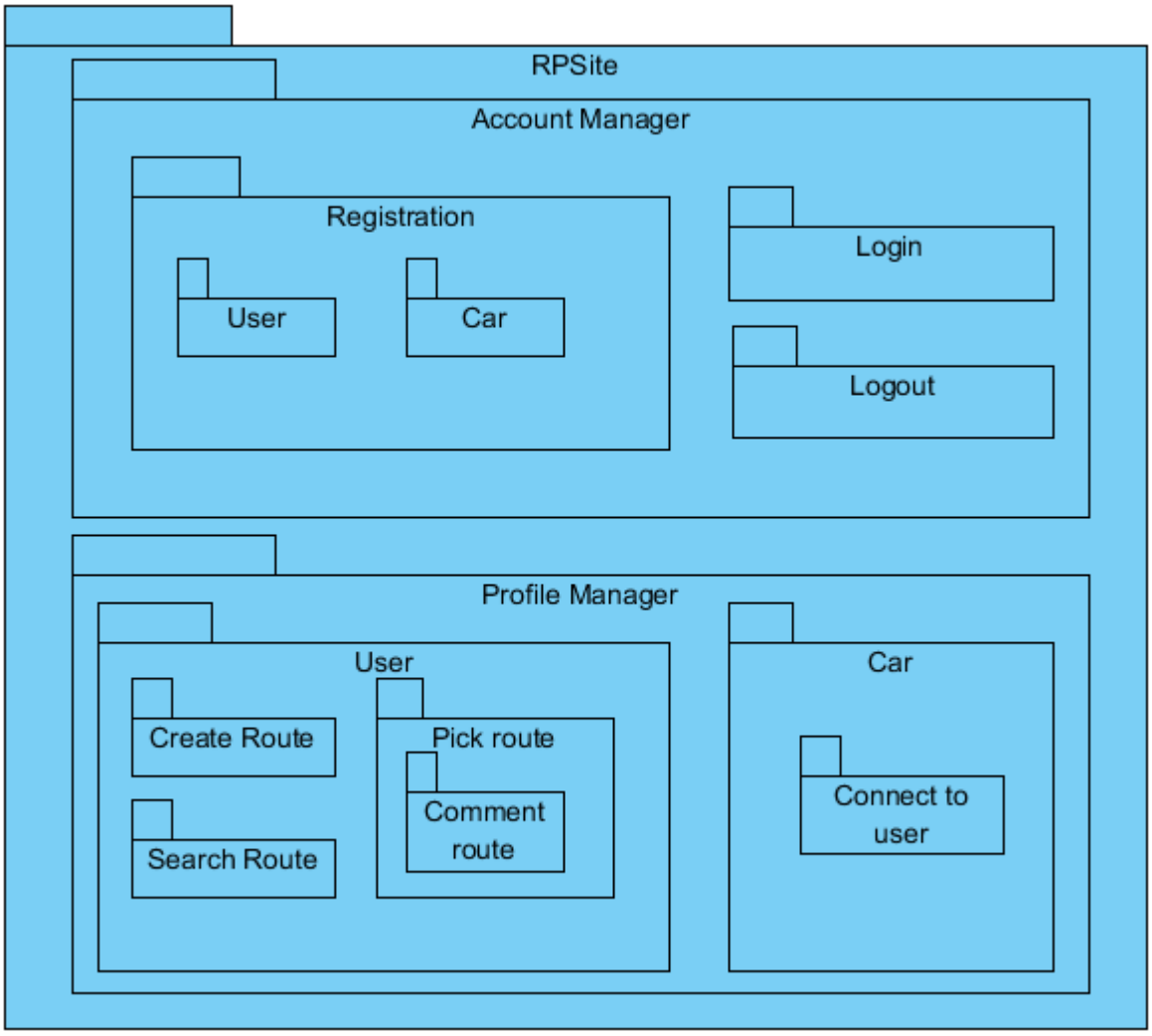
2.3.4.1 Грешки и изключения

При възникване на грешка по време на работата на сървъра, да се показва съобщение за грешка, след което сървърът да продължава нормално работата си.

2.3.4.2 Зависимости от други елементи

Модулът RPServer зависи от Базата от данни. При възникване на грешка при свързването на модула с базата от данни да се показва подходящо съобщение за грешка на съответното приложение и да се известява системният администратор.

2.4 RPSITE



2.4.1 Предназначение на модула

Модулът RPSite представлява уеб сайт, чрез който потребителите ще могат да се регистрират, да управляват профилите си и да се прави регистрация на МПС.

2.4.2 Основни отговорности в системата

Задачата му е да предоставя необходими полета за въвеждане на данни, да предоставя на сървъра данните от регистрацията и да визуализира на потребителя по подходящ начин резулатата от него.

2.4.3 Описание на интерфейсите на модулите

2.4.3.1 **Account Manager -** в този модул е събрана функционалността, свързана с автентикация и ауторизация на потребители и МПС-та.

public User GetCurrentUser()
public User GetCurrentCar()
boolean isAuthenticated()

2.4.3.1.1 **Registration**

2.4.3.1.1.1 **User -** отговаря за попълването на необходимите данни от потребителя и изпращането им до модул Сървър, където се случва самата регистрация. На негово ниво трябва да се валидират данните.

public string GetUsername()
public string GetPassword()
public string GetEmail()
public byte[] GetPicture()
public string GetNumber ()
public boolean ValidateUsername (string username)
public boolean ValidatePassword(string password)
public boolean ValidateEmail (string email)
public boolean ValidatePicture (byte[] picture)
public boolean ValidateNumber (string number)
public void SendRegisterRequest (string username, string password, string email, byte[] picture, string number)

- *Вход:* потребителско име, имейл, снимка, номер и парола
- *Изход:* съобщение дали операцията е успешна и/или списък от грешки

2.4.3.1.1.2 **Car -** отговаря за попълването на необходимите данни за регистрацията на МПС и изпращането им до модул Сървър, където всъщност се извършва регистрацията.

public string GetCarID()
public byte[] GetPicture()
public boolean ValidateCarID (string username)
public boolean ValidatePicture (byte[] picture)
public void SendRegisterRequest (string carID, byte[] picture)

- *Вход:* регистрационен номер на колата и снимка
- *Изход:* съобщение дали операцията е успешна и/или списък от грешки

2.4.3.1.2. **Login** - предоставя подходящи полета за попълване на данните от потребителя, след което ги изпраща на модул Сървър. Този модул е отговорен и за показване по подходящ начин на отговора, който сървърът е върнал - дали влизането е успешно или не.

public string GetUsername()
public string GetPassword()
public void SendLoginRequest (string username,string password)
public boolean ValidateUsername (string username)
public boolean ValidatePassword (string password)

- *Вход:* потребителско име и парола
- *Изход:* съобщение дали операцията е успешна и/или списък от грешки

2.4.3.1.3 **Logout** –изтрива всички данни за текущия потребител, които пази в браузъра или сесията, и извежда подходящо съобщение на потребителя.

public void SendLogoutRequest ()

- *Вход:* void
- *Изход:* съобщение дали операцията е успешна и/или списък от грешки

2.4.3.2 **Profile Manager** е достъпен след като самоличността на потребителя е установена. Функционалностите му са разделени на групи.

2.4.3.2.1. **User**

public void isUserLogged() – валидация дали има логнат в системата потребител, ако няма дава грешка и изисква влизане в системата

2.4.3.2.1.1. **Create Route –** визуализира по подходящ начин възможност за създаване на маршрут на споделено пътуване с включени начална, междинна и крайна точка, час на тръгване, изпращане на информацията до модул Сървър, за да бъде съхранена.

public void CreateRoute(string startingPoint, string intermediatePoint, stringfinalPoint)

2.4.3.2.1.2. **Pick Route –** чрез подходяща визуализация позволява на потребителя да заяви, че иска да се включи в споделено пътуване, като тези заявки могат да бъдат коментирани

public void PickRoute()
public void CommentPickRoute(string comment)

2.4.3.2.1.3. **Search Route** – на потребителя се предоставя възможност за търсене на превоз по начална или крайна точка

public void SearchByStartingPoint (string startingPoint)
public void SearchByFinalPoint (string finalPoint)

- *Вход:* начална или крайна точка
- *Изход:* списък от всички споделени пътувания, чиито начални или междинни точки съвпадат с подадената начална и чиито крайни точки съвпадат с крайната подадена

2.4.3.2.2. **Car**

2.4.3.2.2.1. **Connect to user** - модулът отговаря за подаването на информацията на МПС-то и потребителя на модул Сървър, който отговаря за свързването на МПС-то с потребител

public void ConnectUserToCar(string userid, string carid)

- *Вход:* информация за колата и потребителя
- *Изход:* съобщение дали операцията е успешна и/или списък от грешки

2.4.4 Други

2.4.4.1 Ограничения при употребата

За достъп до уеб сайта е необходима връзка с Интернет и браузър.

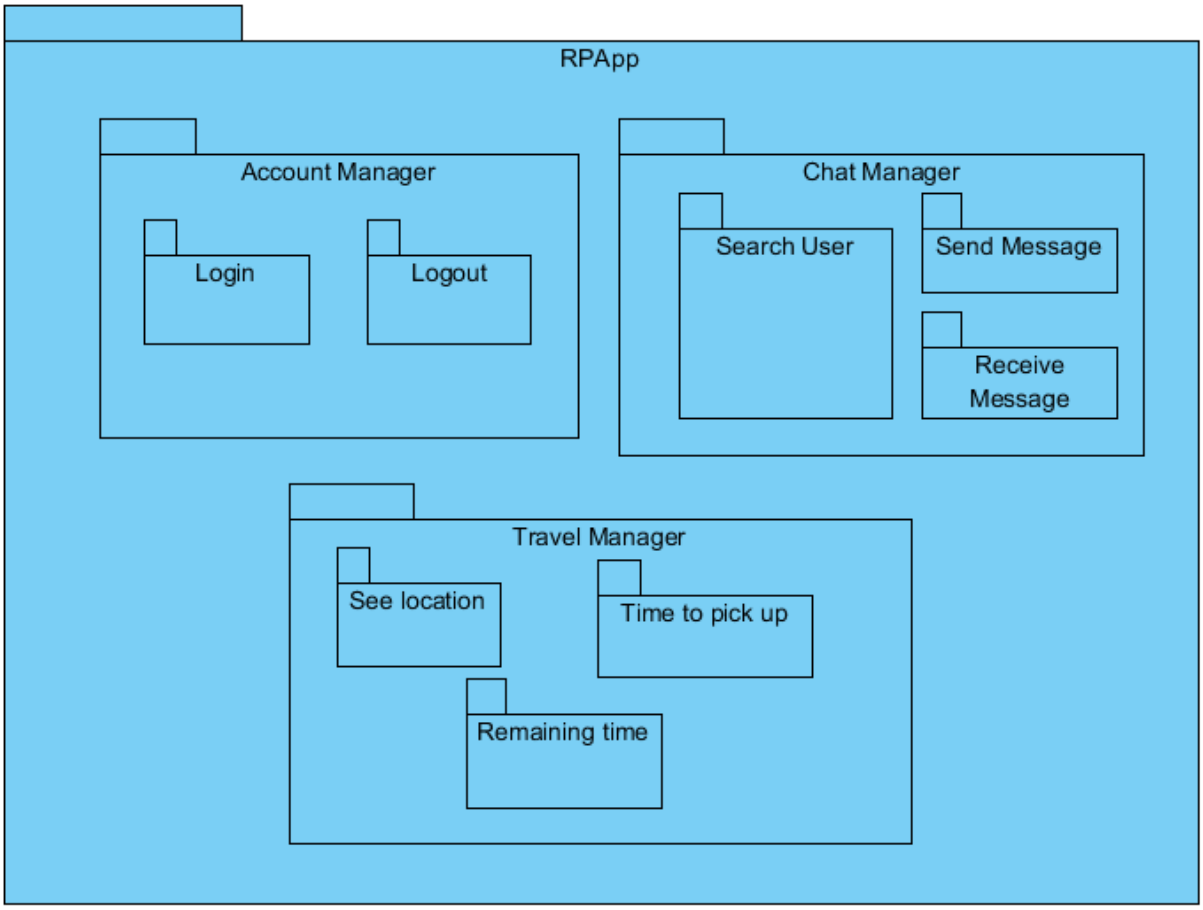
2.4.4.2 Грешки и изключения

При възникване на грешка по време на работата на приложението, тя да се визуализира по подходящ начин на потребителя.

2.4.4.3 Зависимости от други елементи

Модулът зависи от RPServer и Базата от данни. При грешка със свързването до тях, приложението спира работа.

2.5 RPAPP



2.5.1 Предназначение на модула

Това е мобилно и десктоп приложение, което крайните потребители използват за следене на МПС-то, с което ще пътуват, в реално време и да общуват с другите участници в споделеното пътуване

2.5.2 Основни отговорности в системата

Този модул отговаря за клиентската част на приложението, което крайните потребители използват.

2.5.3 Описание на интерфейсите на модулите

2.5.3.1 **Account Manager**

2.5.3.1.1 **Login** - предоставя полета за попълване на необходимите данни и ги изпраща на сървърния модул Login

public string GetUsername()
public string GetPassword()
public void SendLoginRequest (string username,string password)
public boolean ValidateUsername (string username)
public boolean ValidatePassword (string password)

2.5.3.1.2 **Logout -** реагира на задействането на механизма за Logout от потребителя като изпраща заявка до Сървъра. При успех изтрива всички данни за текущия потребител, които пази в локалната база, и извежда подходящо съобщение на потребителя.

public void SendLogoutRequest ()

2.5.3.2 **Chat Manager**

2.5.3.2.1 Search User – позволява на потребителя да търси други потребители по потребителско име, с които да комуникира

public void SearchUser(string userID)

- *Вход:* потребителско име или част от него
- *Изход:* списък от потребители, отговарящи на условието, визуализирано по подходящ начин

2.5.3.2.2 Send Message – изпраща съобщение до друг потребител
public void SendMessage(string userID, string message)

2.5.3.2.3 Receive Message – потребителят получава съобщение, визуализирано по подходящ начин
public void ReceiveMessage(string senderID, string message)

2.5.3.3 **Travel Manager**

2.5.3.3.1 See location – устройството на което е приложението, трябва да има GPS, който да показва на другите потребители, включени в споделеното пътуване, къде в реално време се намира МПС-то.

public void SeeLocationOfCar(string carID)

2.5.3.3.2 Time to pick up – показва на потребителите, включени в споделеното пътуване, колко време остава докато МПС-то дойде да ги вземе

public void TimeToPickUp(string carID)

2.5.3.3.3 Remaining time - показва на потребителите, включени в споделеното пътуване, колко време остава до пристигане до крайната точка

public void RemainingTime(string carID)

2.5.4 Други

2.5.4.1 Ограничения при употребата

За достъп до уеб приложението е необходима връзка с Интернет и браузър.

2.5.4.2 Грешки и изключения

При възникване на грешка по време на работата на приложението, тя да се визуализира по подходящ начин на потребителя.

2.5.4.3 Зависимости от други елементи

Модулът зависи от RPServer и Базата от данни. При грешка със свързването до тях, приложението спира работа.

2.6 PLATFORM ADAPTER

2.6.1 Предназначение на модула

Този модул отговаря за предоставянето на единен интерфейс за достъп до различни операционни системи според нуждите на приложенията.

2.6.2 Описание на интерфейсите на модулите

Модулите съдържат имплементация на интерфейса за съответната операционна система.

2.6.2.1 **Desktop OS**

Windows:

Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10

Linux:

Ubuntu, Debian

MacOS X

2.6.2.2 **Mobile OS**

Android

iOS

Windows: Windows 7, Windows 8

2.6.3 Други

2.6.3.1 Зависимости от други елементи

Platform Adapter-а зависи от операционната система. Ако има нова версия, то трябва да се добави функционалност и за нея.

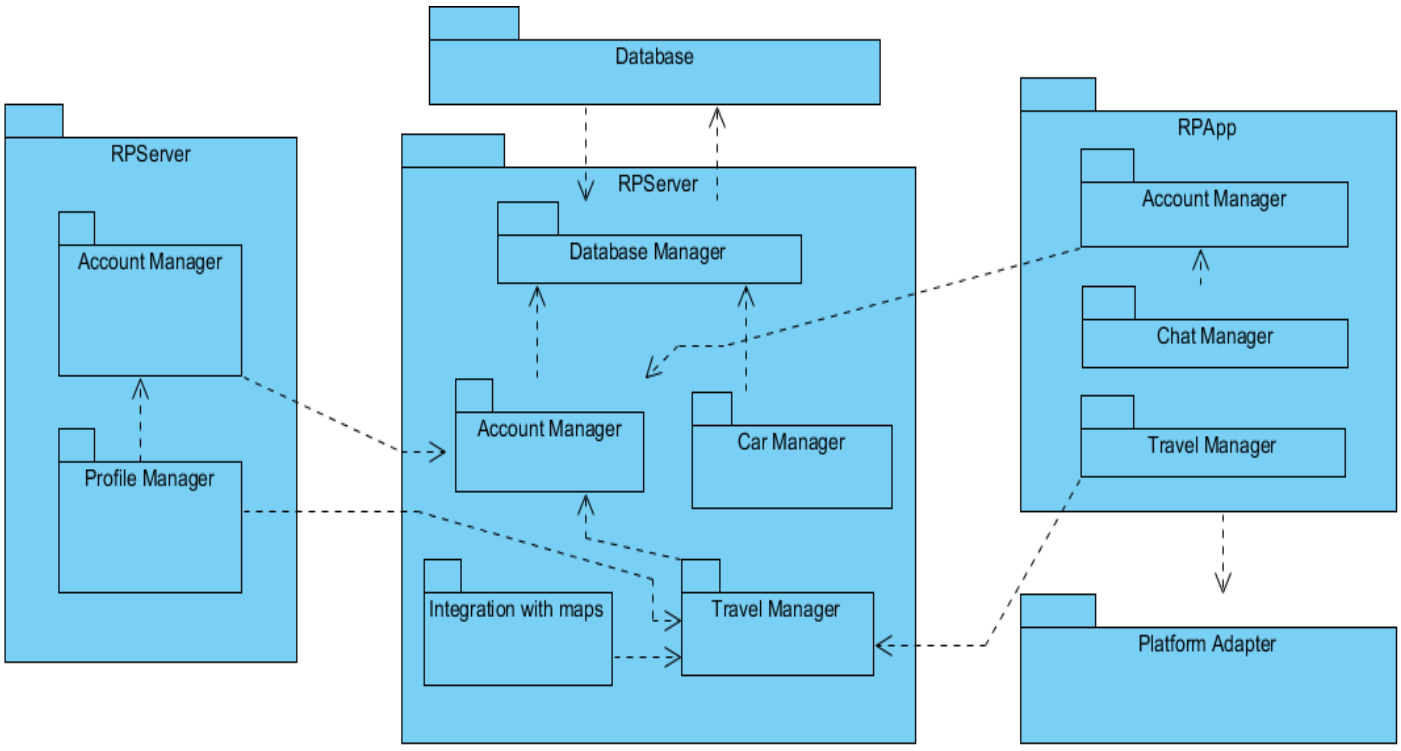
3. Описание на допълнителните структури

3.1 УПОТРЕБА НА МОДУЛИ

3.1.1 Мотивация за избор

Решихме, че за да представим по най-добър начин взаимодействието и връзките между модулите, които са описани в структурата Декомпозиция на модулите, ще изберем структурата Употреба на модули. Така ще можем по-лесно да покажем кои са отделните процеси, които протичат в системата. Също така добавя информация, която е пропусната в структурата Декомпозиция на модулите. Представянето на информацията чрез по-голяма абстракция, което подобрява и четимостта, позволява да покажем къде има възможност за лесно добавяне на нова функционалност, което е жизнено важно за използваемостта на системата.

3.1.2 Първично представяне



3.1.3 Описание на елементите и връзките

Структурата Употреба на модули се състои от модули и връзките между тях, или по - точно кой модул кой използва. Модулите представени в тази структура са описани детайлно (и поотделно) в структурата Декомпозиция на модулите.

3.1.3.1 Вътрешни връзки в RPServer

- Account Manager и Car Manager изпращат и получават информация за регистрираните потребители от Database Manager, който след това се обръща към базата от данни
- Travel Manager използва Account Manager, за да свържи съответните маршрути към дадения потребител
- Travel Manager използва модула Integration with maps, за да отбележи на съответната карта, която се използва, началната, междинната и крайната точка на маршрута
-

3.1.3.2 Вътрешни връзки в RPSite

- Profile Manager използва Account Manager, за да свържи съответна новорегистрирана кола към вече съществуващ потребител

3.1.3.3 Вътрешни връзки в RPApp

- Chat Manager използва Account Manager, за да вземе информацията за потребителите при търсене на потребител за комуникация

3.1.3.4 Връзка между RPServer и базата от данни

- RPServer и Database се свързват единствено чрез модула Database Manager, който се намира в RPServer. Той служи като адаптер между двата главни модула и праща информация на Account Manager, Car Manager и Travel Manager (подмодулите на RPServer) от базата от данни

3.1.3.5 Връзка между RPServer и RPSite

- Account Manager на RPSite се свърза с Account Manager на RPServer, който му предоставя информация за регистрациите и регистрираните потребители
- Profile Manager на RPSite се свърза с Travel Manager на RPServer, който му предоставя информация за маршрутите

3.1.3.6 Връзка между RPServer и RPApp

- Account Manager на RPApp се свърза с Account Manager на RPServer, който му предоставя информация за регистрациите и регистрираните потребители
- Travel Manager на RPApp се свърза с Travel Manager на RPServer, който му предоставя информация за маршрутите

3.1.3.7 Връзка между RPSite и RPApp

- Тяхната връзка се осъществява чрез RPServer

3.1.3.8 Връзка между RPApp и Platform Adapter

- RPApp се обръща към Platform Adapter, който отговаря за предоставянето на единен интерфейс за достъп до различни операционни системи според нуждите на RPApp.

3.2 Описание на обкръжението

Integration with maps (RPServer) дава възможност за интеграция на нови online карти като примерно Google Maps или maps.bg, което улеснява използването на системата, защото всеки потребител може да използва карта, с която вече е свикнал.

3.3 Възможни вариации

Към Platform Adapter могат да се добавят допълнителни различни операционни системи, а към Integration with maps (RPServer) както стана ясно може да се добавят различни online карти.

3.2 ВНЕДРЯВАНЕ

3.2.1 Мотивация за избор

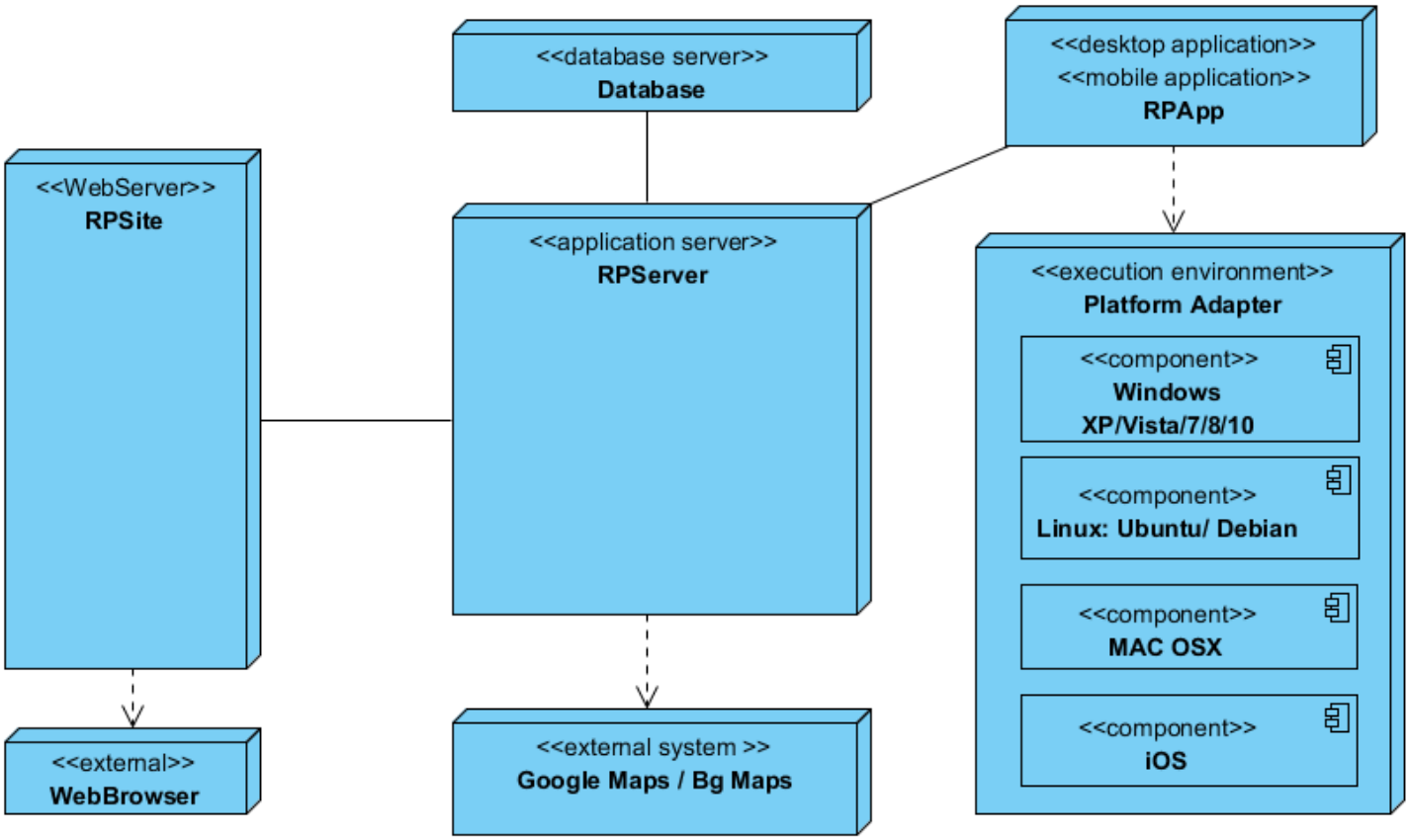
Избрахме структурата на внедряване поради характера на системата – тя се състои от няколко сървъра и клиентски приложения, и за да се разберат особеностите относно бързодействието, интегритета на данните, надеждността, сигурността и т.н. Тази структура показва също и как софтуера се разполага върху хардуера и комуникационното оборудване.

3.2.2 Първично представяне

Основни елементи в структурата са процеси, хардуерни устройства и комуникационни канали. Сървър е стартирана инстанция на софтуер, която може да приема заявки от клиент и да връща подходящи отговори.

Приложенията биват десктоп и мобилни. Това е софтуер, който се инсталира на съответното устройство. Той бива използван от крайния потребител на системата. Приложенията може да се обръщат към сървърите чрез заявки за изпълняване на определено действие.

Външните системи представляват софтуер, който не е физически свързан с настоящата система, но тя го използва за дадена функционалност. Външната система може да е по-специализирана или да се използва по-масово за решаването на проблема.



3.2.3 Описание на елементите и връзките

Database модулът се намира върху *Database Server*. Обособен е на отделна машина с цел предпазване на данните от външни въздействия и евентуални откази на RPServer.

RPServer модулът е разположен на *Application Server*. Той държи главната бизнес логика и клиентските приложения и уеб услуги се обръщат към него чрез заявки.

RPApp е *мобилно и десктоп приложение* за обикновени потребители. Има различни версии в зависимост от желаната операционна система, предоставени от Platform Adapter.

RPSite е *Web Server*, който си комуникира с RPServer за частта по бизнес логиката чрез заявки и с Web Browser за визуализирането на потребителския интерфейс чрез HTML5 страници.

Web Browser е *приложен софтуер*, чрез който на потребителите им бива предоставена възможността да достъпват функционалността на RPSite.

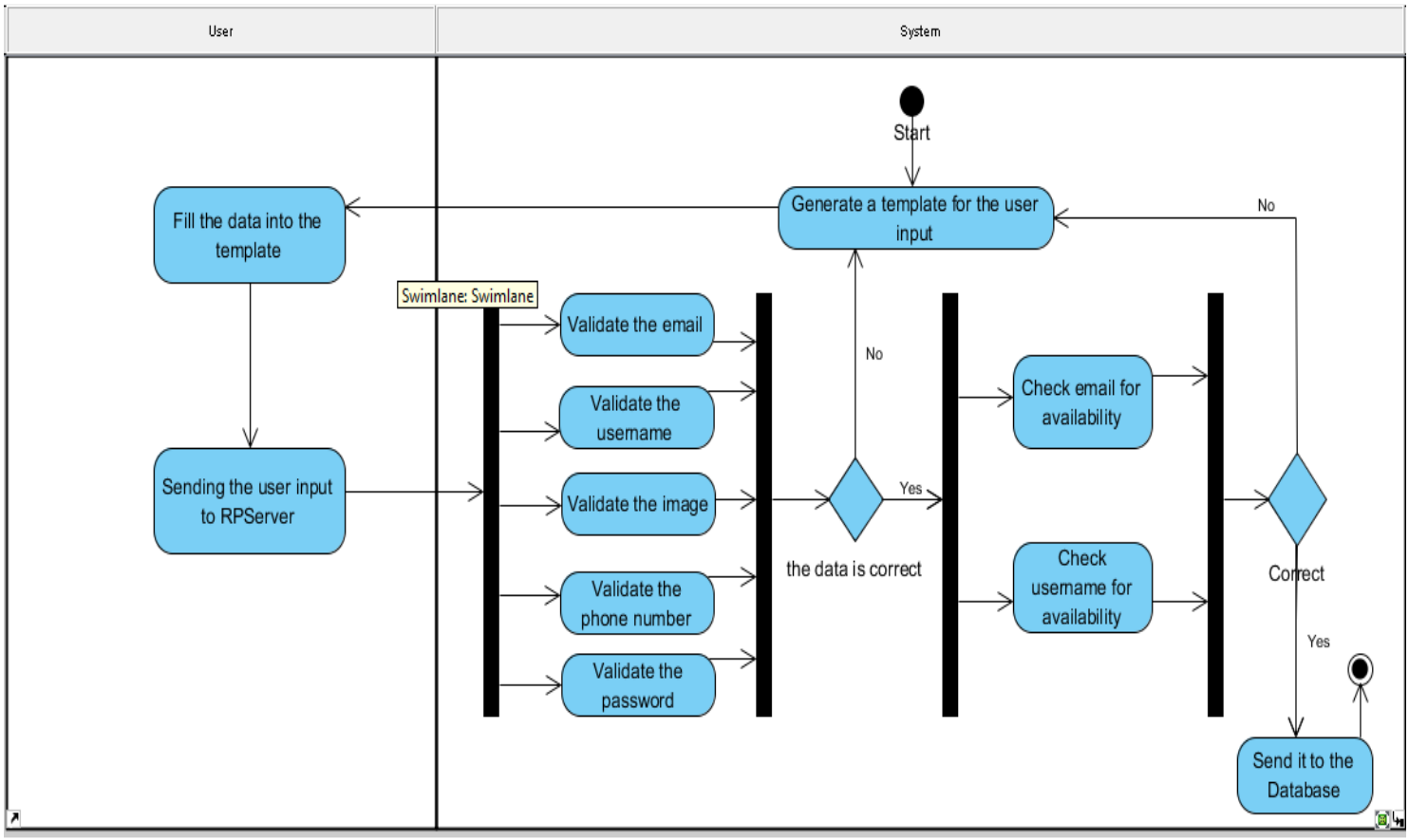
Google Maps / Bg Maps е *външна система,* която потребителите могат да използват съвместно със системата. Могат да бъдат интегрирани и други външни системи от този тип.

3.3 СТРУКТУРА НА ПРОЦЕСИТЕ

Мотивация за избор – представяме процесите от Account Manager на RPServer, които са най-основни за системата.

3.3.1 Процес на регистрация

3.3.1.1 Първично представяне

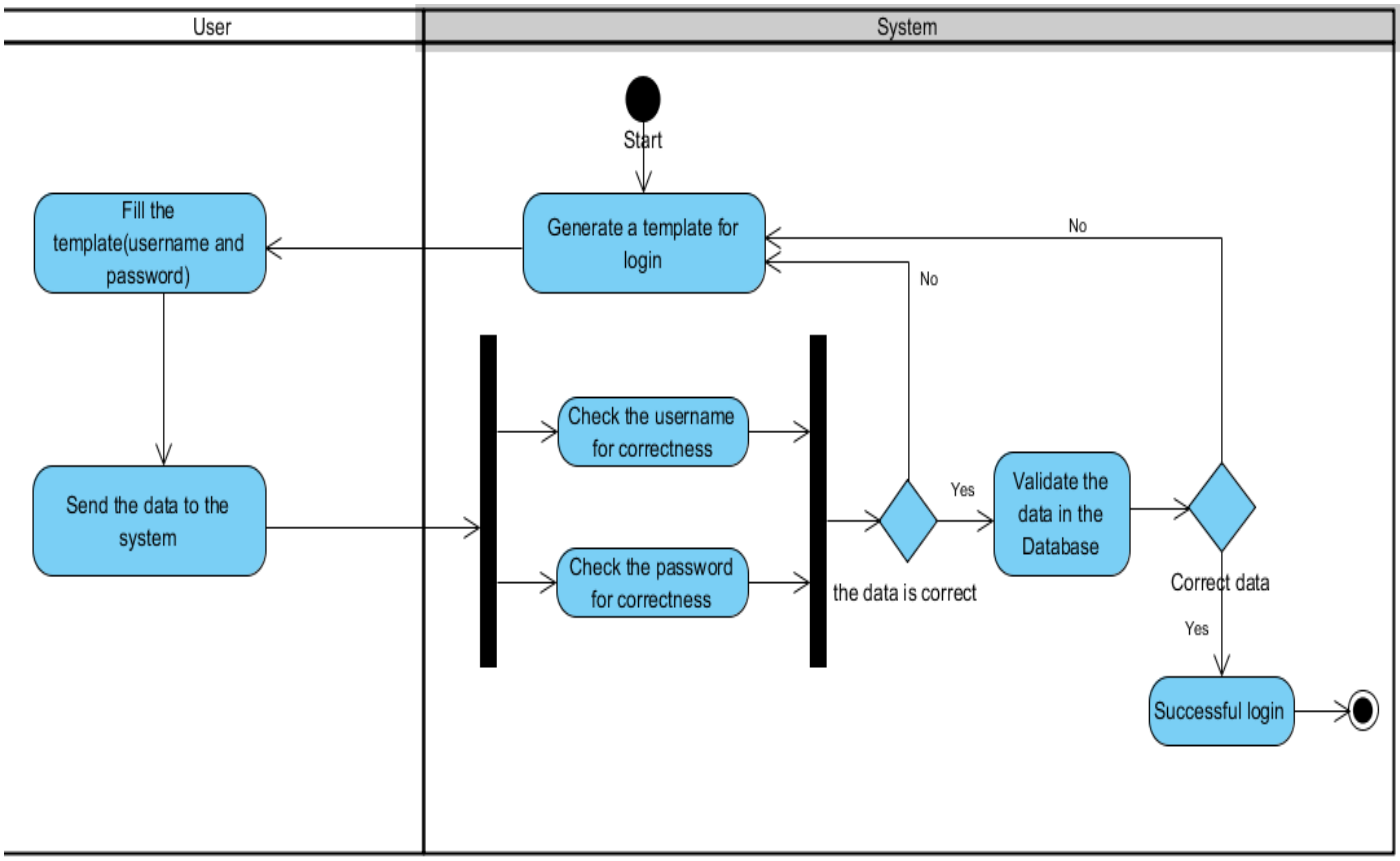


3.3.1.2 Описание на елементите и връзките

Потребителят въвежда данни в генериран за него шаблон за регистрация. След като въведе данните, те биват изпратени към RPServer, където се валидира информацията: e-mail, потребителско име, снимка, телефонен номер парола. Ако информацията не е в правилен формат се връщаме към първата стъпка, където потребителят наново трябва да попълни данните. Ако информацията премине валидацията, e-mail и потребителското име се проверяват за уникалност. Ако има вече регистрирани потребители с такива e-mail и потребителско име, се връщаме към първата стъпка, където потребителят на ново трябва да попълни данните. Ако няма такива потребители, информацията за новорегистрирания потребител се запазва в базата от данни. Процесът завършва успешно.

3.3.2 Процес на влизане в системата

3.3.2.1 Първично представяне

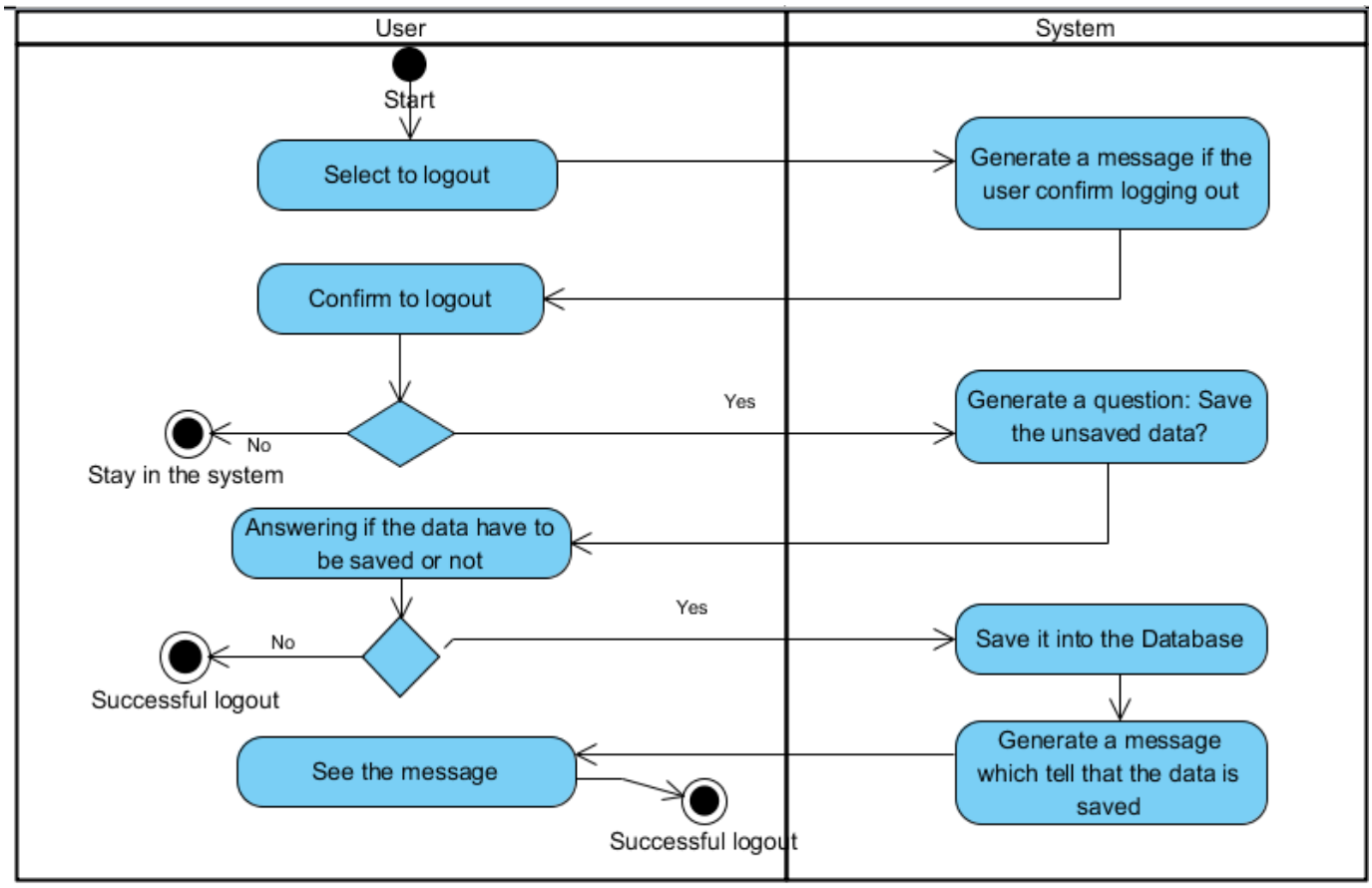


3.3.2.2 Описание на елементите и връзките

Потребителят въвежда данните си в генериран за него шаблон за влизане в системата. След като въведе данните (потребителско име и парола), те биват изпратени към RPServer, където се проверяват за правилност (примерно ако потребителското име не може да съдържа определен символ, а въведеният низ го съдържа; така системата отхвърля влизането в системата, без да се обръща към базата от данни, което е тежка операция). Ако не е правилна информацията, се връщаме към първата стъпка с шаблон за влизане в системата, където потребителят, отново е подканен да въведе информацията си. Ако е вярна, RPServer се обръща към Database за валидация на данните. Ако е грешна, се връщаме към първата стъпка с шаблон за влизане в системата, където потребителят, отново е подканен да въведе информацията си. Ако е вярна потребителят влиза в системата. Процесът завършва успешно.

3.3.3 Процес на излизане от системата

3.3.3.1 Първично представяне



3.3.3.2 Описание на елементите и връзките

При проявено желание за излизане от системата, RPServer генерира съобщение, с което запитва потребителя, дали наистина иска да излезе. Ако потвърди, че не иска да излиза от системата, той остава в нея и процесът завършва успешно.
Ако потвърди, че иска да излезе, системата го запитва дали иска да запамети незапазената информация (примерно маршрут). Ако не иска да бъде запазена, потребителят излиза от системата и процесът завършва успешно.
Ако иска информацията да бъде запазена, тя се запазва в базата от данни. Излиза съобщение за извършената операция и след като бъде видяно от потребителя, той излиза от системата и процесът завършва успешно.

4. АРХИТЕКТУРНА ОБОСНОВКА

Описание на това как софтуерната архитектура удовлетворява основните архитектурни драйвери (функционални и нефункционални ограничения). Представяне на тактиките използвани за реализиране на системата.

4.1 Функционални изисквания :

- Минимална информация при регистрация на потребител – имена, снимка и телефон за връзка
- Регистрация на МПС и свързването му с потребителски профил
- Създаване на маршрут на споделеното пътуване с включени начална, междинна и крайна точка, час на тръгване
- Търсене на споделено пътуване по начална и крайна точка и заявка за него
- Чат между потребителите

Биват реализирани чрез трите приложения на системата:

- *RPServer* – сървър, на който се съхранява основната бизнес логика на системата. Този модул играе и ролята на свързващо звено между базата от данни и RPSite и RPApp
- *RPSite* - уеб сайт, чрез който да се осъществява регистриране и управление на профилите на крайните потребители и да се прави регистрация на МПС
- *RPApp* - мобилно и десктоп приложение, което крайните потребители използват за следене на МПС-то, с което ще пътуват, в реално време и да общуват с другите участници в споделеното пътуване

Тази реализация се показва най-нагледно в Структурата на внедряване.

4.2 Интеграция с различни онлайн услуги за карти (пр. Google Maps) и възможност за интеграция на нови такива.

Реализирано чрез подмодула **Integration with maps** в RPServer, който позволява интеграция с различни онлайн услуги за карти. По-добро описание може да намерите в Декомпозицията на модули.

4.3 Availability – 100% наличност в работни дни

Намаляването на претоварването на системата(съответно и увеличение на нейната наличност) постигаме чрез подмодула Database Manager в RPServer, който се използва като свързващо звено между базата от данни и сървъра(т.е. с цялата система). Ако искаме да направим заявка за добавяне на информация в базата от данни първо правим проверка на данните и ако те са невалидни, въобще не правим това обръщение, което спестява ресурси.

4.4 Performance – Устойчива в пиковите часове (преди и след работа)

Архитектурният стил, който сме избрали (клиент-сървър), ни позволява лесното имплементиране на back-up на системата, което подобрява Performance.

4.5 GPS устройство

Трябва всеки, който използва системата да има GPS устройство, което да е свързано с нея (с цел всички участници в споделеното пътуване да могат да следят автомобила в реално време)

Чрез мобилното приложение RPApp, позволяваме да бъде изпълнено това условие, тъй като то ще бъде използвано на смарт телефони, който имат GPS.

4.6 Платформена независимост

Трябва системата да поддържа възможност за достъп чрез браузър или чрез мобилен клиент за iOS и Android

Това се постига през модула Platform Adapter, повече информация за него може да получите в структурата Декомпозиция на модули.

5. Други

5.1 Версия на системата и бъдещи насоки за развитие

По – горе в документа е представена софтуерната архитектура на системата RidePal. Това е едва първата версия на системата. Очаква се в бъдеще тя да се разраства и повече хора да започнат да я използват. За следващата версия е предвидено оправянето на получилите се бъгове и неточности в настоящата версия, защото колкото и прецизна да е една архитектура, и колкото и добър екип да работи по даден проект, винаги има грешки. За версия две на системата RidePal се планира да се добави възможност хора от различни фирми да споделят своето пътуване. Чрез различни услуги за карти ще се изчислява на какво разстояние са фирмите една от друга, за да знаят потребителите, използващи услугата, с кого могат да споделят своето пътуване.

5.2 Изготвили архитектурата

Настоящата софтуерна архитектура е изготвена от Женя Георгиева и Мария Димитрова, студенти във Факултета по математика и информатика към Софийски университет „Свети Климент Охридски“, София.