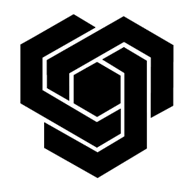
****

**Технически университет – София**

**ДИПЛОМНА РАБОТА**

Тема: Социална среда за възлагане на задачи реализирана на Android мобилно приложение

**Изготвил:**

Студент: *Георги Георгиев Карабойчев*

Факултет: *ФКСT*

Специалност: *КСИ*

Факултетен номер: *121218042*

| *Дата: 07.07.2022* | **Научен ръководител:**  ас. Костадин Панчев |
| --- | --- |
| *гр. София* |  |

**Задание**

**декаларация**

**СЪДЪРЖАНИЕ**

Увод………………………………………………………………………………...6

Първа Глава………………………………………………………………………..7

Анализ на съществуващи решения. Подобни приложения за възлагане на задачи. Избор на развойни среди....………………………………………..….....7

**1.1 Анализ на съществуващи решения**…….…………………………………7

1.1.1 TaskRabbit…………………………………………………….………….…..7

1.1.2 Wunderlist: To-Do List & Tasks………………………………………….…..8

1.1.3 Todoist: To-do lists for task management & errands……………………....10

1.1.4 Any.do: To-do list, Calendar, Reminders & Planne………………………....11

**1.2 Методи за комуникация с WorkIt сървър**…….……………………..…..12

1.2.1 REST програмният интерфейс……..……………………………...….…..12

**1.3 Развойни среди**……..…………………………………………………….....12

1.3.1 Android Studio……..……..…………………………………………….…..12

1.3.2 Visual Studio Community 2017…………………………...…..……..……..13

1.3.3 Microsoft SQL Managment Studio 17……...…………………………..…..13

1.3.4 Microsoft Azure……………………………………………………...……..14

Втора Глава……..……..……………………………………………………..…..15

Проектиране на приложението.………………………………………………....15

**2.1 Изисквания към програмния продукт**……..……………………..……..15

**2.2 Избор на езика за програмиране и софтуерните средства**……..……...15

**2.3 Структура на мобилното приложение**……..………………………….....16

2.3.1 “Model-View-Presenter” архитектурния шаблон……..………………....16

2.3.2 Пакетът “prot3ct.workit.config“……..…………………………………......17

2.3.3 Пакетът “prot3ct.workit.data”……..…………………………………...…..18

2.3.4 Пакетът ”prot3ct.workit.models”……..………………………………..…..18

2.3.5 Пакетът “prot3ct.workit.utils”……..……..……..……………………...…..18

2.3.6 Пакетът ”prot3ct.workit.views”……..……..…………………………….....19

**2.4 Структура на базата данни**……..……..…………....…………………......21

2.4.1 База данни качена в мрежата……..……..………………………………...22

2.4.2 Локална база данни……..…………………………………………...……..25

**2.5 Осъществяване на заявки към сървъра**……..……..…………………...25

2.5.1 “GET” заявки……..……..……………………………………………..…..25

2.5.1 “POST” заявки……………………………………………………….……..26

Трета Глава……..……………………………………………………….....……..28

Разработка……………………………………………………..……………...…..28

**3.1 Реализация на мобилното приложение**……..……..………………...…..28

3.1.1 Начална точка……..………………………………………………………..28

3.1.2 Login……..……..……………………………………………………....…...29

3.1.3 Register……..………………………………………………………...……..29

3.1.4 ListTasks……..……..…..…………………………………………………...30

3.1.5 TaskDetails……..……..………………………………………………....….30

3.1.6 CreateTask……..……..……..…………………………………………..…..31

3.1.7 EditTask……..………..……..…………………………………………..…..31

3.1.8 CompletedTasks……..……..……..……………………………………..…..31

3.1.9 MyTasks……..………..……..…………………………………………..…..32

3.1.10 ListTaskRequests..….…..……..…………………………………………...32

3.1.11 Profile..………………..…..……..……………………………………..…..32

3.1.12 EditProfile..………………...…………………………………………..…..33

3.1.13 Навигационно меню..……………...…..……………………………..…..33

**3.2 Реализацията на сървърното приложениe**……..…..…………......……..34

3.2.1 AuthController……..……..…………….…………………………….....…..34

3.2.1 UsersController……..……..………………………………………….....…..35

3.2.2 TasksController……..……..…………………………………………….…..35

3.2.3 TaskRequestController……..……..…………………………………….…..37

3.2.4 RaitingsController……..……..……………………………………………..37

**3.3 Основни методи и класове, които са заложени в Android**……..….…..38

3.3.1 Интерфейс……..…………………………………………………….……..38

3.3.2 Преминаване между екрани……..…………………………...……..……..38

Четвърта Глава……..……..………………………………………………….…..40

Реализация и работа с приложението…...…………………………….....……..40

**4.1. Изисквания и инсталация на приложението**……..……..……………..40

4.1.1. Инсталация на приложението през устройството…………………..…..40

**4.2. Работа с приложението**……..…………………………………..…..……..44

4.2.1 Създаване на акаунт……..……..……………………………………....…..44

4.2.2 Влизане в акаунт……..……..………………………………………….…..45

4.2.3 Създаване на задача……..……..……………………………………....…..45

4.2.4 Записване за задача……..…………………………………………...……..46

4.2.5 Профил на потребителя……..………………………………....……....…..46

4.2.6 Навигационно меню……………………………...……..………......……..47

4.2.7 Списък с моите задачи………………....………...……..………......……..47

4.2.8 Навигационно меню……………………………...……..………......……..48

4.2.9 Списък със заявители към задача………..……………..………......……..48

4.2.10 Навигационно меню………………...……………..…..………......……..49

# **УВОД**

В днешно време много хора си търсят работа без обвързване, тоест гъвкаво работно време. Това са предимно тийнейджъри, имащи свободно време и желаещи да изкарат пари с такъв вид работа.

В България почти няма активна среда, където човек може да намери работа, която може да свърши, когато му е удобно, и да получи някакво адекватно възнаграждение за свършената работа.

Хората често се нуждаят от помощ с вършенето на домашните задачи или други задължения, затова се нуждаят от някакъв софтуер, който да им помага. Workit решава точно този проблем, позволявайки на потребителите да създават задачи и съответно да назначават определен потребител, който да изпълнява отделните задачи. Дава възможност и за комуникация, за да може да се уточнят подробностите около задачите.

WorkIt позволява на отделните потребители, които търсят някого да им свърши някаква работа или съответно търсят да свършат някаква работа срещу възнаграждение, да се намерят и комуникират много по-лесно.

Сорс код: ”https://github.com/prot3ct/WorkItNow”.

# 

# **ПЪРВА ГЛАВА**

# **АНАЛИЗ НА СЪЩЕСТВУВАЩИ РЕШЕНИЯ. ПОДОБНИ ПРИЛОЖЕНИЕ ЗА ВЪЗЛАГАНЕ НА ЗАДАЧИ. ИЗБОР НА РАЗВОЙНИ СРЕДИ И СРЕДСТВА.**

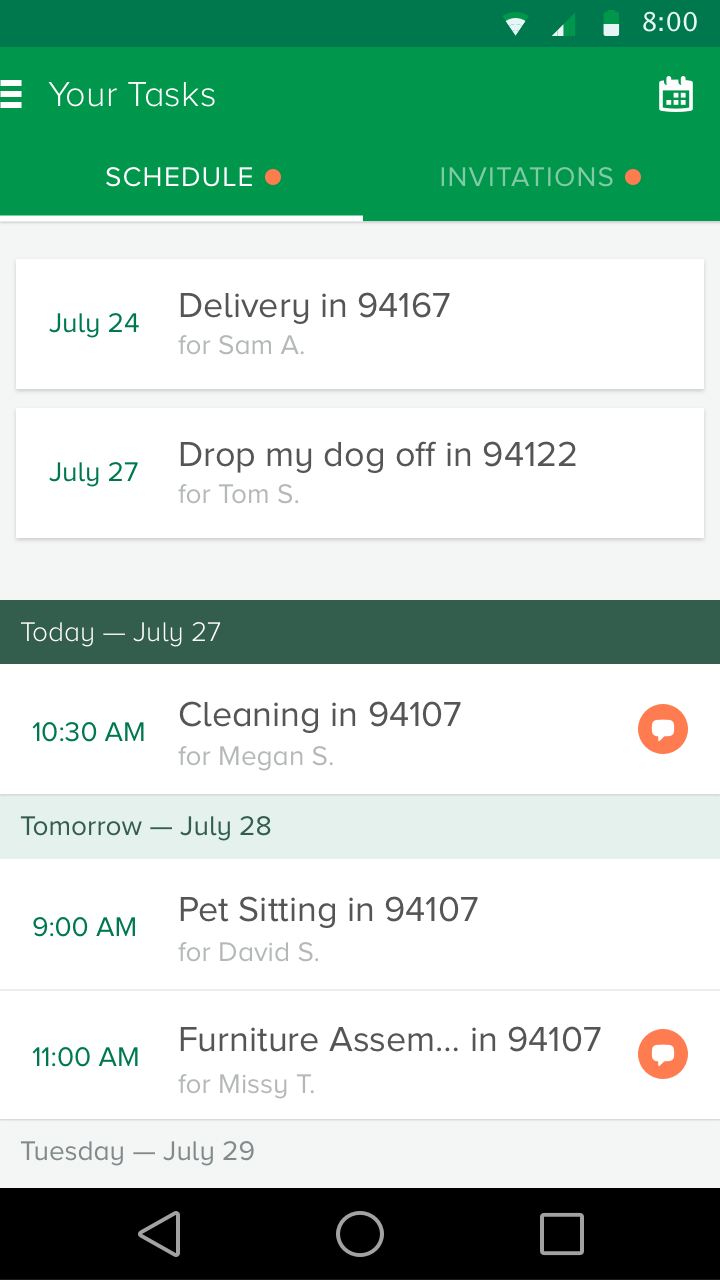
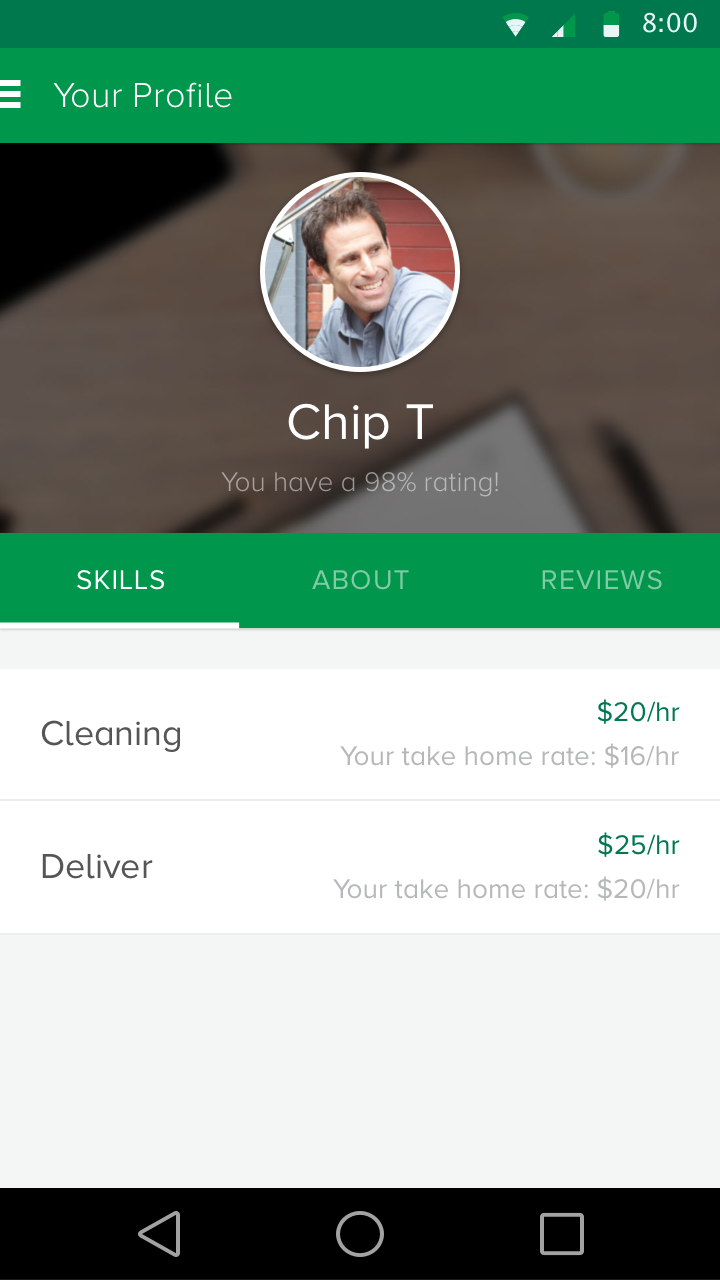
**1.1 Анализ на съществуващи решения**

Съществуват много приложения за възлагане на задачи. Повечето от тях са персонални, тоест потребителя си създава задачи, които трябва да свърши и в последствие му бива напомнено за тях. В рамките на проучването ще се разгледат приложения, които не само позволяват създаване на задачи, но и комуникация между потребители. Приложенията, които са разгледани, са:

TaskRabbit, Wunderlist: To-Do List & Tasks, Todoist: To-do lists for task management & errands, Any.do: To-do list, Calendar, Reminders & Planne.

**1.1.1 TaskRabbit**

TaskRabbit е напълно безплатно, има всички функционалности, които и WorkIt притежава, но се отличава с това, че не е достъпно в България. Другата основна разлика е, че потребителите, които искат да създават задачи, могат да ги създават само по вече създадени шаблони и TaskRabbit автоматично назначава някого, който да извърши задачата.

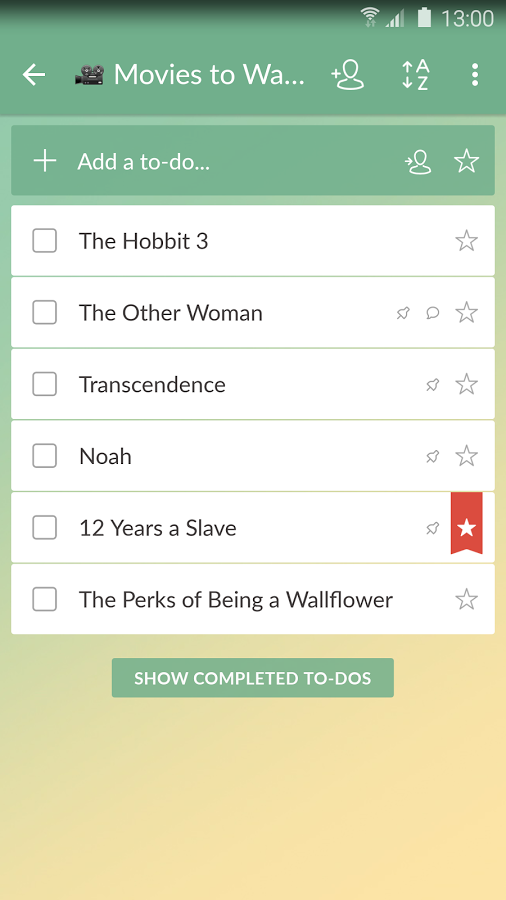
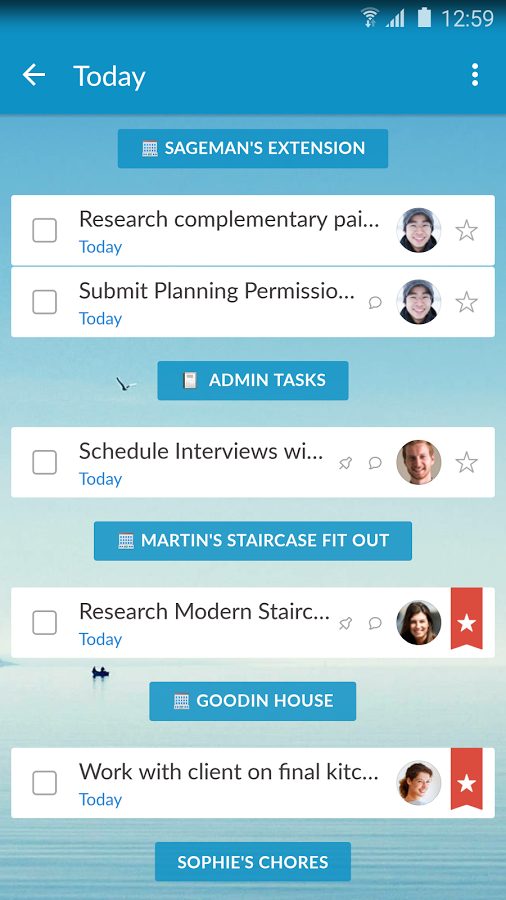


**Фиг. 1.1 Профил на Фиг. 1.2 Списък със собствените потребителя в TaskRabbit задачи в TaskRabbit**

**1.1.2 Wunderlist: To-Do List & Tasks**

Напълно безплатно, много прилича по функционалност на разработеното приложение. Началната страница представлява един изглед към групите, в които участва даден потребител. Към отделните групи може да се добавят задачи или потребители.

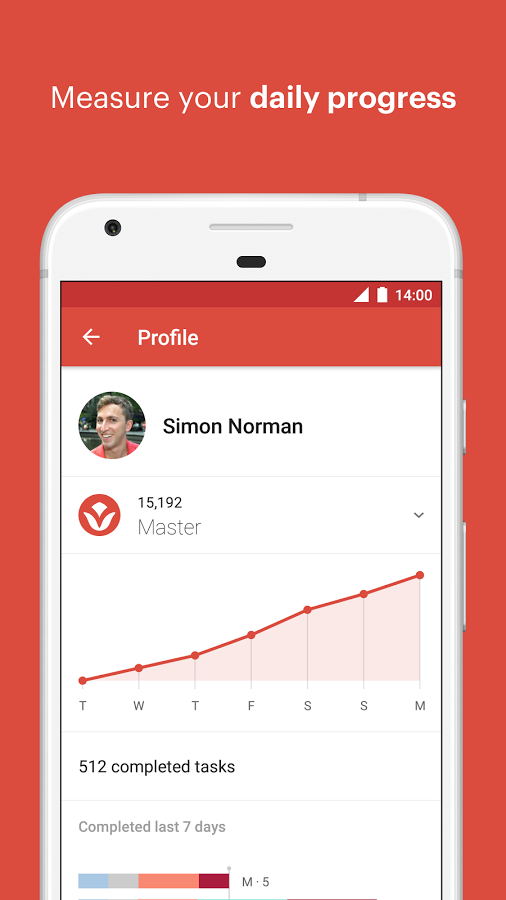
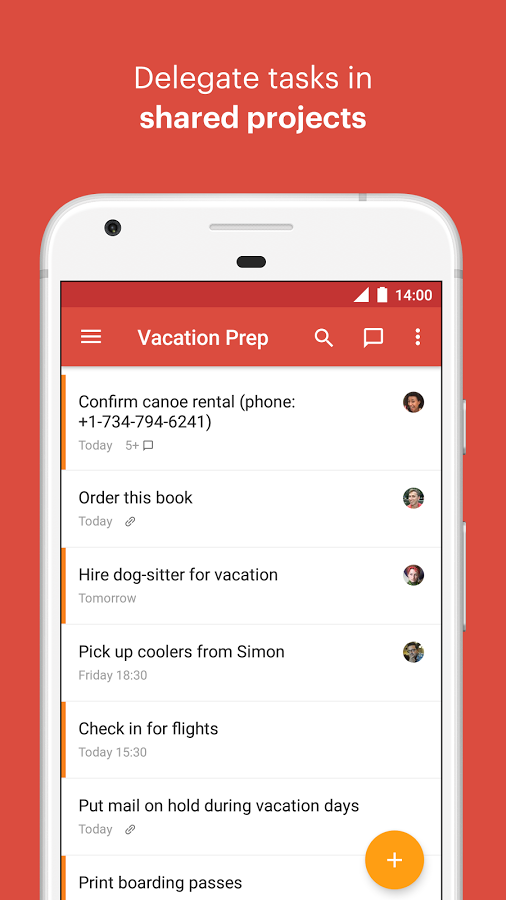
Потребители могат да създават задачи в отделните групи и съответно да избират кой да бъде изпълнител. Като цяло приложението е доста лесно за употреба и просто, но върши много добра работа. През годините “Wunderlist” постига много голям успех и впоследствие групата специалисти, разработили приложението, биват наети от Microsoft, за да работят по “Microsoft To-Do”.



**Фиг. 1.3 Списък с задачи на Фиг. 1.4 Списък със собствените отделните потребители задачи на Wunderlist на Wunderlist**

**1.1.3 Todoist: To-do lists for task management & errands**

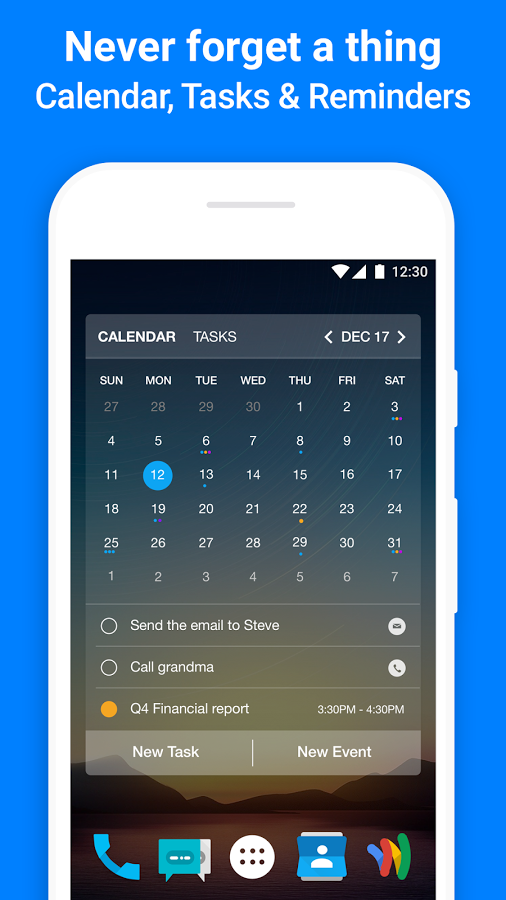
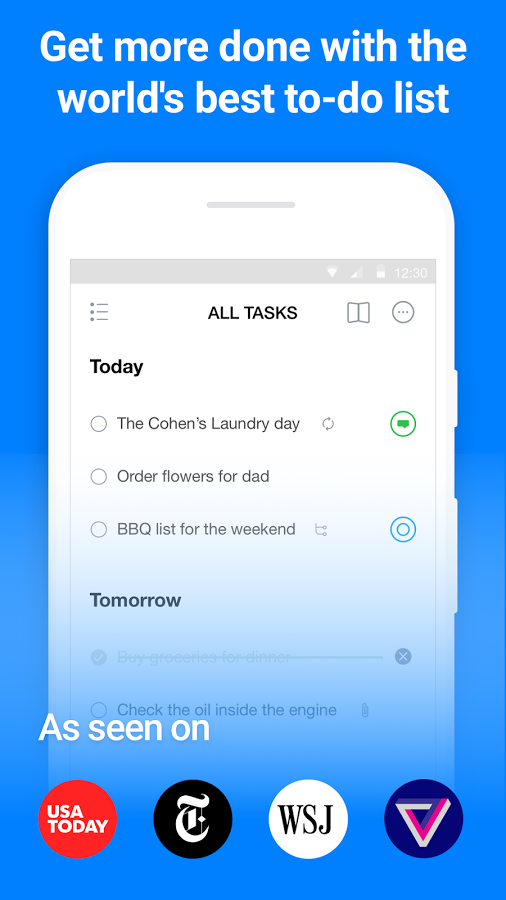
“Todoist: To-do lists for task management & errands” отново е много сходно към досега разглежданите приложения и е напълно безплатно. Има повече от 10 милиона потребители. Достъпно е на всяка платформа (Windows, Andriod, iOS...). Отличава се с това, че може да следи прогреса на отделните потребители и да се види как се справят със своите задачи.(**Фиг. 1.4**). Интегрирано е със повече от 60 популярни приложения като: Dropbox, Amazon Alexa, Zapier, IFTTT, and Slack.



**Фиг. 1.5 Списък с задачи на Фиг. 1.6 Профилна страница на на Todoist Todoist**

**1.1.4 Any.do: To-do list, Calendar, Reminders & Planne**

“Any.do: To-do list, Calendar, Reminders & Planne” е напълно безплатно и има 20 милиона потребители. Приложението е синхронизирано между мобилните, десктоп, уеб и таблет устройствата. Възможна е работа в екипи чрез създаването на група, както и възлагане на задачи на отделните потребители в тази група. Приложението има и приспособление за началния екран на мобилните устройства. В приложението има и вградено разпознаване на говор, за да може още по-лесно да се създават задачи.



**Фиг. 1.7 Списък с задачи на Фиг. 1.8 Календар на Аny.do на Any.do**

**1.2 Методи за комуникация с WorkIt сървър**

**1.2.1 REST програмният интерфейс**

Достъпа до “WorkIt” сървъра става чрез “REST” програмния интерфейс. “Трансфер на репрезентацията на състоянието” (“Representational state transfer”) или накратко “REST” е много известна софтуерна архитектура за достъп до ресурси в уеб средата. Рой Фийлдинг въвежда термина през 2000 г. Той е един от авторите на “HTTP” “Хипертекстовия трансферен протокол” (“Hypertext Transfer Protocol”).

Най напред “REST” архитектурата е била реализирана от “Глобалната световна мрежа” (“World Wide Web”). Основната идея на архитектурата е - потребителят прави заявка към “Уникален идентификатор на ресурс

” (“Uniform Resource Identifier”) или накратко “URI” и съответно сървърът връща отговор, който най-често се състои от “XML”, “HTML” или “JSON”, “HTTP” протокола, най-често използваните операции са “GET”, “POST”, “PUT”, “DELETE”.

**1.3 Развойни среди**

**1.3.1 Android Studio**

За разработката на мобилното приложение е използвано “Android Studio”, което включва “Android Software Development Kit” (Android SDK), което осигурява всички необходими инструменти за разработването на “Android” приложение. Приложенията се пишат на програмният език “Kotlin”. То е базирано на “JetBrains' IntelliJ IDEA” и е разработено специално за разработка на “Android” приложения. “Android SDK” включва компилатор, дебъгер, емулатор, много примери от код и ръководства за използването на отделните инструменти.

До 2015г. Официалната среда за разработка на “Android” приложения е била “Eclipse”, използвайки “Android Development Tools” (ADT) плъгина. От 2015 г. “Android Studio” е официалната среда за разработка на “Android” приложения.

**1.3.2 Visual Studio Community 2017**

За разработката на уеб сървъра е използвано “Visual Studio Community 2017”. “Visual Studio” е “Интегрирана среда за разработка на софтуер” - “integrated development environment” (IDE), разработено от “Microsoft”. То включва редактор, който поддържа “IntelliSense”. “Visual Studio” поддържа 36 различни програмни езика. “Community” версията е напълно безплатна. За разработката на сървъра е използван програмния език “C#”.

Основните компоненти, които поддържа са:

- Редактор

- Компилатор

- Дебъгер

- Място за проектиране на визуална логика (Дизайнер)

**1.3.3 Microsoft SQL Managment Studio 17**

За разработка на базата данни е използвано “Microsoft SQL Managment Studio 17”. То позволява конфигурация, управление, и администрация на “Microsoft SQL Server”. Като цяло ни помага да осъществим цялата комуникация с базата данни. Базирано на “Visual Studio 2010”.

**1.3.4 Microsoft Azure**

Бива пуснато на 1 февруари, 2010г. под името “Windows Azure”, но после бива преименувано на “Microsoft Azure” на 25 март, 2014г.

“Microsft Azure” е “cloud computing” среда, която ни позволява да качваме, тестваме и управляваме приложения чрез глобалната мрежа от центрове за данни на Microsoft. Поддържа “software as a service (SaaS)”, “platform as a service (PaaS)” и “infrastructure as a service (IaaS)” и поддържа много различни програмни езици, инструменти и структури.

# 

# 

# 

# **ВТОРА ГЛАВА**

# **ПРОЕКТИРАНЕ НА ПРИЛОЖЕНИЕТО**

**2.1 Изисквания към програмния продукт**

Създаване на “Android” мобилно приложение, което да комуникира с “REST” сървър. Приложението трябва да има потребители, тоест възможност за създаване и влизане в акаунт. Всеки потребител трябва да има възможност да създава задачи, към които останалите потребители да могат да се записват. Той самия трябва да може да се записва да участва в задачи.

Всеки потребител трябва да притежава оценка, която да показва как се е справил като изпълнител на задачи и като създател. Трябва да може и дава оценка на останалите потребители. Цялата информация трябва да се изпраща на “REST” сървър и да се запазва в база данни. Информацията от сървъра следва да бъде върната по подобаващ начин и да бъде визуализирана на мобилното приложение.

**2.2 Избор на езика за програмиране и софтуерните средства**

За създаването на приложението е използвана мобилната операционна система “Android” и “Windows” с програмните езици “Kotlin”, “C#” и средите - “Android Sdtuio”, “Visual Studio Community 2017”, “Microsoft SQL Server Management Studio 17”. “Kotlin” е официалният език за разработка на мобилни приложения под “Android Studio”, което предлага много инструменти, улесняващи работата на програмиста и спестяващи много време.

За разработката на сървъра е използван езика “C#” под “Visual Studio Community 2017” използвайки “ASP.NET MVC”. Причината за този избор е, че там всичко е много ясно и добре структурирано. Има много по-лесни и по-бързи начина да се напише сървър като например “nodeJS” или “Ruby on Rails”, но там колкото по-голям става проекта, толкова по-трудно става неговото поддържане и развиване. Това е причина и за избор на релационна база данни. При нерелационните бази данни приложението може да се развива много по-бързо, но когато стане прекалено голяма става много по-трудно за работа.

Като цяло “Android Studio” с “Kotlin” са водещ начин в днешно време за разработването на “Android” приложения, както и “ASP.NET MVC” е водещ инструмент за разработка на уеб приложения.

**2.3 Структура на мобилното приложение**

**2.3.1 Архитектурен шаблон “Model-View-Presenter”**

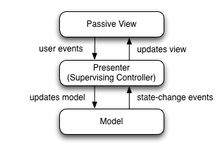
“Model-View-Presenter” е архитектурен шаблон за дизайн в програмирането, основан на разделянето на бизнес логиката от графичния интерфейс и данните в дадено приложение. За пръв път този шаблон за дизайн е използван в програмния език Smalltalk. Произхожда от “Model-View-Controller (MVC)” архитектурата.

Мобилното приложение използва “Model-View-Presenter (MVP)” архитектурата. “MVP” се използва предимно за изграждането на потребителския интерфейс. Основните предимства на “MVP” са много улеснено писане на автоматизирани тестове и приложението бива логически разделено по следния начин:

- Моделът (“Model”) казва какво ще съдържа информацията, която ще се визуализира.

- Изгледът (“View”) е този, който визуализира информацията(модела) и свързва командите на потребители с презентатора.

- Презентатор (“Presenter”) работи с модела и изгледа. Основната му работа е да получи информацията от модела и да я даде на изгледа да я визуализира.

**Фиг. 2.1 Диаграма, която демонстрира “Model-View-Presenter” архитектурния шаблон** 

**2.3.2 Пакетът “prot3ct.workit.config“**

В този пакет се съдържа уникалният локатор на ресурси “URL”, който води към сървъра, съответно с всичките му пътища (крайни точки).

**2.3.3 Пакетът “prot3ct.workit.data”**

В този пакет се съдържат всички класове, които правят заявки към сървъра и съответно получават и обработват получения резултат, например:

“Userdata” - Прави заявките свързани със създаването и ауторизирането на потребителя.

“JobData” - Прави заявките свързани със създаването, променянето и изтриването на дадена задача.

“TaskRequestData” - прави заявките свързани със създаването, променянето, отказването и приемането на желанието на даден потребител да се запише за някоя задача.

В този пакет се съдържа и връзката с локалната база данни на мобилното устройство. Това става чрез “SharedPreferenes”. “SharedPreferenes” представлява един “XML” файл.

**2.3.4 Пакетът ”prot3ct.workit.models”**

В този пакет се съдържат всички модели, което мобилно приложение използва.

**2.3.5 Пакетът “prot3ct.workit.utils”**

В този пакет се съдържат всички класове, които имат по общи функции. Като например:

* EndlessRecyclerViewScrollListener - служи при списъците със задачи, когато скролваме да се зареждат нови задачи.(по-известно като “infinite scroll”).
* OkHttpRequester - изпраща всички заявки към сървъра и получава техният отговор.
* HashProvider - използва се за хеширане на паролата, когато даден потребител се регистрира.
* GsonParser - използва се за обработка на данните, като ги превръща от “JSON” в инстанции на даден клас.

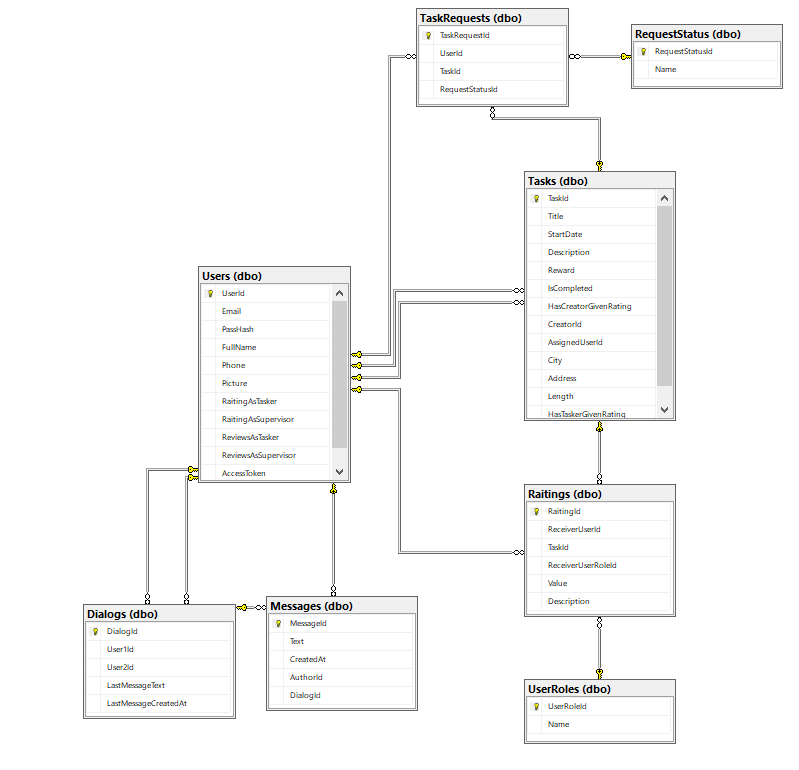
**2.3.6 Пакетът ”prot3ct.workit.views”**

В този пакет се съдържат всички изгледи(всеки един екран от приложение) заедно със своя презентатор, който служи за извършване на логиката на дадения екран. В приложението има следните изгледи:

* login - oт тук потребителя може да влезе в своя акаунт.
* register - oт тук потребителя може да се регистрира.
* navigation - това представлява навигационното меню, което излиза от лявата страна на екрана.
* list\_tasks - виждат се всички задачи, към които даден потребител може да се запише под формата на списък.
* Task\_details - виждат се детайлите за всяка една задача, това е и мястото, от което потребителя може да се запише да изпълни дадена задача.
* create\_task - потребителят може да създава задача, по определени от него изисквания.
* completed\_tasks - потребителят вижда всички задачи, в които е участвал и са приключили.
* assigned\_tasks - потребителят вижда всички задачи, в които е участник като изпълнилнител и не са приключили.
* my\_tasks - потребителят вижда всички задачи, които е създал.
* edit\_task - потребителят има възможно за редактиране на задача.
* profile - потребителят вижда информацията за своя профил или за нечии друг.
* list\_task\_requests - потребителят вижда списък със заявки към някоя негова задача.
* Edit\_prfile - потребителят има възможност да редактира своя профил.

**2.4 Структура на базата данни**

**2.4.1 База данни намираща се на REST сървъра**



**Фиг. 2.2 Диаграма на базата данни**

**2.4.2 Описание на базата данни**

“Users” таблицата служи за запазвана на информацията за потребителя. Притежава следните полета:

* UserId - основен ключ;
* Email - имейл на потребителя;
* PassHash - хешираната парола на потребителя;
* FullName - пълното име на потребителя;
* Phone - телефонния номер на потребителя;
* Picture - профилната снимка на потребителя;
* RatingAsTasker - оценката на потребителя като изпълнител на задачи;
* RatingAsSupervisor - оценката на потребителя като създател на задачи;
* ReviewsAsTasker - брой пъти получавана оценка като изпълнител;
* ReviewsAsSupervisor - брой пъти получавана оценка като създатетел;

“Tasks” таблицата служи за запазвана на информацията за всяка задачи. Притежава следните полета:

* TaskId - основен ключ;
* Title - заглавие на задачата;
* StartDate - начално време на зада а;
* Description - описание на задачата;
* CreatorId - идентификационния номер на създателя на задача;
* AssignedUserId - идентификационния номер на изпълнителя на задачата;
* City - град, в който се намира задачата;
* Address - адреса, на който се намира задачата;
* Length - дължина на в часове на задачата;
* HasTaskerGivenRating - поле, което пази дали изпълнителят е дал оценка за създателя;
* HasCreatorGivenRating - поле, което пази дали създателят е дал оценка за изпълнителя;

“Ratings” таблицата служи за запазвана на оценките на всеки един потребител. Притежава следните полета:

* RatingId - основен ключ;
* ReceiverUserId - идентификационен номер на получателя;
* TaskId - идентификацонен номер на задачата;
* ReceiverUserRoleId - идентификационен номер на ролята на получателя;
* Value - оценка;
* Description - описание към оценката;

“TaskRequests” таблицата служи за запазвана всички заявки на потребителя към отделните задачи. Притежава следните полета:

* TaskRequestId - основен ключ;
* UserId - идентификационен номер на потребителя, който изпраща заявката към задачата;
* TaskId - идентификацонен номер на задачата;
* RequestsStatusId - идентификационен номер на състоянието на задачата, което бива - “Pending”, “Accepted”, “Declined”;

“RequestStatuses” таблицата служи за запазването на видовете състояния на заявките. Притежава следните полета:

* RequestStatusId - основен ключ;
* Name - име на състоянието, биват (“Accepted”, “Declined”, “Pending”);

“UserRoles” таблицата служи за запазването на роли на потребителите. Притежава следните полета:

* UserRoleId- основен ключ;
* Name - име на ролята, биват (“Supervisor”, “Tasker”);

**2.4.2 Локално хранилище за данни**

За локално хранилище за данни приложението използва “SharedPreferences”. “SharedPreferences” представлява един “XML” файл. Информацията там се записва под формата на карта, в която всеки един ключ отговаря на даден елемент.

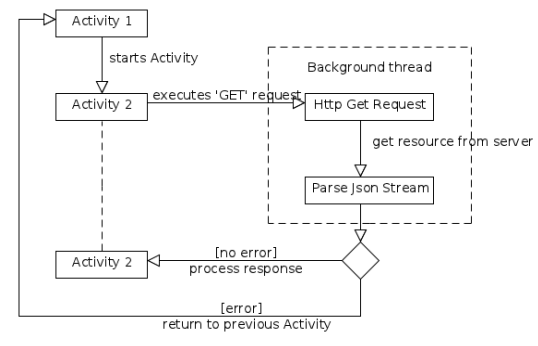
**2.5 Осъществяване на заявки към сървъра**

**2.5.1 “GET” заявки**

Стъпките, през които минава приложението от натискане на определен бутон, стартиращ заявка, до получаване на отговора, са показани на **Фиг. 2.3.**

Основната идея на взимане на данни е следната:

Първоначално мобилното приложение изпраща заявка към сървъра, излиза диалог, който показва, че приложението зарежда(изчаква се отговор от сървъра). Когато сървърът върне отговор, приложението визуализира резултата. Ако сървърът не върне отговор дълго време, тогава заявката дава грешка “Timeout”.

****

**Фиг. 2.3 Диаграма “GET” заявка**

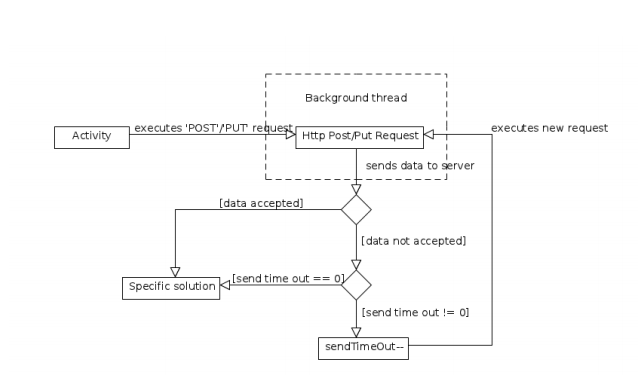
**2.5.1 “POST” заявки**

Начинът на работа на POST заявките е показан на **Фиг. 2.4.**

Чрез “POST” заявките най-често се записва някаква информация на сървъра, но дори да не се запише, то задължително се изпраща някаква информация към сървъра.

Начинът на работа е следният:

Мобилното приложение изпраща “POST” заявка към сървъра, сървърът прави необходимите проверки, например дали потребителят има право на тази заявка или дали е възможно записването на изпратените данни в базата данни, и на базата на проверките връща грешка или резултат, показващ, че всичко е наред.

****

**Фиг. 2.4 Диаграма на “POST” заявка**

**ТРЕТА ГЛАВА**

**РАЗРАБОТКА**

**3.1 Реализация на мобилното приложение**

Ще разгледаме мобилното приложение, описвайки работата, която се извършва във всеки един от екраните, които са следните - “Register”, “Login”, “ListJobs”, “JobDetails”, “CreateJob”, “Navigation”, “Profile”.

За прехвърлянето на информацията между отделните екрани (“Activity”) служи класът “App”, като всяко едно “Activity” наследява “AppCompatActivity”, което прави възможен достъпа му от всички останали екрани. Класът “App” имплементира “Singleton” шаблона за дизайн, който ни гарантира, че в приложението имаме точно една инстанция на даден клас.

Екраните на приложението са следните:

**3.1.1 Начална точка**

Началната точка на приложението е “LoginActivity”. Тя се задава в Manifest.xml файла. В него се описва основната информация за приложението и се използва от “Android build tools”, “операционната система Android” и “Google Play”. Обикновено “Android Studio” генерира този файл. В приложението имаме един навигационен бутон, който ни изкарва навигационно меню от лявата страна на екрана. Това меню присъства на всеки екран, освен “Login” и “Register”.

**3.1.2 Login**

“Login” екранът представлява две полета за въвеждане на текст - за имейл и за парола - и два бутона: един за влизане в акаунт и един за създаване на акаунт, в случай че потребителя няма регистриран акаунт.

Чрез метода “setOnClickListener” се добавя събитие, което ще се случи при натискане на един от двата бутона. При натискане на “Login” бутона се извивка метода “loginUser(String email, String password)”, като първи аргумент се подава имейла на потребителя, а като втори аргумент паролата. Докато се чака отговор от сървъра се появява “WorkItProgressDialog”, който визуално ни показва, че приложението зарежда. При успешно влизане в акаунт се стартира “ListJobs” екрана. При натискане на бутона за създаване на акаунт се стартира “Register” екрана.

**3.1.3 Register**

“Register” екранът представлява четири полета за въвеждане на данни на потребителя. Те са следните: цяло име, имейл, парола и потвърждение на паролата. При натискане на “Register” бутона се вземат данните от четирите полета и се подават на метода “registerUser(...)”, който е част от презентатора.

При успешно регистриране се стартира “Login” екранът и можем да влезнем в новосъздадения акаунт.

При грешка потребителят може да въведе нови данни и да опита да се регистрира отново.

**3.1.4 ListTasks**

“ListTasks” екранът представлява списък с всички задачи, към които потребителят може да се запише. В горния ляв ъгъл има бутон, който скрива и показва навигационното меню. В долния десен ъгъл има бутон за създаване на задача. Списъкът със задачи се взима от сървъра чрез метода “getAllTasks()”. Получените задачи се подават на “taskAdapter”, а той попълва списъка със задачи - “recyclerTaskView”. Всяка клетка от списъка със задачи представлява една задача със заглавие, дата, оставащо време и потребителят, който е създал задачата. При натискане на някоя от клетките се стартира “TaskDetails” екранът, като се изпраща идентификационният номер на задачата към новостартиралия екран.

**3.1.5 TaskDetails**

“TaskDetails” екранът представлява списък с подробностите около една задача. На него бутон за записване към задачата, при натискане се променя на бутон за отказване от задачата. На екрана се вижда и карта (“GoogleMaps”), която визуализира местоположението на задачата. При изпращане на заявката за задача се извиква метода “createTaskRequest”, който изпраща заявка към сървъра.

**3.1.6 CreateTask**

“CreateTask” екранът представлява няколко полета, в които се въвежда информацията за задачата. Полета се следните:

* заглавие на задачата;
* описание на задачата;
* начална дата на задачата;
* времетраене на задачата;
* град;
* адрес;
* заплащане за час;

При натискане на бутона за запазване на задачата, се появява “WorkItProgressDialog”, което визуализира зареждане, докато сървърът върне отговор, че задачата е била запазена.

**3.1.7 EditTask**

“EditTask” екранът е подобен на “CreateTask”, като основната разлика е, че вместо да създава задача, прави промени по вече създадена. При натискане на бутона за запазване на задачата, се появява “WorkItProgressDialog”, което визуализира зареждане, докато сървърът върне отговор, че задачата е била запазена.

**3.1.8 CompletedTasks**

“CompletedTasks” екранът е подобен на “ListTasks”, като основната разлика е, че вместо да се визуализират всички задачи, към които можем да се запишем, се визуализират задачите, които са приключили и трябват да бъдат оценени. Срещу всяка задача има бутон за даване на оценка на съответния потребител, който е участвал в задачата. При натискане на този бутон се показва прозорец, в които можем да изберем каква оценка искаме да дадем и да напишем коментар.

**3.1.9 MyTasks**

“MyTasks” екранът е подобен на “ListTasks”, като основната разлика е, че вместо да се визуализират всички задачи, към които можем да се запишем, се визуализират задачите, които сме създали, като на всяка задача има три бутона - бутон за преглеждане на записалите се хора, бутон за промяна на задачата и бутон за изтриване на задачата. Бутона за преглеждане на зпаисалите се хора е видим, само когато има заявки към задачата.

**3.1.10 ListTaskRequests**

“ListTaskRequests” екранът представлява списък със всички хора, които са направили заявка, че искат да участват в дадената задача. Има 2 бутона за приемане и за отказване на съответния потребител.

**3.1.11 Profile**

Профилният екран представлява подробности относно потребителя. В него има следните подробности:

* профилна снимка;
* имейл;
* телефонене номер;
* цяло име;
* рейтинг като изпълнител;
* рейтинг като създател;
* брой рейтинги като изпълнител;
* брой рейтинги като създател;

Има и бутон за редакция на профил, които стартира

**3.1.12 EditProfile**

“EditProfile” екран е подобен на профилния екран с основната разлика, че можем да променяме следните данни

* профилна снимка;
* имейл;
* телефонене номер;
* цяло име;

**3.1.13 Навигационно меню**

Навигационното меню съдържа следните бутони, които ни водят към нови екрани:

* бутон, който ни води към списъка със задачи, в които можем да участваме;
* бутон, който ни води към задачите, които сме създали;
* бутон, който ни води към задачите, към които сме се записали;
* бутон, който ни води към задачите, към които сме се записали и са приключили и очакват да бъдат оценени;
* бутон, който ни води към профилния екран;
* бутон, който ни изкарва от акаунта и ни връща на екрана за влизане в акаунт;

**3.2 Реализацията на сървърното приложение**

Сървърното приложение се състои от няколко слоя:

* “Controllers” - в тях се определя уникалният указател за ресурс “URL”. Чрез тях сървърът се справя с грешките и подава тялото на заявката на бизнес логиката;
* Модели, които биват три вида:
  + “Models” - служат за образец на това как ще изглежда базата данни;
  + “ModelsDTO” - служат за напасване на обекта, който получаваме в заявката. Известни са като “Data Transfer Object”;
  + “Viewmodels” - служат за образец на моделите, които сървъра връща;
* “BussinesLogics” - “BussinesLogic” класовете служат за изпълняване на логиката;

Приложението има една инстанция на класа “BaseService” и тя се подава на отделните “BussinessLogic” класове. По този начин си гарантираме, че имаме точно една връзка към базата и тя се създава в “BaseService” класа.

**3.2.1 AuthController**

Пътищата на “AuthController” са следните:

* “POST http://workit.azurewebsites.net/api/auth/login” - Влизане в акаунт, В тялото на заявката се подават имейла и паролата;
* “POST http://workit.azurewebsites.net/api/auth/register” - Регистриране на акаунт. В тялото на заявката се подава цялата информация за акаунта;

**3.2.2 UsersController**

Пътищата на “AuthController” са следните:

* “GET http://workit.azurewebsites.net/api/users/{userId}” - Получаваме информация относно потребителя с идентификационен номер “userId”;
* “PUT http://workit.azurewebsites.net/api/users/{userId}” - Обновяване на информацията на профила на даден потребител с идентификационен номер “userId”;

**3.2.3 TasksController**

Пътищата на “TasksController” са следните:

* “GET http://workit.azurewebsites.net/api/users/{userId}/available-tasks/page/{page}?search=...” - взимат се всички неизпълнени задачи, които са достъпни за записване и изпълнение от потребителя;
* “GET http://workit.azurewebsites.net/api/users/{userId}/my-tasks” - взимат се всички създадени задачи, които потребителят е създал;
* “GET http://workit.azurewebsites.net/api/users/{userId}/completed-tasks” - Взимат се всички изпълнени задачи за потребителя, независимо като какъв е участвал в тях;
* “GET http://workit.azurewebsites.net/api/users/{userId}/assigned-tasks” - взимат се всички задачи, към които потребителят е записан като изпълнител;
* “POST http://workit.azurewebsites.net/api/tasks” - Създава се задача, като е необходимо в тялото на заявката да се подадат следните параметри:

- заглавие;

- описание;

- време на начало;

- време на край;

- държава;

- град;

- адрес;

- заплащане за час;

- минимален рейтинг, за да може потребител да се запише за задачата;

- минимален брой изпълнени задача, за да може потребител да се запише за задачата;

* “PUT http://workit.azurewebsites.net/api/tasks” - Обновява се задачата, като е необходимо в тялото на заявката да се подадат следните параметри:

- идентификационен номер на задачата;

- заглавие;

- описание;

- време на начало;

- времетраене;

- град;

- адрес;

- заплащане за час;

* “GET http://workit.azurewebsites.net/api/tasks/{taskId}” - взимат се детайлите за една задача по идентификационния ѝ номер;
* “DELETE http://workit.azurewebsites.net/api/tasks/{taskId}” - изтрива се задача по идентификационния ѝ номер;
* “POST http://workit.azurewebsites.net/api/tasks/can-assign” - проверява се дали потребителя може да се запише за дадена задача;
* “PUT http://workit.azurewebsites.net/api/tasks/{taskId}/assigned-user” - Обновява се изпълнителя на задачта;

**3.2.3 TaskRequestController**

Пътищата на “TaskRequestController” са следните:

* “GET http://workit.azurewebsites.net/api/tasks/{taskId}/requests” - взимат се всички заявки за дадена задача, като е необходимо потребителят да бъде създател на задачата;
* “POST http://workit.azurewebsites.net/api/requests” - създава се заявка към някоя задача;
* “PUT http://workit.azurewebsites.net/api/requests” - обновява се заявка към някоя задача;
* “DELETE http://workit.azurewebsites.net/api/requests/{requestId}” - изтрива се заявка към задача;

**3.2.4 RaitingsController**

Пътищата на “RaitingsController” са следните:

* “POST http://workit.azurewebsites.net/api/raitings” - Създава се оценка за някой потребител;

**3.3 Основни методи и класове, които са заложени в Android операционната система**

За работата на приложението се използват няколко основни термина от “Android”. Те биват разгледани в точки **3.3.1** и **3.3.2.**

**3.3.1 Интерфейс**

Всички екрани в едно “Android” приложение представляват “Activity” класове. Начинът, по който изглежда на всеки един от тези екрани, се определя чрез “XML” файлове, които се намират в “/res/layout” директорията. Когато един екран се създава, той винаги извиква методът “onCreate(Bundle bundle)”, а в началото на този метод се извиква “setContentView(int resId)”, на който се подава идентификационният номер на дадения “XML” файл. На всеки един елемент (“EditText”, “TextView”) можем да задаваме идентификационен номер, чрез който можем да достъпваме отделните елементи в “Activity” класа.

**3.3.2 Преминаване между екрани**

Класът “Intent” служи точно за тази цел. Чрез него в “Android” приложението можем да преминаваме от едно “Activity” на друго, тоест от един екран на друг. За да си инициализираме един “Intent” се нуждаем от контекста на сегашното “Acitvity” и съответно класа на екрана, на който искаме да отидем.

Пример:

Intent intent = new Intent(Context currentActivity, Class destinationActivity);

За да стартираме преминаването трябва да извикаме метода “startActivity(intent).” и да подадем новосъздадения “Intent” като параметър.

**ЧЕТВЪРТА ГЛАВА**

**РЕАЛИЗАЦИЯ И РАБОТА С ПРИЛОЖЕНИЕТО**

**4.1. Изисквания и инсталация на приложението**

За работата на приложението е нужно наличието на Android устройство - смартфон или таблет с версия на операционната система 2.2 и нагоре.

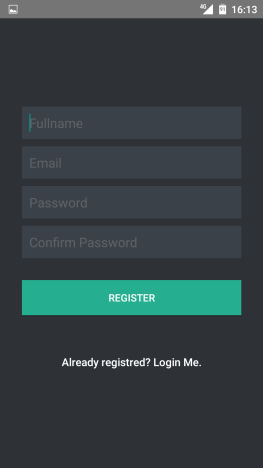
**4.1.1 Инсталация на приложението**

За инсталацията е нужен инсталационният файл на приложението (APK). За да се придобие с този файл даденият потребител трябва да го свали на телефона (фиксираната памет или SD картата), за да бъде видим от предварително качен файлов мениджър. След отварянето на APK файла, потребителят ще бъде попитан за достъп на: „Мрежова комуникация“, „Системни инструменти“, „Вашите лични данни“. Потребителят трябва да се съгласи, за да се инсталира

приложението и да натисне бутона „Инсталиране“.

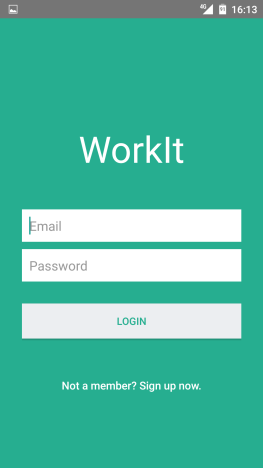
**4.2. Работа с приложението**

Внимание! За работата на приложението, е нужна връзка на устройството към интернет.

**4.2.1 Създаване на акаунт**

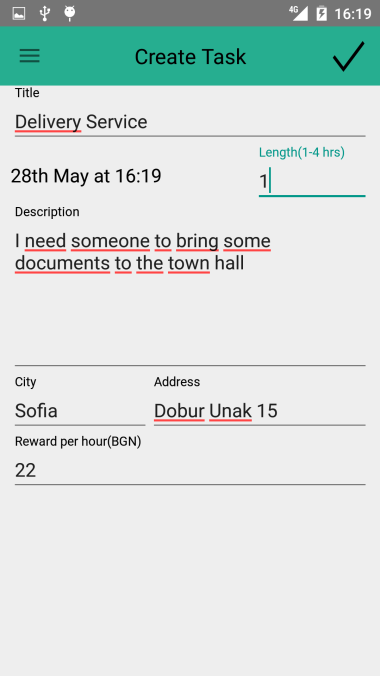
При създаване на акаунт е необходимо да се въведат всички полета, които са пълно име на потребителя, имейл и парола. Имейлът трябва да бъде уникален.

**Фиг. 2.5 Създаване на акаунт**

**4.2.2 Влизане в акаунт**

При влизане в акаунт е необходимо потребителят да въведе своя имейл и парола.

**Фиг. 2.6 Влизане в акаунт**

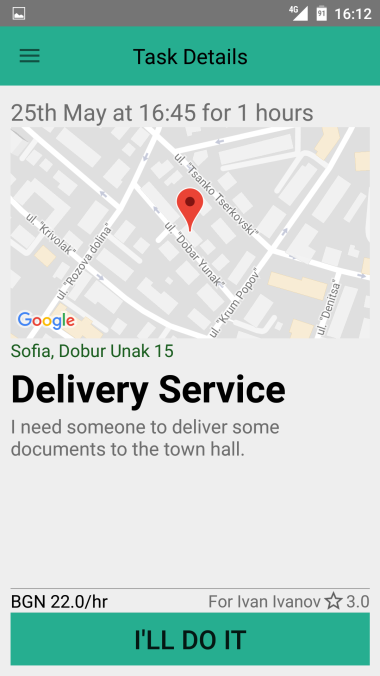
**4.2.3 Създаване на задача**

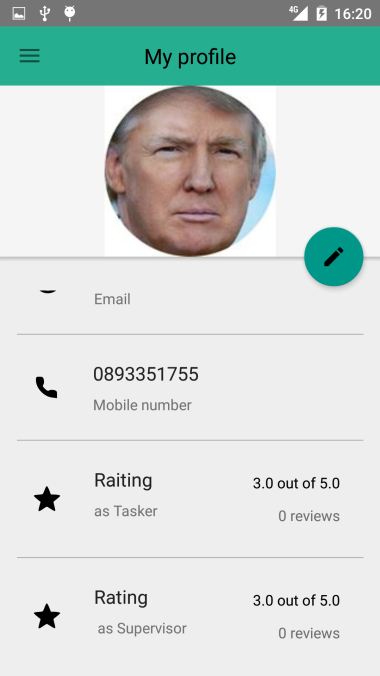
При създаване на задача е необходимо да се въведат следните данни:

* Заглавие;
* Описание;
* Дата;
* Град;
* Адрес;
* Заплащане на час;
* Минимален рейтинг;
* Минимален брой изпълнени задачи;

**Фиг. 2.7 Създаване на задача**

**4.2.4 Записване за задача**

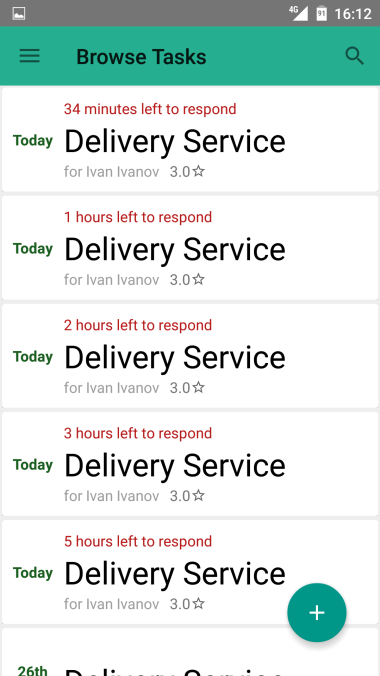
При записване потребителят вижда подробностите за задачата и може да избере дали иска да участва.

**Фиг. 2.8 Записване за задача**

**4.2.5 Профил на потребителя**

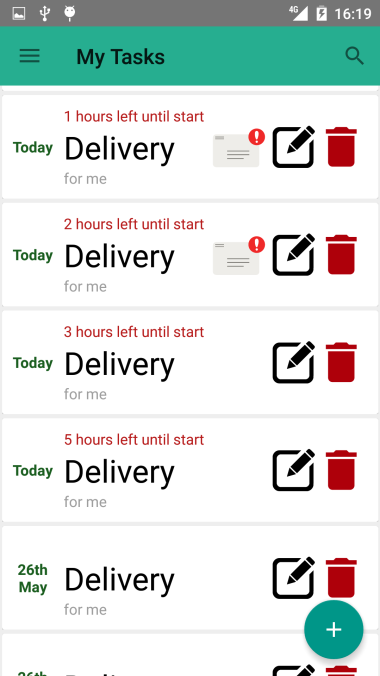
В профилната страница излизат подробности за потребителя.

**Фиг. 2.9 Профил**

**4.2.6 Списък със задачи**

В списъка със задачи можем да видим всички задачи, които предстоят и нямат изпълнител.

**Фиг. 2.10 Списък със задачи**

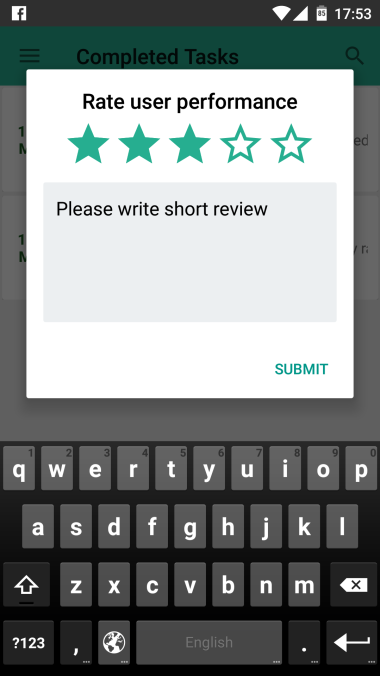


**4.2.7 Списък с моите задачи**

В списъка със задачи можем да видим всички задачи, които сме създали и не се приключили

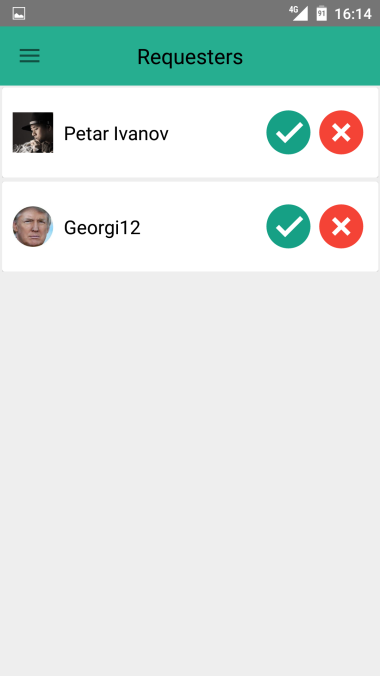
**Фиг. 2.11 Списък със**

**моите задачи**

**4.2.8 Даване на оценка**

Можем да изберем каква оценка искаме да дадем и съответно да да напишем коментар към нея.

**Фиг. 2.12 Даване на оценка**



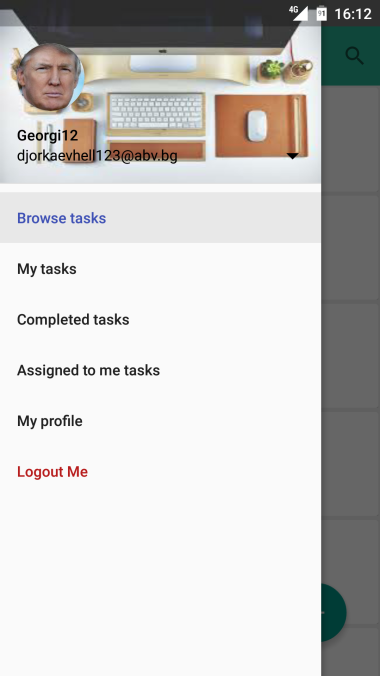
**4.2.9 Списък със заявители към задача**

В този списък можем да видим името на потребителя и профилната му снимка, както и да го приемем или отакжем

**Фиг. 2.13 Списък със**

**заявители към задача**

**4.2.10 Навигационно меню**

От навигационното меню, можем да извършим следните действия:

* Да разглеждаме задачи, които очакват някой да ги свърши;
* Да разглеждаме всички активни собствени задачи;
* Да разглеждаме всички завършени задачи;
* Да разглеждаме всички задачи, към които сме се записали;
* Да отидем на екрана с профила;
* Да излезем от акаунта;

**Фиг. 2.14 Навигационно меню**

**Заключение**

Цел на настоящата дипломна работа е разработване на Социална среда за възлагане на задачи реализирана на Android мобилно приложение. Тази цел беше постигната чрез реализиране на “Android” мобилно приложение и сървърно приложение, предоставящо “REST” програмен интерфейс.Разработеното приложението покрива всички изисквания на даденото задание, като неговата функционалност е разширена. Софтуерният продукт е пряко приложим във всекидневието на различните потребители, дори вече е достъпен в пазара за “Android” приложения – “Google Play”. В бъдещите версии на приложението се предвижда възможност за докладване на задачи и потребители за злоупотреба, както и чат между потребителите.

**Използвана литература:**

1. https://en.wikipedia.org/wiki/Wunderlist
2. https://play.google.com/store/apps/details?id=com.todoist&hl=bg
3. https://play.google.com/store/apps/details?id=com.anydo
4. https://en.wikipedia.org/wiki/Representational\_state\_transfer
5. https://en.wikipedia.org/wiki/Android\_Studio
6. https://en.wikipedia.org/wiki/Microsoft\_Visual\_Studio
7. https://www.visualstudio.com/vs/whatsnew/
8. https://en.wikipedia.org/wiki/SQL\_Server\_Management\_Studio
9. https://en.wikipedia.org/wiki/Microsoft\_Azure
10. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter
11. https://www.tutorialspoint.com/http/http\_requests