|  |
| --- |
| HelpMeOut - Сайт за взаимопомощ |
|
| *Проект по „Софтуерни технологии“, 2014 г.* |
| Факултет математика и информатика, Софийски университет |

|  |
| --- |
| Изготвили: |
| 80662, Михаил Костадинов Господинов, Компютърни науки 3ти курс 2ри поток 3та група |
| 80654, Калоян Христов Калудов, Компютърни науки 3ти курс 2ри поток 3та група |
| 80703, Христо Атанасов Аврамов, Компютърни науки 3ти курс 2ри поток 3та група |

Ръководител: доц. Димитър Биров

2014 г.

Съдържание

[1 Изисквания 5](#_Toc392508877)

[1.1 Изисквания за сигурност на системата 5](#_Toc392508878)

[1.2 Изисквания за функционалност на системата 5](#_Toc392508879)

[1.3 Изисквания за ефективност на разработката 6](#_Toc392508880)

[1.4 Изисквания за прилагане на добри практики при разработка. 6](#_Toc392508881)

[1.5 Изискване за ефективност на системата. 6](#_Toc392508882)

[1.6 Изискване за минимални характеристки на базата от данни. 7](#_Toc392508883)

[1.7 Изискване за съвместимост с браузари 7](#_Toc392508884)

[1.8 Препоръчителни изисквания за използвани технологи 7](#_Toc392508885)

[2 Увод 8](#_Toc392508886)

[3 Описание 9](#_Toc392508887)

[3.1 Общо 9](#_Toc392508888)

[3.2 Основни Концепции 10](#_Toc392508889)

[3.2.1 Потребители 11](#_Toc392508890)

[3.2.2 Въпроси 14](#_Toc392508891)

[3.2.3 Отговори 16](#_Toc392508892)

[3.2.4 Организаци 17](#_Toc392508893)

[3.2.5 Справки и статистики 19](#_Toc392508894)

[4 Функционален Дизайн 20](#_Toc392508895)

[4.1 Нива на достъп на страниците 20](#_Toc392508896)

[4.1.1 Анонимен 20](#_Toc392508897)

[4.1.2 Логнат 20](#_Toc392508898)

[4.1.3 В Организацията 20](#_Toc392508899)

[4.1.4 В Групата 20](#_Toc392508900)

[4.1.5 Модератор на група 20](#_Toc392508901)

[4.1.6 Модератор на организацията 20](#_Toc392508902)

[4.1.7 Създател на организацията 21](#_Toc392508903)

[4.1.8 Сайт модератор 21](#_Toc392508904)

[4.2 Web структура на приложението 21](#_Toc392508905)

[4.2.1 Общи елементи 21](#_Toc392508906)

[4.2.2 Homepage 23](#_Toc392508907)

[4.2.3 HelpCender/Dashboard 23](#_Toc392508908)

[4.2.4 Логин страница 28](#_Toc392508909)

[4.2.5 Регистрационна страница 29](#_Toc392508910)

[4.2.6 Профилна страница 30](#_Toc392508911)

[4.2.7 Създаване на въпрос 31](#_Toc392508912)

[4.2.8 Преглед на въпрос 33](#_Toc392508913)

[4.2.9 Чат 35](#_Toc392508914)

[4.2.10 Аудио видео разговори през webrtc 36](#_Toc392508915)

[4.2.11 Постижения 37](#_Toc392508916)

[4.2.12 Организации 39](#_Toc392508917)

[4.2.13 Списък категории 41](#_Toc392508918)

[4.2.14 Списък въпроси по категория 42](#_Toc392508919)

[4.2.15 Справки и статистики 42](#_Toc392508920)

[4.2.16 За нас 43](#_Toc392508921)

[4.2.17 Ранклиста 43](#_Toc392508922)

[4.2.18 404 Страница за грешка при ненамиране на страница 44](#_Toc392508923)

[4.3 Сигурност 45](#_Toc392508924)

[4.3.1 Cross Site Scripting defence 45](#_Toc392508925)

[4.3.2 Cross Site Request Forgery defence 45](#_Toc392508926)

[4.3.3 Captcha срещy ddos and password stealing 45](#_Toc392508927)

[4.3.4 Passwords hashing(bcrypt) 45](#_Toc392508928)

[4.3.5 SSL 46](#_Toc392508929)

[4.4 SEO Изисквания 46](#_Toc392508930)

[4.5 Use Case-ове 46](#_Toc392508931)

[4.5.1 U001 Анонимен Потребител се аутентикира пред ситемата 46](#_Toc392508932)

[4.5.2 U002 Анонимен Потребител се опитва да достъпи страница изискваща Ниво : Логнат 47](#_Toc392508933)

[4.5.3 U003 Аутентикиран Потребител задава нов въпрос 47](#_Toc392508934)

[4.5.4 U004 Аутентикиран Потребител отговаря на въпрос със съобщение 48](#_Toc392508935)

[4.5.5 U005 Аутентикиран Потребител отговаря на въпрос със чат 48](#_Toc392508936)

[4.5.6 U006 Аутентикиран Потребител менажира профила си 48](#_Toc392508937)

[4.5.7 U007 Аутентикиран Потребител създава и менажира организацията си 48](#_Toc392508938)

[4.6 Гласуване 49](#_Toc392508939)

[4.7 Изкачване в ранк 49](#_Toc392508940)

[5 Архитектура 49](#_Toc392508941)

[5.1 Физическа архитектура на системата 49](#_Toc392508942)

[5.1.1 Обобщено 50](#_Toc392508943)

[5.1.2 Схема на мрежата 50](#_Toc392508944)

[5.1.3 WEB SERVER 50](#_Toc392508945)

[5.1.4 SQLite Server 50](#_Toc392508946)

[5.1.5 ICE Server 51](#_Toc392508947)

[5.2 Софтуерна Архитектура 51](#_Toc392508948)

[5.2.1 Обобщено - Софтуерен Стек 51](#_Toc392508949)

[5.2.2 MVC 59](#_Toc392508950)

[5.2.3 Модел на базата от данни 60](#_Toc392508951)

[5.2.4 Aутентикация 64](#_Toc392508952)

[5.2.5 Архитектура за разговори 65](#_Toc392508953)

[Библиография 67](#_Toc392508954)

# Изисквания

Това са сепцифичните изисквания, на който системата трява да отговаря за да бъде одобрена като надеждна и приложима в подходящи условия за работа.

## Изисквания за сигурност на системата

Изисква се ползване на SSL. Изисква се защита от автомативзирани и полуавтоматизирани атаки. Да се имплементира защита от зловредни заявки , разчитащи на същесвтуваща сесия на потребилите.Да се създаде защита срещу всички съвременни методи за разване и крадене на пароли. Да се предовратяват своевременно всички опити за DDOS атаки към сърварите.

## Изисквания за функционалност на системата

Изисква се системата да имплементира подходящ интрефейс , за филтрирания и сортирания на зададените въпорси. Да се създаде подходящ и удобен за потребителите интерфейс за задаване на въпроси. Да има достатчно много и интуитивни начини за отговор на различни въпроси през самата система, а също така и да има и интеграции с платформи за въшна комуникация(емайл интреграции, огроври през Facebook, Google +, и други). Да се създадът максимално удобни функционалности, чрез който да се създават и да се удоволетворяват изискванията на огранизации. Основната идея е да може системата да се прилага за олесняване работата на различни учреждения и фирми. Да се въведе йерархия между потребителите на дадената организация и да се налагат информационна сигурност и нова на досъп.

## Изисквания за ефективност на разработката

Проекта да се разработи за максимално бързо време и същевременно да са приложени добри, практики и да се следи за създаването на качествен код. В случай на необходимост от промени в логиката или структурата на приложението, да се постигне максимална ефективност(минимално време) за имплементиране на промените.

## Изисквания за прилагане на добри практики при разработка.

Да се приложат изпитани практики за разработка на уеб приложения, и да се разчита на надеждни и доказали се с времето езици и технологии.Да се спазват известни в общноста конвенции за писане на качесвтен код. Да се следят и използват всички модерни и нови практики, за който общестовеното мнение на експертите в дадената технология е положително. Препоръчителни технологии са : Ruby (web frameworks: sinatra or rails), sql , (template language for html с цел по бърза и удобна разрабока)

## Изискване за ефективност на системата.

Да се постигне добра ефектифност на системата, при сортирания и фитрлирания както и при други специфични операции. Обемът на заетата сървърна памет да не надминава минималната необходима(да няма зареждане на излишни функционалности(библиотеки), който не са необходими за разработката на приложението).\*Забележка: Да се обмисли да ли използването на големи framework-oве е подходяща.

## Изискване за минимални характеристки на базата от данни.

Изисква се базата от данни да предоставя максимална скорост, и да не надвишава минималното необходимо количесво памет на сървъра. Препоръчителни технологии: Някоя разновидност на SQL, заради най-добрата и продуктивност от страна на скорост и изпитаните и качества от страна на улесняване на разработката,

## Изискване за съвместимост с браузари

Изисква се системата да се подъжа безпроблемно на всички съвременни браузари , (chrome , firefox, opera , ie8 +).Изисква се голям процент от функционалностите на приложението да работи и на браузари за мобилни телефони. Препоръчителни технологии за изпълняване на тези изксвания: javascirpt: Jquery, underscore; CSS: Twitter bootstrap, HTML: template language like ejs or haml.

## Препоръчителни изисквания за използвани технологи

Предвид изискванията към системата описани горе, тук ще бъдат бъдат изброени препоръчителни технологии, който да бъдат използвани при разработка на приложениет. В случай на избор на технологии различни от тук препоръчаните да се напише обяснение под формата на свободен текст, който да представлява аргументацията за избора на конкретната непрепоръчана технология.

Препоръчителни сърварен език: Ruby v 2.0.0 или по голяма. Да се използва framework за уеб разработка,Препоръчителни framework-ове са sinatra или rails. Да се обмислят проблемите възникващи поради големината на framework-a rails и да се прецени дали той да предостави твърде много функционалности, който ще останат неизползвани.

Препоръчителна база от данни: SQL , или някакъв нейн диалект, който би спомогнал за бързодействието на приложениет. Диалекта на езика за работа с релационната база от дани да бъде по избор на екипа разработващ приложениет.

Препоръчителни клиентски технологии: Javascript: Да се използва Jquery, поради голямата му мощност и доказаните му с времето качества, а и също така поради съвместимостта му с всички съвременни браузари.При необходимост от допълнителни фунционалности да се използва underscore . CSS: да се използва Twitter bootstrap поради съвместимостта му с всички съвременни браузари и доказаните му качества в разработката на уеб приложения.HTML: Да се използва темплейтен език като haml или ejs за да се ускори процеса на разработка, катко и качесвото на кода.

Препоръчителни технологии за имплементация на комуникациите в реално време: Да се използва WEBrtc поради бързото му навлизани и широкото му разпорстраниение на пазара.Друг аргумент за използването на тази технология е, че за използването и е необходим само javascript и също така е съвместима с библиотеките, който са препоръчани в тази секция.Като сървар за комуникациите в реално време да се използва ICE Server, поради лесната му настройка, сравнително ниска цена и добрите фунцкионалност, който предоставя.

# Увод

Проектът , който е описан в този документ, представлява уеб приложение за взаймопомощ между потребители, които имат належащи проблеми(такива който трябва да бъдат решени в определен срок като например „утре имам изпит по математика помогнете с тази задача”).Приложението е пригодено за използване и от фирми и огранизации и предоствя и съответните нива на сигурност и конфиденциялност.

В този документ се разглежда проектът от концептуална , функционална и архитектурна гледна точка.

# Описание

## Общо

Първата важна част от проекта, се състой в лесен за потребителите интерфейс , чрез който да могат да представят добре интересите , уменията и темите по който биха имали нужда от помощ и в които биха могли да помагат. За целта има обособени създадени от админите категории(като например математика, биология, соцялни умения, медицина и други) от който всеки потребител може да избере, а също така може да се създават нови категории, след одобрение от администратор. За да може всеки да разбере колко достоверен е отговора, получен от даден човек , уменията на всеки потребител се измерват в нива. Това означава, че когато някои отговаря на много въпроси от раздел математика, и хората оценяват отговорите му като добри (чрез скала от 0 до 5 звезди), то неговото ниво на умение математика ще нарастне и другите потребители ще видят , че отговорите му по тази тема вероятно са достоверни.

Втора част от проекта е инерфейса, за създаване и отговаряне на въпрос.При създаване на въпрос има две опций с максимално опростен дизайн. Първата е създаване на въпрос изискващ отворен отговор. Описва се въпроса, задава му се сфера и се задава крайния час, в който е необходимо да бъде получен отговор. След това въпроса се изпраща, и всеки потребител , може да го види , заедно с таймер който отброява, колко време остава до крайния срок за отговор. След изтичане на крайния срок въпросът изчезва и товя позволява на потребителите да виждат само акутални въпроси, на които все още не е било отговорено.

Втората опция е да се създаде въпрос изискващ отговор с да или не. Потребителя описва въпроса, задава му сфера и го изпращя. В последствие всеки може да види въпроса и да избере от опциите да и не, и да види диаграма на отговорите на другите потребители. За да може да се отговарят на въпроси, има страница, в която потребителите да сортират въпросите по тяхно желание. Тази страница включва възможности за сортирания търсене по категория, филтриране по свободен техт, търсене на въпроси от определен/и потребител/и както и всички комбинации от горе описаните. Когато потребител избере да отговори на въпрос с да и не интефейса е тривиялен( два бутноа съответно за да и не). Ако потребител избере да отговори на отворен въпрос има много възможност. Отговора може да бъде даден под формата на коментар към въпроса, чрез лично съобщение (под формата на чат) , чрез емайл , или чрез видео или аудио разговор.

Третата важна част от проекта са допълнителната стимулация да бъде отговаряно на чужди въпроси. Рейтинг система ще показва уменията на отговарящите и ще ги поставя в дневни, седмични и месечни класации, като за първите в месечните класации ще има предвидени допълнителни стимулации( грамоти , награди под различна фрома и други) .Система за предизвикателста ще стимулира потребителите да отговрят на въпроси от различни сфери, или на по голям брой въпоси.(Например предизвикателсто: „Отговорете на въпоси от 5 разлчни сфери). Освен това всеки потребител ще може да получи подробна справка, за отговорите който е давал , въпросите който е задавал и оценките който е получавал . Такава справка би могло да бъде използвана, като представителна извадка дори при кандидатсване за работа.

Четвъртата важна част от проекта е възможността за създване на организации и техни потребители, който да имат възможността да употребяват система за да им спомага в работня процес. Всяка огранизация има публичен акунт, с който може да прави изяви пред света на HelpMeOut, а също така да създава и амдинистрира свои затворени страници и групи с подходящо членове и права на видимост.

## Основни Концепции

HelpMeOut е система за взаимопомощ която свежда решаваното на проблем за ограничено количество време до задаване на въпрос и получаване на различни по тип отговори. Потребителите задават и категоризират въпросите си по различни области , и дават отговори в категориите с които са запознати.

Основна концепция на системата е възможността за множествеността на методите за отговор на въпроси , както и оценяването на тези отговори , “gamification” на системата чрез постижения , получаване на опит и изкачване в ранк. При заплащане потребител може да създаде организация в която той , и администраторите които е задал да могат да пресъздадат йерархията на корпорацията му. Потребителите в организация могат да бъдат организирани в групи , чрез които да могат да работят заедно върху въпроси.

### Потребители

Като почти всяката система и HelpMeOut има система за потребители, които са централна част от логиката на цялото приложение(както за създаването му и хората който са го създали, така и за използването му и всеки човек, който го намира помощ в него).

Поребителите, ще могат да влизат в системата през стандартна логин фрома. За създаването на потербител ще могат да се използват , данните от социялните мрежи като фейсбук, туитар, гоогъл плюс, но ще трябва и да посочат и някой допълнителни данни, като уменията който притежават например. По подробно върху приложенията, същността и употребата на тези умения и интереси, ще се спрем в следващоте точки (1.2.1.1 Умения и 1.2.1.2 Интереси).Естествено всеки потребител ще може да си направи, независима от каквато и д а била соцялна платформа(за тези който ,са зле настроени към предоставянето на прекалено много информация в интернет е нужно само да въведът потребителско име, с което другите потребители да ги виждат и парола, с която да влизат в системата).В следващите под глави ще се спрем върху основните концепции свързани с потребителите и пълноценната им употреба от всички.

#### Умения

Уменията на един потребител изразяват сферите, в който той смята че има познания, и също така сферите в който другите потребители на системата са оценили отговорите му като полезни. Още при регистрация потребителят посочва такива умения(области), в който смята, че има познания както и точното ниво на познания според неговата преценка.В последствие, нивата на тези умения, който е посочил потребителя могат да се променят в положителна или отрицателна посока в зависимост от това какви оценки са били поставени на отговорите му от останалите потребители .Разбира се, всеки потребител ще може да се развива и в умения, който не е посочил при регистрацията си.Това ще става съвсем просто: При отговаряне на въпрос от даден потребител, ако отговорът му е отбелязан като полезен, отговорилияя получава точки в дандено ниво на умения и така подобрява личните си нива на умения, или придобива нови(например при първо отговаряне на въпрос от категория, по която до сега даден потребител няма положително оценени отговори, и нейното положително оценяване, той получава първо ниво от даденото умение).

При твърде ниско оценени отоговори , потребителя може да намали нивото си на умение в дадена категория, или дори да го загуби. Това е мярка за да се предовратят, хора който злонамерено да дават грешни отговори, или просто да се подготвят неподготвени по някоя тема.

#### Интереси

Интересите на потребител са типовете или категориите въпроси които той следи (от които той се интресува) , това е начин да филтрира въпросите които иска да види , или на които той иска да отговаря. Разликата между интереси и умения може на пръв поглед да изглежда тънка, но всъщност концепциите на двете неща са съвсем различни.

Уменята са нещатата, който потребителя умее, а интересите тези , който са интересни, за него като това може да са и такива, в който той няма никакъв опит. Интересите на даден потребител се задават при регистрацията му, но естествено могат да се променят и след това за да може, модела на HelpMeOut да е максимално близък и интуитивен за потребителите.

#### Опит

Основна концепция на системата е “gamification”-a. Задаването и отговарянето на въпроси се стимулират чрез “bar-filling” и “leveling”. При оценяването на въпрос потребителя получава “опит”. Опитът на потребител е няколко вида : Опит в дадена област , тоест до колко той владее дадено умение , които зависят от отговорите му по въпроси от тази категория , и личен опит на база на всички негови умения, по-високия опит на потребител му дава по-големи привилегии.

#### Ранк

При надминаване на дадена граница от опит той придобива по-висок ранк в системата към който са свъзани привилегии , най-високите ранкове потребители служат за своеобразни избрани от общността модератори. Друго предимство на високия, ранк е че потребителите с най-високи постижения, във всяка област ще получават при желание, справка за всички въпроси на който са отговорени успешно и грамоти гарантирани от HelpMeOut, за практически познания в дадена облат(реално тези сертификати няма да имат фомрална стойност, но биха могли да бъдат като повод за гордост между приятели така и добра препоръка за някои работодатели).

#### Постижения

Постиженията са още един аспект на “gamification”-a на системата , те представят цели , някои от които трудно постижими , други почти невъзможни , към които потребителите да се стремят . Пример : Потребител гласува отрицателно за отговор който вече има отрицателен рейтинг и получава постижение “Salt in the wound - Downvote an already negatively rated answer”. Всяко постижение, ще носи дадена награда, като не всички постижения ще носят позивитни награди, отново за да се ограничат злонамерени отговори. Наградите ще бъдат от получавате(може и отрицателен) бонус към ранга си до печелите тениска от HelpMeOut ( за постижения, като бъдете с най-много добре отговорени въпорси в дадена категория)

### Въпроси

Въпросите са основния аспект и функционалност на HelpMeOut . Всички въпроси имат краен живот(това е и най-важната логическа концепция на един въпрос) , както и са в поне една категория , така че да могат да бъдат откривани от потребители. Въпросите са винаги подредени по спешност - оставащо време до разрешаването им. След изтичането на един въпрос или отбелязването му от създателя като отговарен, той бива скрит и повече неможе да бъде видян от никой, освен при създаването на доклади за отговорите на даден потребител, посикан лично от него. Ситуацията при въпросите зададени в рамките на организации и тяхните публични и скрити страници е по различна и ще бъде подробно описана в графата Организации.

#### Time-out на въпросите

Всеки въпрос живее определен , краен период от време , зададен от потребителя и ограничен до най-много две седмици преди да бъде скрит. Въпроса може да бъде затворен по-рано ако бъде маркиран като отговорен от задаващия го.Тук се намира и най-уникалната част от HelpMeOut. Въпросите винаги са подредени по оставащото време до изтичане.Разбира се всеки потребител може да филтрира въпросите, които вижда по категории, интереси ,време , стрингове, и други, а също да предприеме вторични сортирания по имена , задаващи и други, но основното сортиране по време никога неможе да бъде нарошено.Това е така защотото основната концепция на HelpMeOut е да се помага на хората в непосредствена нужда,възможно най-бързо и от възможно най-много хора(тоест вероятността да получиш желания отговор нараства с намаляването на времето оставащо ти до изчезване на въпроса)

#### Видове Въпроси

Съществуват няколко вида въпроси , на база на време или на ограничение по тип отговор. Те са следните:

* + - * 1. Yes/No Въпроси

Това е въпрос на който може да се отговори само с да/не . Отговорите на тези въпроси получават оценяване на база на на това дали съвпадат с отговора одобрен от задаващия го потребител. Например имаме въпрос :

“Числата на Кармайкъл , безкрайно много ли са ?” , 70% от отговорите са “Да” , другите 30% “Не” , задалия въпроса одобрява “Да” , и всички отговорили по този начин получават опит.Тук идеята, е да се предостави на потребителя знания на малки порции, с който тои да се забвалява, да трупа опит, или да повиши общатата си култура в дадена сфера.

* + - * 1. Нормални Въпроси

Нормалния въпрос е въпрос , който може да бъде с по голям обем, и на който може да бъде отговорено посредством всички начини за комуникация предоставени от HelpMeOut. За да е в категория нормален оставащото време на въпорса трябва да бъде над един ден.

* + - * 1. Спешни Въпроси

Спешен въпрос е нормален въпрос , който има оставащо време за отговор под един ден . Според философията на HelpMeOut, колкото по спешен е въпроса (колкото по малко време остава до изтичането му ), толкова по-вероятно е да му бъде отговорено, защотото системата се грижи да усигори, зависимост между спешността на въпроса и броя потребители, от дадената категория който ще го видят.

### Отговори

Зададените въпроси могат да получават отговори чрез всички начини за комуникация предоставени от HelpMeOut . Отговорите могат да бъдат оценявани от други потребители, като всеки отговор има различена стойност при оценяване.Даден потребител може да оценява един въпрос само един път, като за да е максимално отоговарящо на човешките разбирания, високата оценка на отговор е по ценена ако е дошла от по уважаван потребител(такъв с по висок рейтинг).

#### Оценяване

Весеки отговор може да получи положителен или отрицателен рейтинг , което съответно увеличава или намаля личния опита на отговарящия потребител както и опита му в дадена област.Оценяването може да бъде не само “like” и “dislike” но и от едно до пет или друга стойност при някои видове отговори. Оценката ще се цени повече ако е от потребители с по голям рейтинг.

#### Видове Отговори

Възможни са множество типове отговори , всеки носещ различно количество точки при оценяване , на база на усилието нужно за това.

* + - * 1. Директен / Бърз

Отговор подобен на пост във форум . Не се изисква задалия въпроса да е онлайн за да може да види отговора. Този тип отговори носи добро количество точки тъй като изисква време да се форматира и да се отговори чрез него.

* + - * 1. Чат

Директен чат-руум , който задалия въпрос може да отвори за да проведе по-бърз разговор. Всяко съобщение от чата може да бъде оценено , като носи нисък брой точки.

Съобщения които съдържат линкове , или клипчета от youtube се изнасят от страни , за по-голямо удобство на потребителя , и могат да бъдат оценявани отделно. Въпреки че, тук оценката на всяко съобщение носи малък брои точки , това е един от най-добрите начини да се спечелят такива, защото при ползотворна информация потребителят може да даде, максимум точки на всички съобщения, а също и бонус точки, при желание след края на чата.

Агрегатор на линкове

Съдържа агрегирани линкове и клипчета от чата. Тъй като при разговор това което всъщност интресува потребителя са ресурси с които да разреши проблема си , това предоставя лесен интерфейс за работа с изпратени от помагащата страна ресурси.

* + - * 1. Аудио/Видео Отговор

Директен разговор между задаващия въпрос и потребител който иска да отговори на него. Това е най-високо оцененият тип отговор защото изисква най-високо количество усилие от страна на отговарящия. Задалият въпроса оценява разговора след края на коренспонденцията.

### Организаци

Потребители след договаряне с администратор, могат да създават организации с обособена йерарархия в тях, и широк набор от интерфейси за администриране и използване на групата.Всяка притежава акаунт на групата, който освен за администриране, може да се използва и като потребителски акаунт за отговаряне на въпроси и други. Това ще даде възможност, дадени администрации, да трупат авторитет в HelpMeOut, помагайки на обикновените потребители.

#### Създаване и Администриране

Създаването на организация ще бъде посредством помоща на администратор на системата, за да може да бъде максимално добре и интуитовно представена йерархията в нея. Тук потребителите ще имат шефове и подчинени, а организацията ще има свой групи и затворени страници.

#### Групи в организацията

Оснвна част от организациите са групите. Група е концепция от организацията в която могат да бъдат добавени потребители, който имат отношение към нея. Една група може да бъде публична или не(тоест всеки или само определени потребител да може да задава въпроси в нея) .Същто така за максимално удобство ще може да бъде ограничавано, кой може да отговаря на въпроси в дадена група и по какъв начин.Това ще позволява създаването на страница за въпроси от потребители/клиенти на дадена огранизация, например страница на ФМИ, в която всички студенти да могат да задават въпроси, но само преподавателите да отговарят.

#### Затворени страници за взаимопомощ

Затворените страници позволяват на организациите да ограничат достъпа, до информацията си, не само от случайните потребители, но и от някой от членовете на организацията , които нямат отношение към контректния проблем зададен в затрворената страница за взаймопомощ.Една организация, може да има много затворени страници за взаимопомощ съдържащи, потребители от и извън оргазнизацията по избор.

#### Йерархии в Организации

За да може HelpMeOut да обслужва максимално добре нуждите на организациите, има въведена строга йерархия между потребителите и. Всеки потребител може да получва задачи(под формата на въпроси във съответствие със концепцията за въпрос на HelpMeOut )от горе стоящите в нея и да задава такива на подчинените си. Това максимално ще улесни потребителите и комуникацията вътрешно за огранизацията.

#### Специални функционалности на организациите

Спецялните функционалности за организациите, ще са пазене на хронология с задавани въпроси и отговори, възможности за експортиране на доклади и справки, катко и имейл интеграции, а също интеграции със API-ита на соцялните мрежи.(някой от тези функционалности ще са достъпни и за обикновенните потребители)

#### Потребителски акаунт на организацията

Всяка организация има потребителски акаънт , с които може да помага на потребителите в системата, както и да задава въпроси, на който те да отговарят.Това ще позволи на всяка организация, да гради имидж в HelpMeOut, и ще и позволява да следи общественото мнение и належащо проблеми във дадена сфера.

### Справки и статистики

HelpMeOut подържа различни функционалнности за справки и статистики за всички организации и за специално пожелалите за това потребители.

#### Съставяне на доклади и статистики

HelpMeOut има удобен интерфейс за съставяне на репорти, по желание на потребителя. За да бъде постигнато максимално удобство потребителите могат да запазват шаблони за доклади, които да преизползват многократно.

#### Експортиране в ексел и уорд формат

Всички видове доклади могат да бъдат представени в табличен(excel) формат, както и в pdf format.

#### Експортиране в XML и JSON формат

Всички видоде доклади могат да бъдат представени в json и XML формати за да бъде възможно интегрирането на различни програми със HelpMeOut .

# Функционален Дизайн

## Нива на достъп на страниците

Това са нивата на аутентикация нужни за достъпването на определена уеб страница във системата

### Анонимен

Потребител който не е влезнал в системата и не е аутентикиран пред нея. Това е най-ниското ниво на ауторизация.

### Логнат

Потребител влезнал в системата със своите детайли.

#### Логнат потребител със висок ранк в ситемата

Това е логнат потребител който със действията си , си е осигурил висок ранк във системата посредством опит и знания. Той има някои допълнителни привилегии пред логнатия потребител.

### В Организацията

Потребител който е член на контекстната организация.

### В Групата

Потребител който е член на контекстната група от организация.

### Модератор на група

Потребител с елевирани права в контекстната група от организацията. Той има модераторски права над страниците на групата.

### Модератор на организацията

Потребител с елевирани права в контекстната организация. Той има модераторски права над страниците на групите от организацията , както и на страниците на организацията , като единствено е ограничен в премахването на права на потребители от организацията , премахване на потребители от организацията и разтрогването на организацията.

### Създател на организацията

Има пълни права над контекстната организация.

### Сайт модератор

Има пълни права навсякъде , може да създава организации.

## Web структура на приложението

Това е описание на това как потребителя може да взаимодейства с приложението , така че да осъществява целите си.

### Общи елементи

Елементи които присъстват на всяка страница от системата.

#### Главна навигация

Главната навигация съдържа най-важните и най-ползваните линкове . Лист от линковете в навигацията за всички потребители. Показва различни резултати на база нивото на достъп.

**fRG1OKLOmmxkwMmgBTgmEjlZUGnhzij4lgNrfXxXzxjbpEmAsV_VtOFNvvn9c-EASpF4yUJ72WZZ82i6JBTcikhqXASEwPxA7fb9sKmrTvCgmi___pnOHc-YqOvJZtfrEwФиг 1**. *Изглед на главната навигация*

Вариации на главната навигация на база ниво на достъп на потребителя, под формата на страници които той вижда, всяко ниво съдържа предшестващите.

**Ниво Анонимен**

* Начална Страница
* Страница За Нас

**Ниво Логнат**

* Въпроси
* Ранклиста

**Ниво Модератор**

* Администриране на категории
* Администриране на въпроси и отговори

#### Лого

Лого на системата. При натискане води към заглавната страница.

**sJf1FmaMQYFfgPKH2PDbuTjDJikbKtPJ_4pzFWh7Oz2gdp5y9zSsNmqM9y8S7wkkWB6EY3j_w8z7XmUe8-hz0RP0Z0e1I7YuBLdsg0PmMl43xzoA7n7OyvfFvz5i3Uq5yAФиг 2**. *Лого на ситемата*

#### Логин Контрола

Логин контролът предоставя възможност за достъп до личните страници на потребителя , организацията или групата му. Тази контрола се променя в зависимост от нивото на достъп на потребителя.

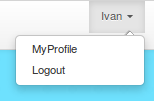
**Ниво : Анонимен**

Предоставя линк към логин страницата.

https://lh4.googleusercontent.com/d7UW_7TeioD4hbRgmY4_oIaTL37zc0BgTE_hFsdMObNPer8SOHzaQaMxosLm7ucZ-UNWhV_48YWYNEdNsqC7o5v0yyuUQBH57LpbOwGB1R8Fb6KdEan95cRWlLFguyOFxQ**Фиг 3*.*** *Визия на логин контролата при достъп на анонимен потребител*

**Ниво : Логнат**

Предоставя линк към личния профил на потребителя , както и възможност да се разлогне.

**Фиг 4**. *Визия на логин контролата при достъп на логнат потребител*

**Ниво : В организация и Ниво : В група**

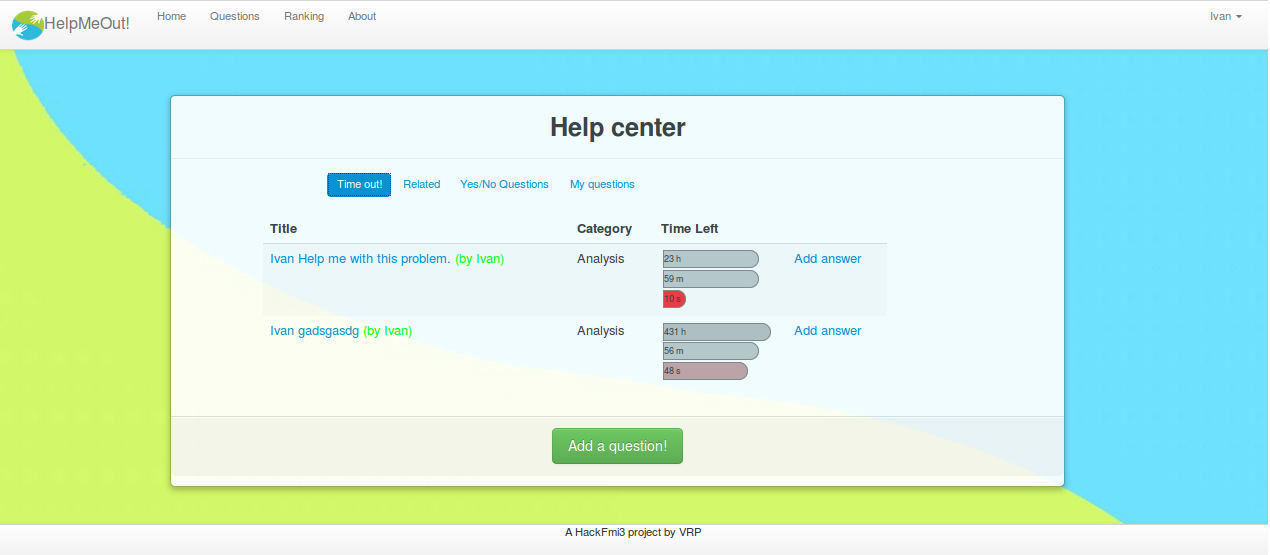
Към линковете на ниво логнат се добавят линк към страницата на организацията. Ако потребителя е в група от организацията , също вижда линк към страницата на групата.

### Homepage

Това е началната страница. Тя е главно маркетинг ориентирана и има за цел да убеди потребител да се регистрира чрез красива визия. на нея има единствен линк към регистрационната страница освен стандартните елементи.

### HelpCender/Dashboard

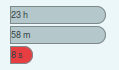
Това е началната страница за всеки логнат потребител. Тук той може да види всички въпроси сортирани по време на изтичане.

**Фиг 5.** *Изглед на дашборд страницата*

Ако е избран Time out табът или тези , който са свързано с неговите умения(сортирани първо по време, и в последствие по ниво на уменията на потребителя и категории на въпросите т.е по нагоре излизат въпросите, в чиито категрии уменията на потребителя са по високи.), ако е избрал табът Related . Табът Yes/No Questions показва на потребителя да/не въпросите, а табът My questions, показва всичките въпорси зададено от потребителя отново сортирани по време на изтичане(тук се показват дори и изтеклите въпроси, за да може потребителя да си припомни или прегледа отново някои от дадените му отговори).

#### Таймер на релевантни въпроси

Това е динамичния таймер , чрез който се привлича вниманието на потребителите върху спешността на въпроса(всяка от диаграмините намалява в реално време при изтичането съответно на секунда, минута или час.)

**Фиг 6.***Таймер показващ време по малко от ден*

Точното време се изчислява като се изпраща от сървара времето за изтичане на въпроса и часовата зона, в която е зададен и посредством клиентски javascript се пресмята правилно оставащото време от гледната точка на потребителя гледащ въпроса.

#### Сортирания

Сортиранията в тази страница стават чрез натискане на дадена колона в таблицата с въпросите. Всяко сортиране тук е вторично. Това означава, че винаги най-спешните въпроси ще останат най-отгоре.

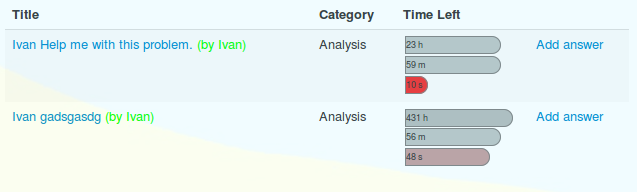
**https://lh6.googleusercontent.com/ONxshZyQoFfQquCP7kVoOZbhZ7hS7nzELwqJ8Posui6FUmAtFO9mM7T_Nk3qXxY6MqBRet_dt8MfdXQxnyzn-pR959UJzp9PYZEru5wBAhl_eoXw31q_QbC41PEYZMgGOQФиг 7.** *Пример за категории по които е възможно сортиране.*

#### Филтрирания

Филтрирането става по класическия начин чрез въвеждане на текст, в поле, ограничения на оставащо време,или фитриране по брой отговори,категория и други.Удобен инерфейс позвлява на потребителите да избират категории,от които да виждат или не въпроси , а също така и възможност за записване на предпочитанията им за филтриране за постоянно.

* + - * 1. Time Out!

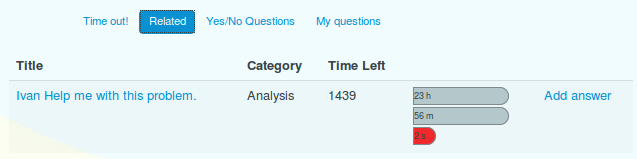
Time Out е графата, където потребителя ще може да вижда всички въпроси след прилагането на филтърът ,който е избрал(ако има такъв).

**Фиг 8.** *Визуално представяне на графата Time out!*  


В колоната title се намира заглавието на въпроса следвано от потребителя, който го е задал в зелено, следващата колона показва категорията на въпроса.Таймерът вече бе разгледан в по-горната част на този документ. Бутоннът “Add Answer” пренасочва потребителя към страницата за отговаряне на въпрос, която ще бъде разгледана по-нататък в този документ.

* + - * 1. Related

Related е графата, в която потребителя вижда въпроси, от сферите, по които има умения.

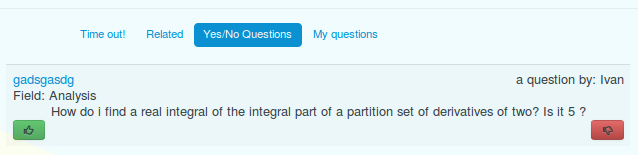


**Фиг 9**.*Визуално представяне на графата Related*

Тук филртри не се прилагат,освен ако потребителят не пожелае изрично това. Структурата за презентиране на въпрос е същата като в предната графа.

* + - * 1. Yes/No

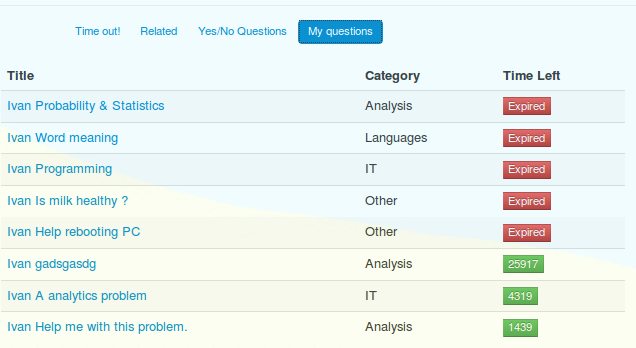
Yes/no Questions е графата , в която потребителите могат да разгледат въпросите с отговори да и не и да отговорят на такива.

**Фиг 10.***Визуално представяне на графата Yes/no Questions*

Тук също може да се дефинират филтрирания подобно на предните графи. За разлика от предните две графи тук цялото съдържание на въпроса е показано за потребителя ,за да може той да отговори посредством зеленото копче в дясно на екрана за да и червеното копче в ляво за не. След отговаряне потребителя вижда диаграма на отгоровите в проценти, и ако е отговорил със според мнозинството правилния отговор получава опит и реитинг в категорията на въпроса.

* + - * 1. My Questions

В графата My Questions всеки потребител може да види въпросите, които е задавал и да гледа отговорите,които е получил.

**Фиг 11.** *Визуално представяне на графата My Questions*

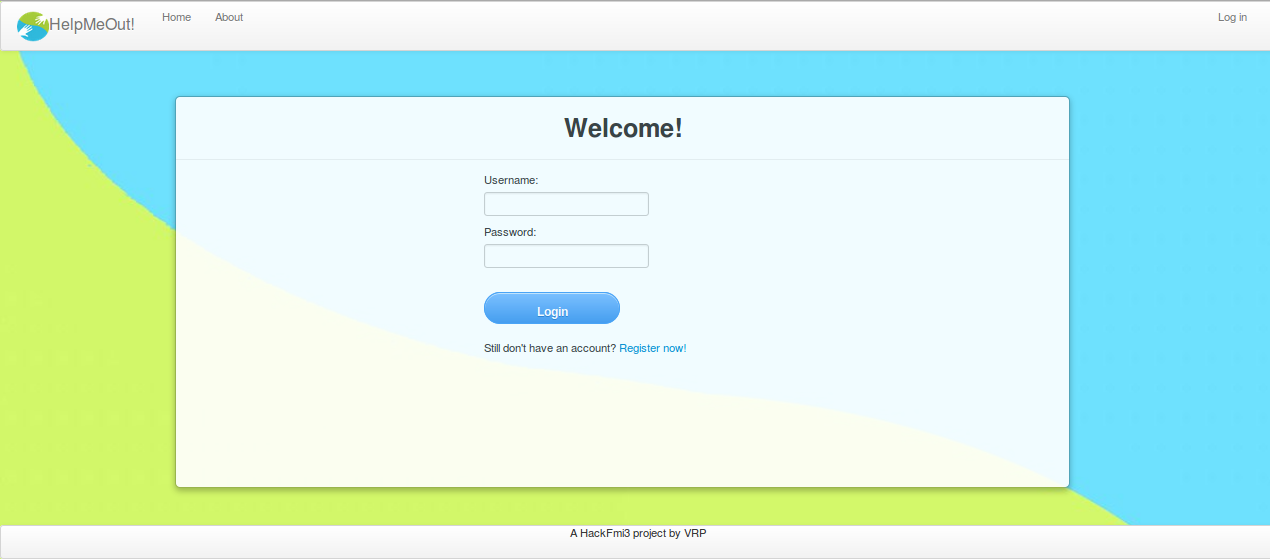
От тази страница потребителите също могат да премахват или отбелязяват като отговорени някои от въпросите си. Всеки ред в таблицата е връзка към интерфейса за разглеждане на отговори към даден въпрос.

#### Любими теми/юзери

Всеки потребител може да отбележи една или няколко теми като свой любими , и по този начин навсякаде, където вижда въпроси тези категории ще бъдат с приоритет. (тоест ако 2 въпроса имат приблизително еднакво време този, който е от любимата категория ще бъде по напред). Идентична е и ситуацията с избиране на любими потребители.

### Логин страница

Това е страница със логин форма , която предоставя на потребителя начин да влезне в системата. При грешно въведена комбинация от име и парола , потребителя трябва да бъде уведомен за грешката си чрез съобщение “Wrong username or password”.

**Фиг 12**. *Визуално представяне на логин страницата*

При въвеждане на паролата тя трябва да не се показва и да бъде заменена от други символи които да скрият реалния и текст при въвеждане. Username и Password са задължителни полета.

При голям брой грешни опити за достъп на потребителя се представя Google Captcha за да докаже че е човешко същесто , а не bot който crawl-ва страницата и се опитва да налучка валидна комбинация от потребител и парола.

Освен логин формата, за нерегистрираните потребители е предоставена възможност да се присъединят към системата чрез линка “Register now!” който води до страницата за регистрация

### Регистрационна страница

Регистрационната страница служи за присъединяването на нови потребители в системата. Не е забранено на потребителите да създават множество регистрации.

Регистрационната страница Има следните задължителни полета:

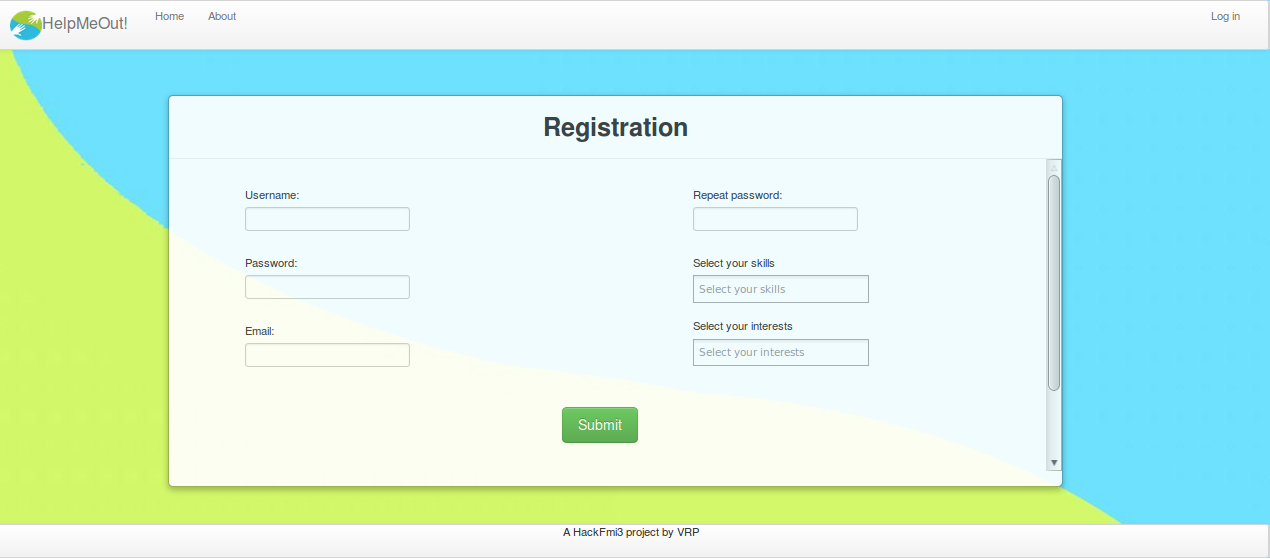
* Име - потребителско име
* Парола - парола за достъп
* Повторение на паролата - проверка за това дали паролата е написана правилно
* Еmail - електронна поща на потребителя

При разминаване на “Парола” и “Повторение на парола” на потребителя се предоставя съобщение за грешка което му казва че двете пароли се различават.

При Email несъoтветсващ на изискваниятя на RFC 2822 [RFCEM] му се предоставя съобщение че е предоставил невалиден адрес за електронна поща.

На регистрационната форма има също и набор от полета които не са задължителни за въвеждане и могат да бъдат въведени от профила на потребителя. Те са :

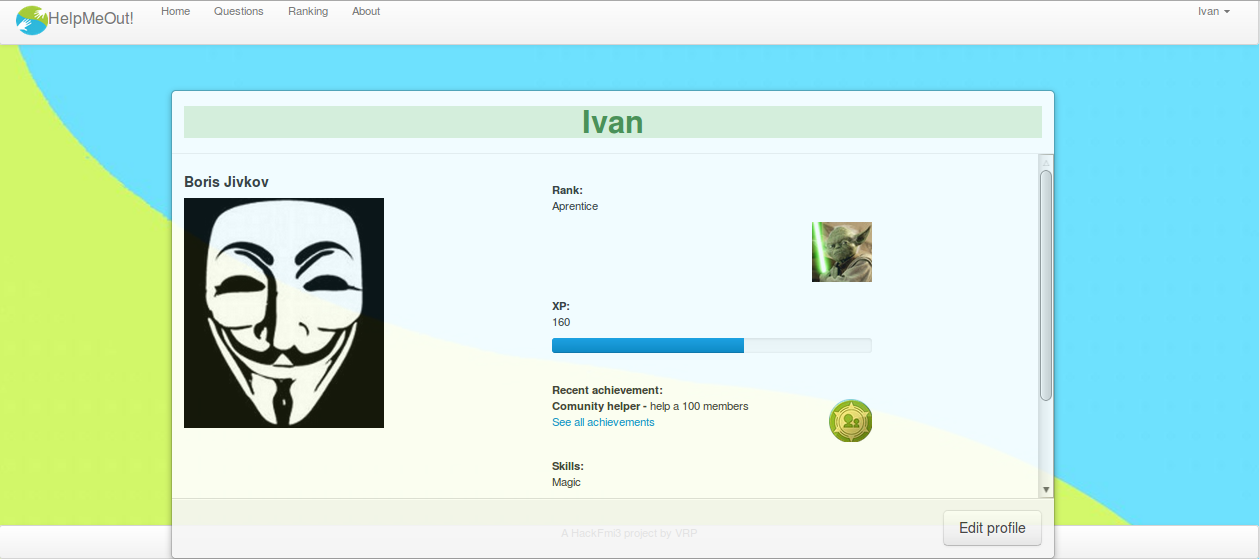
* Избор на умения - умения които потребителя твърди че притежава , при избор тук те му се добавят като най-нисък ранк.
* Избор на интереси - умения от които потребителя се интресува и иска да следи.

**Фиг 13**. *Визуално представяне на регистрационната страница*

Ако потребителя се опита да се регистрира с вече съществуващо или заето потребителско име , трябва да му се представи грешка , че потребителското име вече е заето и регистрацията да се провали.

### Профилна страница

Профилната страница представлява място от което може всеки потребител на системата да разгледа данните за профила си и да промени тези над които има права.

**Фиг 14.** *Снимка на профилната страница на потребител с име Ivan*

#### Левел(Ниво)

Това е мястото, където потребителя може да получи информация за текущото си ниво, а също така и за оставащия до следващото ниво опит.

#### Ранг

Това е мястото, където потребителя може да види ранга си както и натискаики подробности да види ранговете на всяко едно умение владяно от него. Също така от бутона подробности ще може да види и страницата с класации по рангове.

#### Опит

Опитът на даден потребилет е графично изобразен в графиката с заглавие XP. Там съвсем интуитивно може да се види колко остава до следващотото ниво.

#### Постижения

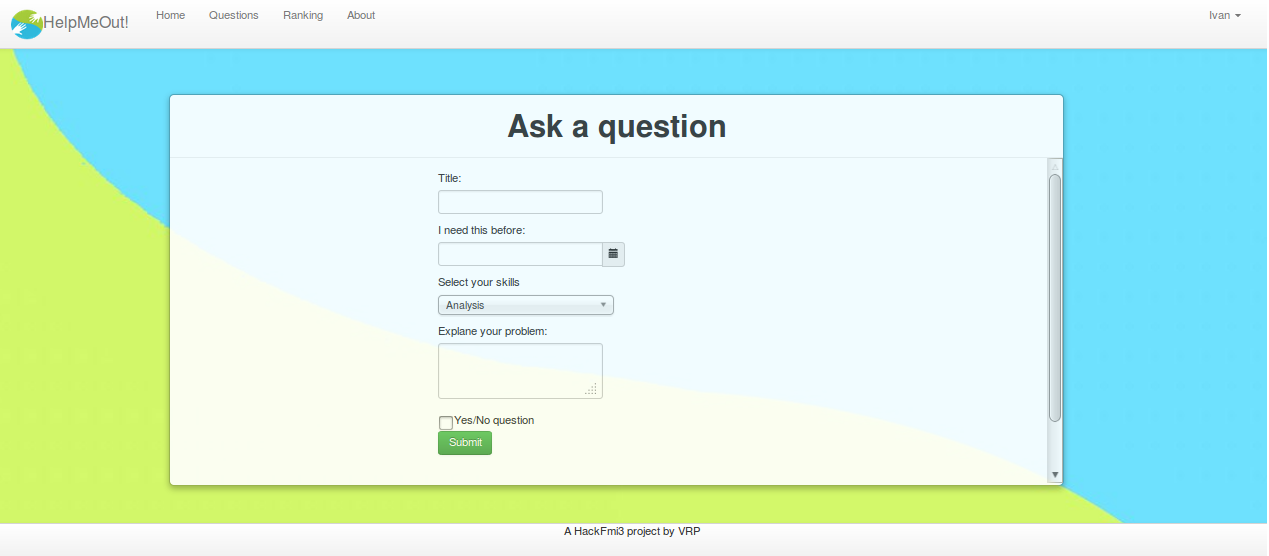
Потребителя вижда емблемите(идеята тук е да изглеждат като спечелени медали) на всички взети от него постижения. Всяка емблема е връзка към подробно описание на контретното постижение, както и история за това кога, как и защо потребителя е взел това постижение.

#### Промяна на данните в профила

Чрез копчето Edit в долния ляв ъгъл, потребителите ще могат да променят иформацията си като името с което да се виждат пред останалите потребители, профилната си снимка и личната си иформация.

### Създаване на въпрос

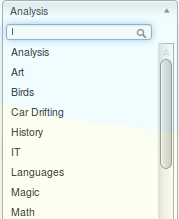
Това е една от основните странцити във HelpMeOut , защото всеки един потребител които поиска помощ, ще трябва да мине през, нея.

**Фиг 15.***Изображението показва формата за създаване на нов въпрос*

В полето Тitle се въвежда заглавието на въпроса ,което ще се вижда с уоголемен шрифт.Долното поле позволява посредством удобен интерфейс да се избере дата и час на изтичане на въпроса. След това потребителя избира категория за въпроса и в последствие, описва същността му.

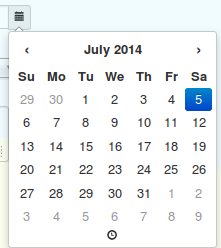
#### Контрола за избор на умения

Потребителския интерфейс за избор на категория предоставя удобно търсене и филтриране по подстринг,което е максимално опростено , такаче да бъде интуитивно за всеки потребител.

**Фиг 16.***Потребителки интерфейс за избор на категория*

#### Date-Time picker контрола

Този интерфейс за избор на дата и час предоставя, лесен начин за избор ,предоставяйки календар и часовник(при натискането на часовничето долу на картинката).

**Фиг 17.** *Потребителски интерфейс за избор на дата и час.*

### Преглед на въпрос

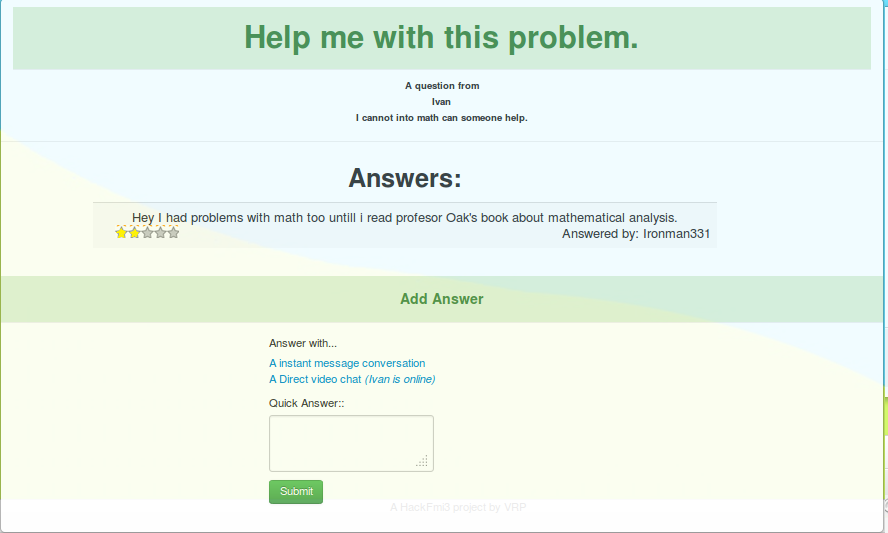
Страницата за преглед на въпрос съдъжа , заглавието на този въпрос , името на потребителя задал въпроса , и самия текст на въпроса.

На въпроса могат да се поставят множество отговори различни по тип. И по количество опит което дават. Всеки отговор показва текста на самия отговор , потребителя който го е дал , както и рейтингът на въпроса.

Директните отговори могат да бъдат оценявани със звезди , от 1 до 5 , като 2 или по-малко звезди носят отрицателен рейтинг (виж оценяване на отговори по-надоло).

Във формата на екрана може да бъде променяно едно единствено поле : Бърз отговор , то е задължително поле и представлява отговора на въпроса. Зададен по-нагоре.

На страницата също има линкове до други метода за комуникация. Задалият въпроса може да маркира отговор за верен, като hover-ne над него и прекрати видимостта на този въпрос във дашбордовете на останалите потребители. Или да избере заглавието на въпроса си и от показалото се меню да го затвори , като отговорен чрез други методи.

**Фиг 18***. Изглед на страницата за отговор на въпрос*

#### Отговаряне на въпрос със бърз отговор (пост)

Това е метод за отговаряне подобен на форум с разликата че отговорите могат да бъдат оценявани от потребителите.Отговарянаето става посредством описване на текст в предназначеното за това място и натискането на бутонна “Submit”. Оценяването на този вид коментари става посредством интерфейса за оценяване от 1 до 5(поставяне на 1 или 5 звездички). След оценяване на отговор потребителя изпратил го получава точки опит и също точки ранг , в категорията на въпроса.Чрез копчето за отговаряне с моментно съобщение(“A instant message conversations”) може да се отговаря на въпрос и с моментно съобщение.

#### Отговаряне със instant messaging chat

Това е линк към страница за директна конферентна комуникация със задалият въпроса и хората които се опитват да му помогнат. Този линк се показва само ако задалия въпроса е онлайн. При избиране на тази опция се зарежда интерфеиса за чат, с линк агрегатор и възможност за оценка на всяко изпратено инстантно съобщение. Подробности в раздела Чат.

#### Отговаряне на въпрос със Audio/Video разговор

Започва директна аудио-видео връзка със задалият въпроса.Този линк се показва само ако задалият въпроса е онлайн. Зарежда се интерфейса за видео и аудио разговори през WebRTC.

#### Отговар във Facebook

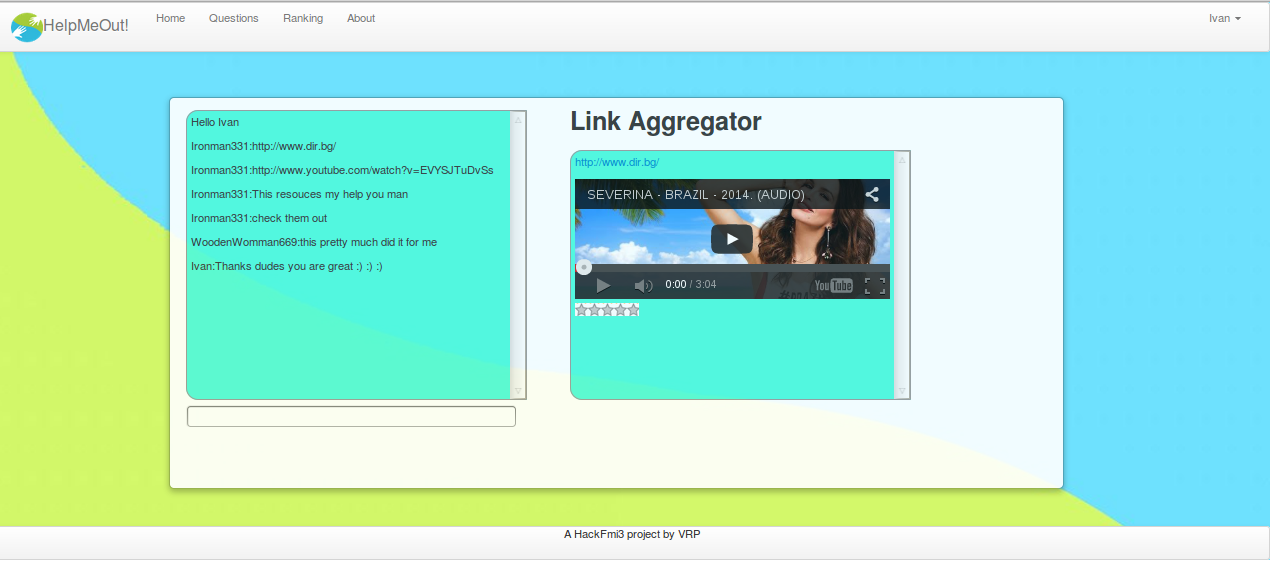
Ако задалият въпрос потребител има Facebook account , може да се изпрати съобщение до него през фейсбук. Съобщенията през Facebook и Twitter не могат да се оценяват.

#### Отговор във Twitter

Ако задалият въпрос потребител има Twitter account , може да се изпрати съобщение до него през twitter. Съобщенията през Facebook i Twitter не могат да се оценяват

### Чат

На този екран се провежда конферентен разговор между задалия въпрос и хора опитващи се да му помогнат. На страницата имаме два основни елемента : чатът и линк агрегаторът. Във чата получаваме съобщенията написани от другите участници ,всеки един от отговорите може да бъде оценен , като се hover-не и се кликне like от излезлия контрол за гласуване, а във линк агрегаторът се запазват всички линкове и клипчета от youtube , които са постнати. Линквете също могат да бъдат оценени , със звезди, от 1 до 5, като под 2 звезди носи отрицателен рейтинг.

**Фиг 19.***Instant Messaging Conversation екран.*

#### Агрегатор на линкове

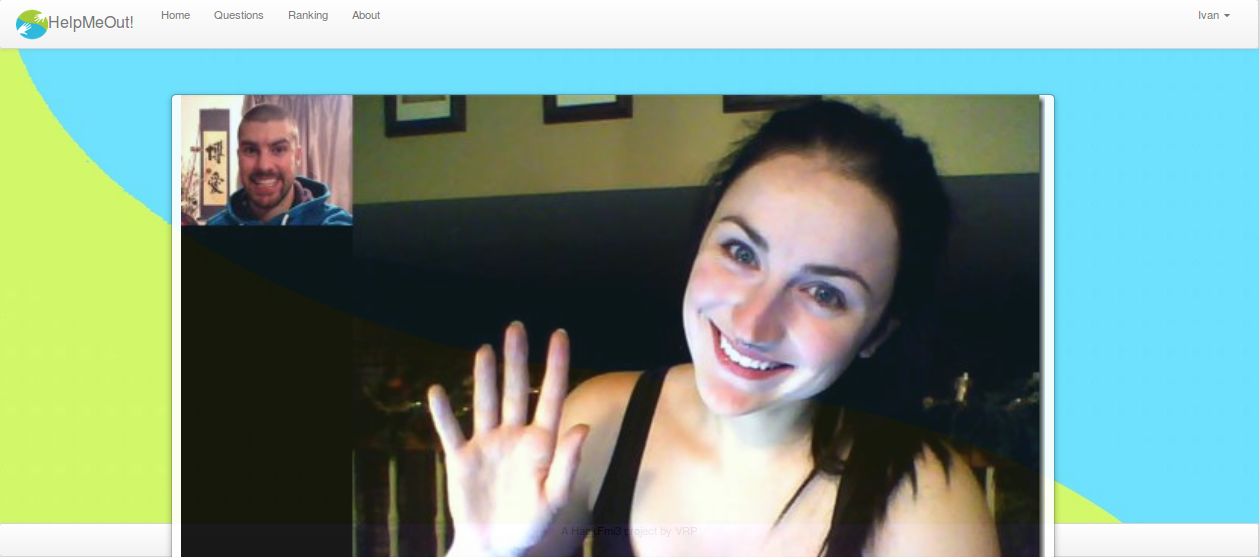
Линковете постнати във чата се добавят към края на десния прозорец наречен агрегатор на линкове. Линковете трябва да могат да се кликат и да бъдат оценявани със звезди от 1 до 5.

#### Агрегация на YouTube клипчета

Клипчетата от youtube които са постнати също трябва да могат да се агрегират във агрегатора, подобно на линковете , но за разлика от тях , трябва да се embed-vat , и да показват самото клипче. Те също могат да бъдат оценявани с оценка от 1 до 5 чрез звезди.

### Аудио видео разговори през webrtc

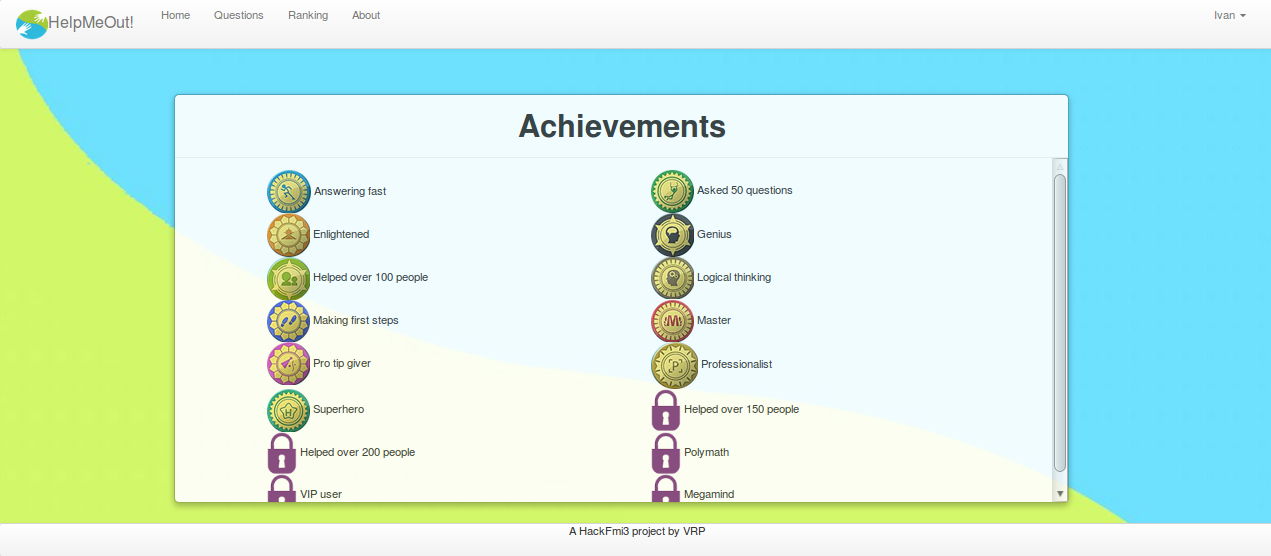
На страницата има единствен прозорец , който показва потребителя със който се провежда видео разговор и във горния ляв ъгъл - самият ти , потребителят.

**Фиг 20.**Аудио Видео разговор между потребители

Кликането с десен бутон върху чата изважда меню за оценяване и прекратяване на разговора. Оценяването е от една до десет звезеди , като оценка под три звезди е отрицателна. Видео разговорът става по WebRTC , който е поддържан само от най-новите браузъри , за това потребител който не поддържа технологията получава съобщение за грешка която му съобщава че трябва да обнови web-browser-ът си.

### Постижения

При случването на определени действия в системата , потребителят получава нови постижения. На страницата със постиженията са показани всички възможни постижения . При посочването на постижение със мишката може да види датата на която го е получил , изискванията му , както и колко други потребители са го получили.

**Фиг 21.***Постижения на потребителя с име Ivan*

Постиженията които потребителя не е отключил не показват иконката си , а показват катинар , който показва че потребителя още не е получил това постижение.При получаване на ново постижение , потрибителя получава notificatоn, като червен знак (!) до името си. Следва списъкът със всички възможни постижения във системата.

| Постижение | Изискване за придобиване |
| --- | --- |
| **Answering Fast** | Answer 5 questions under 10 minutes |
| **Enlightened** | Receive an answer to 10 questions |
| **Helped over 100 people** | Have a answer ranked 4 or more on more than 100 questions |
| **Making first steps** | Answer your first question |
| **Pro tip giver** | Participate in 50 conference chats |
| **Superhero** | Enter a videochat 10 seconds after the asker posts the question |
| **Helped over 200 people** | Have a answer ranked 4 or more on more than 200 questions |
| **VIP User** | Have an answer ranked 5 by more than 30 people , and not ranked lower than 4 |
| **Asked 50 questions** | Ask 50 questions |
| **Genius** | Achieve the last rank in 2 fields |
| **Logical Thinking** | Answer 10 Yes/No questions correctly in a row. |
| **Master** | Achieve the last rank in a field |
| **Professionalist** | Achieve a rank higher than 4 in more than 10 fields. |
| **Helped over 150 people** | Have a answer ranked 4 or more on more than 150 questions |
| **Polymath** | Achieve a rank higher than 4 in more than 15 fields. |
| **Megamind** | Achieve a rank higher than 4 in more than 30 fields. |
| **Sleeping Dragon** | Have 10 or more interest fields with no rank. |
| **Confucius Apprentice** | Master all of your interest fields |

**Таб. 1**. *Списък от всички възможни постижения в системата*

### Организации

Организациите позволяват на потребителите да създадат йерархия и да спомагат работния си процес, чрез получаване или предоставяне на помощ посредством системата на HelpMeOut.

#### Създаване

Създаването на организация става чрез заявяване на такова желание, и след одбрение от администратор.След като тези две неща са изпълнени, с помоща на спецялист посредством лесен потребителски интерфейс се пресъздава максимално правдоподобно организацнионната схема , и връзките между потербителите в нея. Създава се ясна йерархия, под формата на частична наредба , и потребителски акаунт на организацията, с който тя може да прави изяви и извън групите си като обикновен потребител.

#### Администриране

Администрирането на организации става чрез интерфейс достъпен през потребителския акаунт на организацията.Основни операции при администриране са създаване и администриране на групи , създаване на връзки между потребителите в организацията.

#### Боравене с йерархии

Боравенето с йерархии в HelpMeOut е съвсем просто. Всеки потребител принадлежащ към дадена организация има потребители на който може да поставя въпроси пряко (“назначава им въпроси за помощ”, подчинени), потребители който могат да му поставят въпроси за помощ(шефове) и потребители с който няма отношения (тоест такива на който неможе да назначава въпроси , нито пък те на него)Потребители, който нямат отношения могат да комикират посредством основните страници, в които и двамата са участници.

#### Създваване на затворени страници за взаймопомощ

Създаването на затворени страници за взаимопомощ стават чрез бутона “Create Private page for this organization”. След това през заредилия се инерфейс се добавят потребителите от огранизацията членове на тази страница, и се задават правата им за задаване и отговаряне на въпроси.

#### Групи

Групите представляват страници, в които организациите да организират и подреждат задавните въпроси.Затворените страници всъщност са групи с опция private и без потребители-членове, които да са извън тази организация.

* + - * 1. Създаване

Създаването става чрез бутон “Create group in this organization” .В появилия се интерфейс се избират опциите за видимост на групата(public or private).Ако дадена група е публична всеки потребител от системата може да я види. Ако ли не досъп до нея имат само тези потребители, които са посочени като нейни членове.Избират се също членовете на тази група и се натиска копчето “Submit”. Ако групата е с опция private и без членове извън органзиацията, то тя автоматично бива пречислена към затворение странници за взаймопомощ на организацията.

* + - * 1. Администриране

Операциите по администриране на групи включват добавяне и премахване на потребители от група и променяне на видимоста на група. Това става посредством копчето за редактиране на група. Ако група е създадена като затворена страница, тя неможе да бъде направена с видимост “публична”, но ако тя е създадена като група с параметрите на затворена страница, то тогава видимоста подлежи на промяна.

#### Потребителски акаунт на организацията.

Всяка организация получава потребителски акаунт на свое име. Той служи за две основни неща. Първото е, че дава възможноста на всяка организация да участва активно в отговарянето и задаването на въпроси на и към обикновенните потребители (неучастници в организация).Второто приложение на този акаунт е да се администрира съответната организация. Всички интерфейси за администриране и създаване на затворени страници и групи са досъпни само чрез потребителския акаунт на организацията.

### Списък категории

Това е страница на която виждаме всички категории със възможност за сортиране по голямо множество от параметри. Параметрите по които категориите могат да се филтрират са следните:

* Брой потребители
* Азбучен ред
* Брой въпроси
* Брой отговори
* Средно количество отговори за въпрос
* Процент на въпросите на които е намерен отговор
* Брой Майстори - Брой на потребите със максимален ранк в тази категория

Всяка една от категориите в таблицата може да бъде избрана , водейки към списък въпроси от тази категория.

### Списък въпроси по категория

На тази страница се показва списък от въпроси от дадена категория. Въпросите на тази страница могат да се филтрират и сортират по няколко параметъра , параметрите по които могат да се сортират въпросите са :

* Азбучно
* Време на задаване (стандатно сортиране)
* Брой останали категории
* Брой отговори

### Справки и статистики

Всяка организация и някои потребители който са пожелали тази услуга ще могат да използват доклади за въпросите,които са им задавани , и/или на който са отговаряли.

#### Съставяне на докалди и статистики.

Съставянето на доклад става чрез копчето “Create report” , в последствие в заредения интерфейс се избират подходящите опции(зададени и/или отоговорени въпроси, фитлтриране по дати , оценка на отговора,категории) и се натисика “Generate”

Ако потребителя иска да запази, схемната на доклада (опциите, който е избрал) натиска “Save as custom ”.

#### Експортиране в ексел и и уорд формат

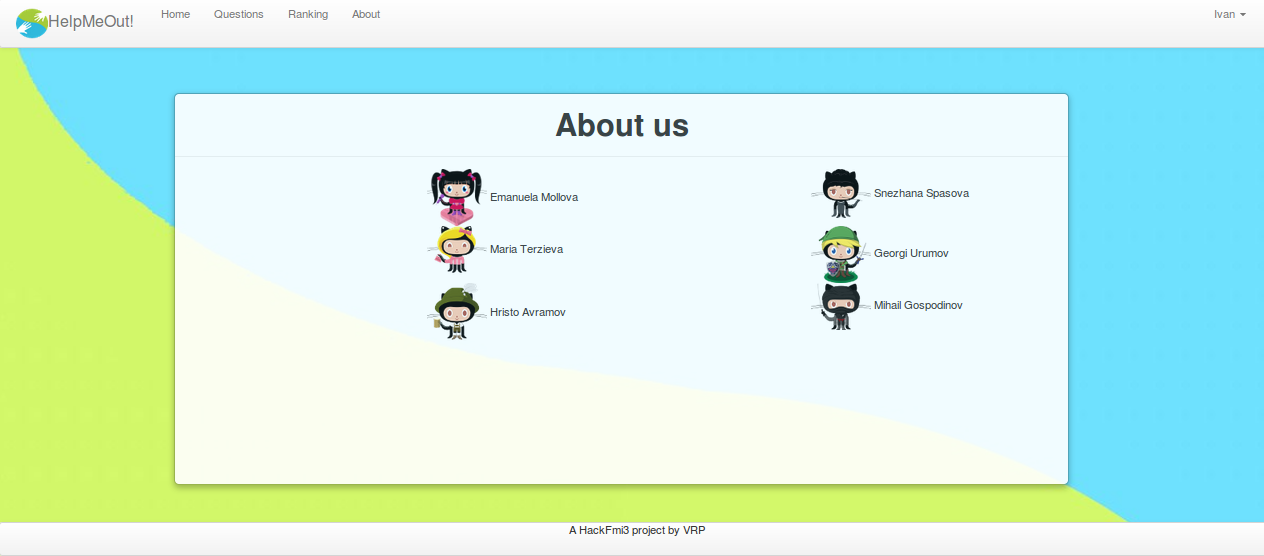
След като репорт бъде генериран, той може да бъде свален в ексел,уорд или пдф формат чрез копчето “Export in” и избиране на съответния формат.

#### Експортиране в json и XML формат

Всички видоде доклади могат да бъдат представени в json и XML формати за да бъде възможно интегрирането на различни програми със HelpMeOut .Toва става чрез POST заявки към съответния URL за генериране на репорти,последван в края си съответно от .json или .xml.

### За нас

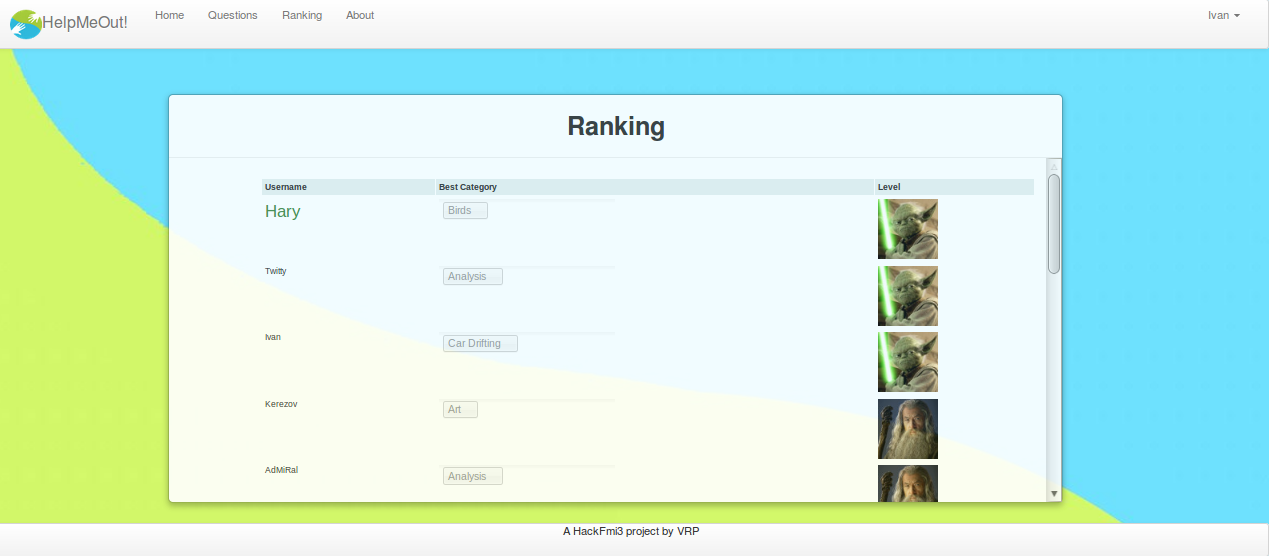
На тази страница има обща информация за създателите на проекта , както и за неговите цели и планове за бъдещо развитие. Съдържа контактна информация чрез която потребителя може да се свърже или да останови контакт със поддържащия персонал на системата при нередности или бъгове.

**Фиг 22.** *Изглед на страницата "За Нас"*

### Ранклиста

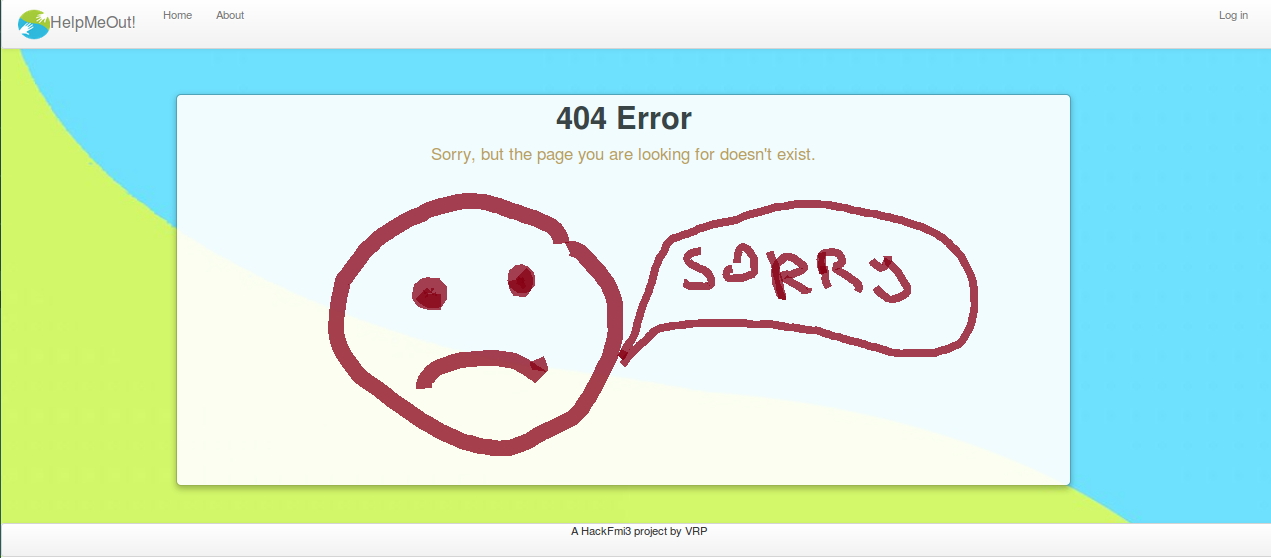
Ранклистата е станица на която са показани потребителите сортирани по опит , на тази страница те могат да се филтрират по категориите в които те са на-добри. За всеки потребител са показани следните данни :

* Име на потребителя
* Категория в която потребителя е най-добър
* Общият ранк на потребителя

**Фиг 23.** *Изглед на страницата с ранклистата*

### 404 Страница за грешка при ненамиране на страница

Страница която се показва при опит за отваряне на страница която не е мап-ната на път в контролер руутингите. При показване на тази страница се запазва запис във log файл за да могат да се следят неуспехите.

**Фиг 24.** *Изглед на страницата за грешка 404*

## Сигурност

### Cross Site Scripting defence

Cross Site Scripting представлява успешното пускане на потребителски скрипт от клиентска страна върху уеб страница видяна от други потребители. Като защита от такива сме предриели:

Всички данни минават през вградено от създателите на синатра ескеипване, което не позволява да бъдат предриемани Cross Site Scripting атаки.Също така при печатане на каквато и да е информация задавана от потебители се използва вградено ескейпване на темплейтния език haml.

### Cross Site Request Forgery defence

Cross Site Request Forgery представлява изпълняване на злонамерени заявки разчитайки, че потребителя има сесия за определен уеб сайт.Пример за такива са всички тайни и скрити бутони , който ни карат да харесваме разни неща във фейсбук без дори да подозираме за това си дейсвие. Като защита срещу Cross Site Request Forgery сме реализирали стандартния метод, при който преди всяко зареждане на форма, сървъра изпраща уникален код на клиентския браузар, който се запазва в сесията на сървъра и ако не бъде върнат обратно заявката бива отхърлена.

### Captcha срещy ddos and password stealing

На всички страници на които потребителят трябва да се аутентикира , при повече от 3 неуспешни опита , пред него се предствавя reCaptchа форма , която той трябва да попълни , за може да прави следващи опити за аутентикация във системата. Потребителят , ако не е човешко същество , а crawler , няма да успее да премине през капчата , и ще спре опити за налучкване на пароли. Тази технология предпазва също и от ddoc атаки(опити за претоварване на сървъра чрез изпращане на голям брой едновременни заявки)

### Passwords hashing(bcrypt)

(Виж в архитектурата.)

### SSL

Сайтът ползва SSL за криптиране на трафика до браузъра на юзера , това помага против Man in the Middle и Replay атаки, както и против разкриването на информация за потребителя чрез Sniffing.

## SEO Изисквания

Следните правила се спазват при имплементиране на системата за да бъде възмножно най-добре индексируема от търсачки и кролери:

* Наличие на robots.txt , във който е посочен линк към sitemap.xml
* Sitemap.xml , съдържащ всички релевантни страници на уебсайта
* Присъствието на поне един и единствен h1 елемент на всяка страница
* Описателен браузър title за всяка страница
* Ползване на ARIA и семантични HTML5 тагове
* Валиден html
* Google analytics
* Ползване на <meta name=”keywords”> и <meta name=”description”> тагове , като description таг-а е с максимална дължина 155.
* 50%+ отношение на текст / html
* Alt тагове за всички имиджи
* Search-friendly URLs
* Ползване на <meta name=”author”>

## Use Case-ове

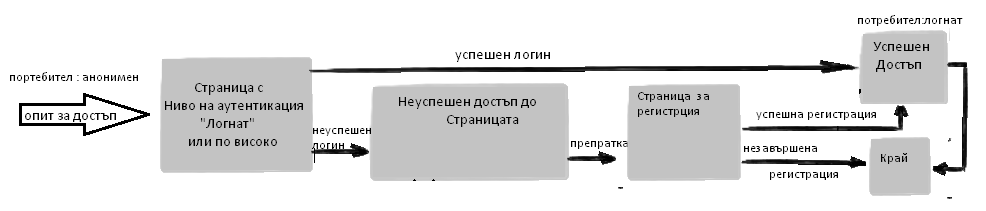
### U001 Анонимен Потребител се аутентикира пред ситемата

* Потребителят влиза в системата
* Системата представя началната страница
* Потребителя натиска LogIn линка
* Системата представя пред него логин страницата (1)

#### U001.1 Успешна аутентикация

* Потребителя въвежда детайлите си правилно
* Потребителя е препратен към дашборда си , и е аутентикиран в системата
* Край

#### U001.2 Неуспешна аутентикация

* Потребителя въвежда детайлите си неправилно
* Потребителя е препратен към логин страницата със съобщение за грешка
* Use Case-a продължава от (1)
* **Фиг U001,U002.** *Use Case 001, 002*

#### U001.3. Три опита за неуспешна аутентикация

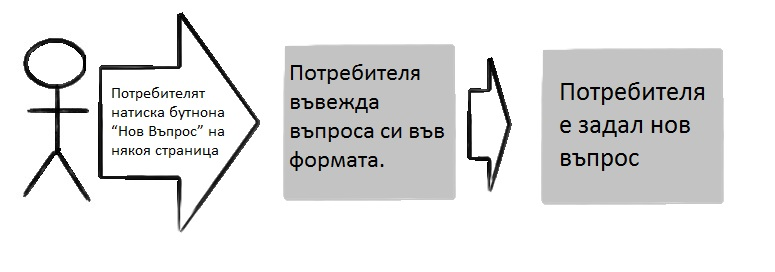
* Потребителя въвежда детайлите си неправилно за 3ти път
* Потребителя е препратен към логин страницата със съобщение за грешка, като на нея допълнително е рендерирана reCaptcha за установяването на хуманноста на потребителя
* Use Case-a продължава от (1)

### U002 Анонимен Потребител се опитва да достъпи страница изискваща Ниво : Логнат

* Потребителят се опитва да достъпи дашбордът
* Ситемата го препраща на Login страницата
* Потребителят натиска линк към Register страница
* Системата представя пред потребителя register формата
* Потребителят попълва регистрационната форма успешно
* Потребителят е препратен към дашбордът вътре в системата със своите детайли
* Край

### U003 Аутентикиран Потребител задава нов въпрос

* Потребителят натиска бутнона “Нов Въпрос” на някоя страница
* Потребителя въвежда въпроса си във формата.
* Потребителя е задал нов въпрос
* Край

**Фиг U003.** *Use Case 003*

### U004 Аутентикиран Потребител отговаря на въпрос със съобщение

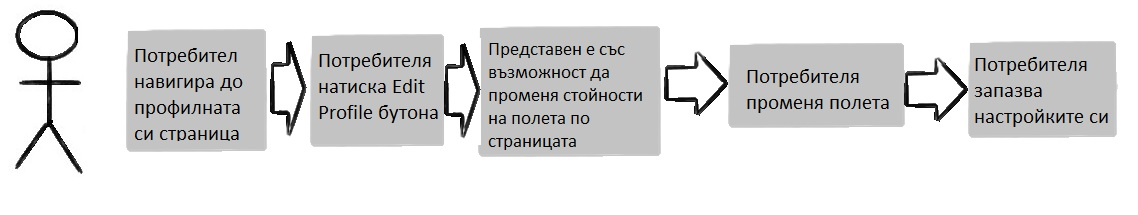
* Потребителя навигира до въпрос
* Потребителя попълва quick answer формата
* Потребителя е направил нов отговор
* Край

### U005 Аутентикиран Потребител отговаря на въпрос със чат

* Потребителя навигира до въпрос
* Потребителя избира live chat опцията като задалия въпроса е онлайн
* Потребителя започва чат със задалия въпроса
* Чата свършва при напускането на задаващия или потребителя
* Край

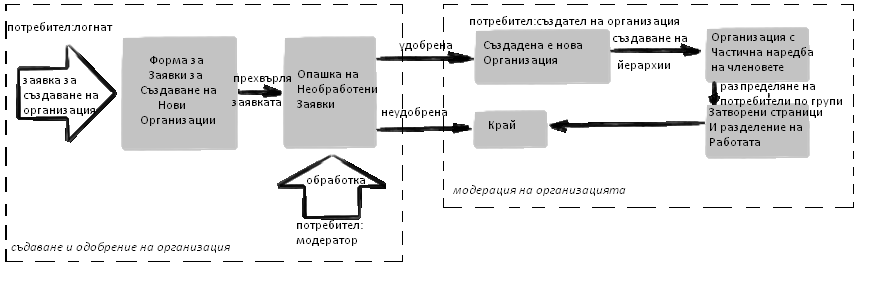
### U006 Аутентикиран Потребител менажира профила си

* Потребител навигира до профилната си страница
* Потребителя натиска Edit Profile бутона
* Представен е със възможност да променя стойности на полета по страницата
* Потребителя променя полета
* Потребителя запазва настройките си
* Край

**Фиг U004-U006.** *Use case 004 to 006*

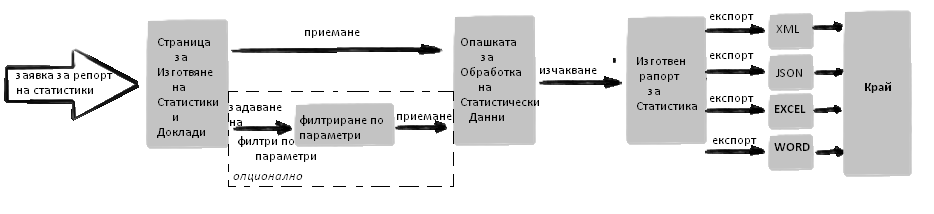
### U007 Аутентикиран Потребител създава и менажира организацията си

* Потребител навигира до страницата Oрганизации.
* Потребителя рекуества създаването на нова организация
* Потребителя бива одобрен за създаването на организация
* Потребителя навигира до страницата на своята организация.
* Потребителя създава дървото на потребителите в организацията си чрез tree tool-a и ги групира на в групи според нужните им права и възможности
* Край

**Фиг U008.** *Use Case 008*

### Експортиране на статистики във XML , JSON , WORD , EXCEL

* Потребител навигира до страницата Експорти.
* Потребителя рекуества създаването на статистика
* Потребителя опционално филтрира и сортира резултатите си
* Заявката се нарежда на опашката за обработка на статистически данни
* След като бъде обработена заявката , потребителя може да изиска изготване на експорт на данните в XML , JSON , EXCEL , WORD.
* Край

**Фиг U008.** *Use case 008*

## Гласуване

Стратегията за изчисляване на брой опит получен за отговор когато той получи нов глас се състои от няколко стъпки :

* P e ранкът на потребителя който е гласувал като число , където 1 е най-високия ранк
* Ако главуването е с “like”/”dislike” Q = 2 при like. Q = -1 при dislike
* Ако гласуването е в следвстие на изтичане на Yes/No Question при правилен отговор Q = 1 + (процентът правилни отговори) \* 1 иначе , Q = -1 - ( 1 - (процент правилни отговори)\*1)
* Ако главуването е със звезди изчисляваме R = 3 при максимален брой звезди 5 , или R = 4 при максимален брой звезди 10. Смятаме Q = (рейтинг - R)\*1.5
* смятаме опитът = Q + Q \* ( 1 / P )

## Изкачване в ранк

Потребител се изкачва в ранк , когато достигне нужния брой опит за достигането на този ранк. Ако опитът му падне под нивото на ранк , той го губи , както и правата които получава със него.

# Архитектура

Дефинира основните архитектурни решения за реализация на системата , на две нива : физическо и софтуерно.

## Физическа архитектура на системата

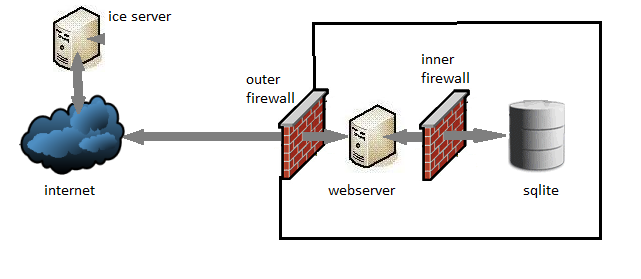
Следва описанието на физическата архитектура на системата .

### Обобщено

Системата е разпределена между три физически машини , едната помещава sqlite базата от данни , другата webserver-a и трета със ICE server.

### Схема на мрежата

Webserver-a и SQLite server-a са изолирани в отделна мрежа. Докато ICE server-a поради трудноста за менажирането му е външен service под наем. SQLite server-a допълнително е изолиран чрез още един firewall , докато Webserver-a седи в демилитаризирана зона.

**Фиг 25.** *Физически модел на мрежата на системата*

### WEB SERVER

Web server-а е физическата машина отговорна за content delivery на данни по http/https и е основната част от системата. Той седи в демилитаризирана част от мрежата и позволява достъп до него от външни потребители на портове 80 и 443.

### SQLite Server

SQLlite server-a е физическа машина , на която е ситуиран sqlite3 сървер , служещ за основна база данни на уебсървера. Той е допълнително защитен с още 1 firewall тъй като съдържа важна информация.

### ICE Server

ICE Server-a е service под наем , на него има ICE server който спомага за NAT Traversal-a така че да могат да работят видео разговорите по WebRTC.

## Софтуерна Архитектура

### Обобщено - Софтуерен Стек

Нашата система се състой от машина със Linux Mint операционна система, на която е вдигнат Thin web сървър, който през Rack си комуникира със Sinatra приложение, всичко писано на Ruby 2.1

#### Операционна Система(Linux Mint)

Операционната система, на която трябва да стъпи нашето web приложение е Linux Mint. Тя е една от по-популарните в момента дистрибуции на Linux и е широко използвана и съответно добре подържана. Като цяло Linux базираният сървър е по-надежден от еквивалентният Windows базиран сървър по няколко причини.

Като за начало, чисто статистически по-голямата част от web приложенията са host-нати на Linux базирани сървъри. Има добра причина за това. Тази операционна система е известна с това, че е в състояние да продължава да работи безотказно през изключително дълги периоди от време. Понякога имаме примери за uptimes от порядъка на седмици, месеци или дори години.

Друга важна причина разбира се е сигурността, с която са известни Unix базираните операционни системи. В днешното интернет пространство сигурността на личните данни и информацията, която нашите потребители поставят в наши ръце, е нещо, с което не може и не бива да се прави компромис.

Из сред всички дистрибуции на Linux за наша host операционн система избрахме именно Linux Mint понеже поне по последни проучвания тя е втората по популярност дистрибуция ва Linux и съответно има голямо community и от хора, които биха били от голяма помощ на нашия проект. Освен това повечето членове на екипът ни разполагат с машина, на която е инсталиран Linux Mint.

#### Език

Езикът, на който е писан проекта е Ruby 2.1.0 , билднат локално на всяка development машина от сорс-код.

* + - * 1. Защо Ruby?

Решението да използваме именно Ruby за този проект е съвсем естествено предвид поставените цели. Ruby е от онези езици, на които със доста малко код може да се постигне доста, както откъм функционалност, така и от страна на създаване на абстракции. Способността да се пишат бързо програми, които да могат почти веднага да влязат в употреба и да се използват на момента, е ключово за постигане на успех. Ruby предлага онази гъвкавост и бързо прототипиране (rapid prototyping), която е така незаменима в днешно време. Ruby е един от най-гъвкавите езици за програмиране, който е доказал своята надеждност, когато стане дума за създаване на web приложения.

Може би най-добрата причина да използваме Ruby обаче си остава разбира се наличието на толкова много готова функционалност под формата на библеотеки, които в Ruby светът се наричат gem-ове. Почти няма широко използвана функционалност за която да не съществува gem в Ruby. Това означава, че това е език, който ни позволява да започнем да мислим за важните неща свързани със всеки един проблем, вместо да се налага да губим време с boilerplate код. На Ruby съществуват DSL-и за всевъзможни неща не случайно. Самата му структура позволява бързо надграждане и развиване на нови идеи. За всяка задача си има правилните инструменти и Ruby е може би най-правилният инструмент, когато стане дума за бързо създавене на web приложения, буквално от нищото.

* + - * 1. Gem-ове

В тази секция описваме gem-вете които са избрани за проекта. Gem се наричат безплатните библеотеки от функционалности съществуващи из Ruby web пространството. Голяма част от тях са широко използвани и добре подържани парчета софтуер, които се използват в редица успешни проекти. Всеки от тях е добре подбран според нуждите на нашия проект. Идеята е да се преизползва съществуващ софтуер за решаването на проблеми, за които съществува готово добре работещо и изпробвано от много хора решение. За тяхното подбиране се консултирахме и със специалисти, които реално използват тези технологии в много проекти, за които времето вече е доказало, че са успешни.

На няколко пъти подбора от gem-ове беше променян по време на фазата на планирането на проекта. След дълги дискусии се достигна до този окончателен списък от технологии, които всички сме съгласни, че ще свършат чудесна работа без излишно “преоткриване на колелото”. Едно от основните изисквания към всеки gem, разбира се, е да бъде напълно безплатен и open-source. Следва списък от осемте предварително подбрани библеотеки, които са използвани в проекта.

gem 'sinatra'

Съществуват няколко големи framework-а за създаване на web приложения писани на Ruby. Може би най-известният от тях разбира се е Rails. За нашия проект решихме да не използваме, този мастодонт. Причините за това решения бях много и комплексни. Една от основните и може би най-важната причина беше, че нашия екип не разполага с необходимия опит с Rails. Sinatra е логичния избор в случая.

Sinatra е DSL (Domain Specific Language) за създаване на web приложения. Неговото преимущество пред Rails специално за този проект е че достатъчно гъвкав и позволява съвместното използване и на други технологии. Създаден е от Blake Mizerany. Разпространява се свободно и напълно безплатно под MIT лиценза. Sinatra зависи от Rack за своя web server interface. Малък и гъвкав е. За разлика от по-големите в бранша, като Rails например, не използва стандартния MVC (Model-View-Controller) модел. Основният фокус на Sintra е да предостави възможността за създаване на web приложение, колкото е възможно по-бързо и с минимално усилие. Използва се в големи сайтове като GitHub, LinkedIn. Големи организации като Apple, NSA и BBC също го използват.

gem 'sinatra-websocket'

sinatra-websocket е gem разширение за sinatra , който позволява да се върне респонс “101 Switching Protocols” и да създаде websocket връзка със потребителя , позволяваща директна комуникация (без request/response) за целите на разговори на живо.

gem 'haml'

Haml (HTML abstraction markup language) е markup език който представлява реална алтернатива за създаване на HTML темплейти. Основните концепции залегнали зад неговото създаване, както и реалната му утпотреба, са чистия красив markup. Идеята е че е по-лесен за четене и разбиране от суровия HTML. Това се постига чрез използване на индентация за формиране на блокове, доста по-DRY красив дизайн и други. Същевременно не се губи никаква ефективност или свобода на изразяване на начина по който искаме да представим съответния документ.

Haml е създаден от Hampton Catlin, но се подържа основно от Nathan Weizenbaum. Haml е лицинзиран (също като Sinatra) със MIT лиценза, което го прави достъпен безплатно. Използва се широко в Ruby on Rails приложния.

По тези прочини избрахме да използваме Haml за нашия проект с надеждата това да ускори работата по него, тъй като така не се занимаваме директно с HTML.

gem 'sequel', '~>4.3'

Sequel е може би един от най-полезните gem-ове в проекта. Посредством sequel можем лесно да създаваме нова SQLite база данни, която да съдържа информацията за клиентите на нашето web приложение. Това което го прави толкова добър, е неговата способност да създава автоматично база от данни от вече съществуващите Ruby класове. Тези му характеристики превръщат този gem в доста лесен избор за нашия проект. Sequel ни спестява доста време и работа свързана с организацията и поддръжката на базата данни.

gem 'sqlite3'

Sqlite3 е Ruby gem, който позволява на нашето Ruby приложение да си комуникира директно със SQLite3 база данни. Типично за Ruby използването на базата данни става доста лесно и бързо чрез този gem. SQLite3 е де факто стандартния начин Ruby да си общува с такава база данни. Екипът ни избра да използва тази технология просто защото е сред най-често срещаните решения на този проблем и няма алтернативи, които съществено да се различават по отношение на функционалност или други характеристики. SQLite3 е просто точно това, което ни трябва - нито повече, нито по-малко.

gem 'attr\_encrypted'

attr\_encrypted е gem за Ruby, който прави нещо много просто. Той просто създава аксесори (getters and setters) за променливи в даден Ruby клас. За разлика от стандартните аксесори (attr\_accessor) обаче променливите декларирани посредством attr\_encrypted автоматично се криптират със някакъв ключ. Когато даденият обект трябва да бъде сериализиран, той се криптира със съответния ключ. Основната причина за използването на този gem е очевидно сигурността на данните, които са предоставени от клиентите на нашата система. Освен това му преимущество този gem работи доста добре със Sequel друг gem избран за нашия проект.

gem 'chartkick'

chartkick е gem който служи за създаване на “чартежи” и всякакви други графики използвани за представяне на информацията, като например pie chart, line chart, bar chart и всевъзможни други. С помоща на този gem е се създават подобни javascript чартежи буквално само със няколко реда Ruby код. Работи със почти всички големи frameworks за създаване на web приложения включително Ruby on Rails, Padrino и разбира се Sinatra.

Доста добре поддържан е gem-а и ни дава точно това от което се нуждае нашия проект. Когато стане дума за визуализация на информация някъде из нашето web приложение, chartkick ни помага да предоставим на потребителя красиви графики, с чиято помощ той да разбере по-добре информацията, която му предоставяме.

gem 'rerun', require: false

Rerun е gem, който служи за менажиране процеси или по-специално тяхното рестартиране, когато е необходимо. На практика rerun наблюдава файловата система и в момента, в който забележи разлика във някакви определени файлове, които му е казано да “наблюдава”. По подразбиране се наблюдават само файлове с разширения rb, js, css, coffee, scss, sass, erb, html, haml, ru, slim и md, каквито в нашия проект се срещат в изобилие. Rerun помага за автоматично тестване, както и за подържане на сървъра в изправност дори при промяна на конфигурационни файлове.

Използваме го защото е полезен инструмент при разработването на подобен проект, тъй като при промяна на файл , се прави пълен рестарт на сървера и виждаме резултатите от промените мигновенно.

#### Уеб сървър

Цялата система се разделя на основно 3 части. Първата е web сървъра, който в нашия случай е Thin. От другата страна стои Sinatra, който се грижи за логиката. По между им намиламе Rack който служи за интерфейс между двете.

За сървър на Ruby използваме Thin. Основните му преимущества са, че е много малък (за това му викат thin) и освен това е много бърз и лек. Същевременно е доста лесен за използване (както всичко останало в Ruby) и това ни помага да вдигнем нашия сървър почти веднага след началото на проекта. Thin ни дава всичко от което имаме нужда, без излишни неща, които биха ни забавили.

За да свържем Thin със framework-а, който сме си избрали (а именно Sinatra) използваме gem наречен Rack. Това е де факто стандартния интерфейс между web сървъри, който поддържат Ruby и съответния framework. За да се използва Rack просто трябва да се предостави app, който е ruby обект, който отговаря на call метода. Той трябва да приема хеш със средата (the environment) като параметър и трябва да връща масив от 3 елемента. Първия трябва да е http response кода, следван от хеш от хедъри и накрая тялото на response-а, който трябва да е Ruby обект отговарящ на each метода. И типично в стил Ruby това е всичко, което нашето приложение трябва да свърши за да свърже Sinatra със сървъра Thin.

#### Поддържани версии на html/browsers/ecmascript

Нашето web приложение цели да поддържа всички модерни browser-и, като Firefox 26, Chrome версия 35.0.1916.153, Opera 18 и Internet Explorer версия 8 и нагоре. HTML версията от която ще се нуждаят клиентите е HTML 5. Също така и Ecmascript 5

* + - * 1. Javascript библиотеки

Направата на динамичен интерфейс , който да спомага работата на юзера налага изискване за JavaScript библиотеки които да улеснят изграждането на интерактивната система.

JQuery

Проектът се нуждае от JQuery като javascript библеотека за скриптиране на HTML от страната на клиента. Тя позволява да се пряват динамични web страници минимално усилие. Лицензът и е MIT.

JQueryUI

JQueryUI е колекция от графични елементи за създаване на GUI (Graphical User Interface) в web страници и web базирани приложения. Елементите са реализира с помоща на HTML 5 и CSS 3. Както си личи от името JQueryUI е реализира използвайки самата JQuery библеотека. По тази причина лицензът отново е MIT.

* + - * 1. CSS билиотеки

За улеснение на фронтенд екипа са избрани билиотеки за спомагане на изготвяне на дизайна на проекта

Twitter Bootstrap

Twitter Bootstrap е библеотека която ни помага със изгледа на нашето web приложение. Bootstrap ни предоставя множество от стилове и стилови дефиниции за всички ключови HTML компоненти. По този начив получаваме еднакви стилове и външен вид на всички наши UI елементи и всякакви други части от страницата на проекта.

Създадена е (както си личи от името) в Twitter от Mark Otto и Jacob Thornton, но бързо набира популярност и след време става изцяло open-source под MIT лицевза, което ни позволява да я използваме в нашия проект.

* + - * 1. Websockets

Websocket е протокол, който предоставя full-duplex комуникацонни канали през TCP връзка, които може да свързва web browser и web server. Този протокол прави възможна живата връзка между отделните клиенти на нашето приложение и по тази причина е от особена важност за нашия проект. За щастие се подържа от всички широко използвани web browser-и като Google Chrome, Internet Explorer, Firefox, Safari и Opera.

* + - * 1. Bcrypt

Bcrypt е известен алгоритъм за криптиране на пароли. Имаме намерение да използваме него в нашия проект за да осигурим сигурността на нашите потребители. За момента обаче използваме друг алгоритнъм описан по надолу. Има си специален gem, който се казва Bcrypt-ruby и служи като wrapper за тази функционалност.

* + - * 1. Google reCaptcha

reCaptcha е gem за Ruby, който създава автоматично CAPTCHA изображения, по технологията на Google. Идеята на тези изображения е да зъщитят даден web сайт от злонамерени потребители, които биха използвали автоматизирани програми (bot) за да правят атаки срещу съответния сайт. Използвайки reCaptcha нашия проект е защитен от подобни атаки, със може би най-добрия софтуер за целта.

* + - * 1. WebRTC

WebRTC (Web Real-Time Communication) е софтуер, който позволява на потребителите на нашето приложение да общуват в реално време един със друг. Все пак това е основна част от предназначението на нашия проект. Оригинално проектът е започнат от Google, по късно с помоща на W3C се създава и стандартен протокол за целта. Проектът на Google разбира се е open-source и е широко използван.

* + - * 1. База от данни (SQLite)

Базата от данни, която използва нашия проект е SQLite. Това е релационна база данни писана на C и съответно е доста бърза и надеждна. В нашия проект я използваме съвместно със sqlite3 gem-а на Ruby за да се възползваме изцяло от предимствата, които тя предлага.

### MVC

Използваме стандартния MVC pattern, в който проектът е разделен на грубо казано три отделни, независими части - Model, View и Controller.

#### Модели (описание на това какви са ни моделите)

Моделите ни се създават на базата на съществуващите в Ruby проекта класове, които се транлират с помоща на gem-а Sequel в съответната схема в релазионната база данни SQLite. С други думи използваме модела известен като “code-first approach”, при който първо се пише кода и след това от него се генерира структурата на базата данни.

По този начин имаме генерирани модели за User, който съдържа в себе си връзки към въпроси (Questions) и отговори (Answers) свързани с този потребител. Респективно имаме модели за Answer и Question. Освен тях имаме и модели за AnswerType към който сочат всички въпроси, Field и Rank, който да обединява потребителите в групи според техния ранг.

#### Views (за haml view-tata)

View-тата работят чрез Haml. Напрактика имаме Haml view-та, които се превръщат

в еквивалентни HTML 5 и CSS 3 страници, които показваме на потребителя. View-тата се изпълняват в контекста на контролера имайки достъп до данни от него.

#### Controller (контролери , рутиране)

Нашите Controller-и работят на следния принцип. Всяка POST и GET заявка пристигнала в сървъра има следния вид:

*<ИмеНаСайта> / <ИмеНаКласНаКонтролера> / <ИмеНаМетодОтКонтролера> / [<опционален параметър>]*

Така съотвния POST или GET се трансформира автоматично в извикване на метод на съответния controller.

### Модел на базата от данни

Базата от данни се генерира чрез code first approach . Създадените руби модели се мапват към релации във sqlite релационна база от данни. Домейн моделът на класовете е следният , като са изпуснати тривиалните релации N:M и са посочени само по-сложните с допълнителни атрибути.

#### Потребител (User)

Представлява потребител на системата , потребителят притежава следните свойства:

* Потребителско Име - Логин името на потребителя
* Хеш на Паролата - SHA256 hash на парола + salt
* Salt на паролата - Допълнителен стринг който се апендва към паролата
* Име - Реалното име на потребителя
* Фамилия - Реалната фамилия на потребителя
* Email - Електрона поща на потребителя
* Опит (Experience) - Количество опит което потребителя е спечелил отговаряйки въпроси (сбор на всички рейтинги на отговорите му)
* Знания [N : M към Област през Потребители\_В\_Области] - Области в които потребителят има знания
* Интереси [N : M към Област] - Области в които потребителят би желал да придобие знания
* Въпроси [1 : N към Въпрос] - Въпроси зададени от потребителя
* Отговори [1 : N към Отговор] - Oтговори които е дал потребителя
* Ниво (Level) - текущо ниво на потребителя
* Ранк [1 : N към Ранк] - Ранк на потребителя в системата
* Организация [N : 1 към Организация] - Oргарнизация към която е потребителя, null за хора без оргарнизации.
* Шеф [N : 1 към потребител] - Пряк управник на потребителя (само за хора в оргарнизации)
* Подчинени [1 : N към потребител] - Пряки подчинени на потребителя (само за хора в оргарнизации)
* Администрирана Оргарнизация [N : 1 към Организация] - организация която потребителя администрира , null ако не администрира.
* Администрирани Групи [M : N към Група] - групи които потребителя администрира 0 или много.
* Постижения [N : M към Постижение] - постижения на потребителя

#### Oбласт(Умение)

Тази релация представлява област знания. Съдържа следните свойства

* Име - Име на областта
* Знаещи [M : N към Потребител през Потребители\_В\_Области] - Потребители със тази област на знание
* Интресуващи\_Се [M : N към Потребител] - Потребители интрсуващи се от тази област.
* Въпроси [М : N] - Въпроси от тази област

#### Потребители\_В\_Области

* Потребител [N : 1 към Потребител]
* Област [N : 1 към Потребител]
* Ниво
* Опит

#### Въпрос

Представлява зададен въпрос във системата. Притежава свойствата :

* Заглавие - заглавие на въпроса
* Съдържание - самият текст на въпроса.
* Време на създаване - кога е създаден въпроса
* Време на изтичане - кога изтича въпроса
* Време на отговор - кога е отговорен въпроса
* Тип на въпроса
* Области [1 : N към област] - области които засяга въпроса
* Питащ [1 : 1 към Потребител]
* Отговори [1 : N към Отговор]
* Група [N : 1 към Група] - Група към която може да е зададен въпроса , само за организации.

#### Отговор

Релация представляваща модел на отговор

* Съдържание - Текст на отговора (може да не е само plain text в случай на отговор в чат)
* Време на отговаряне - кога е направен отговора
* Тип на отговора
* Рейтинг - количеството рейтинг което е натрупал въпроса (пази се за forward compatibility , ако се промени количеството рейтинг на отговор от този тип)
* Въпрос [N : 1 към Въпрос] - въпрос към който е отговора
* Отговарящ [1 : 1 към Потребител]
* Гласове [1 : N към Глас]

#### Глас

Представлява глас за отговор

* Знак - Положителен/Отрицателен
* Стойност - допълнителна стойност за някои гласове
* Въпрос [N : 1 към Въпрос]
* Потребител [1 : 1 към потребител]
* Текущ Опит на Отговарящия - служи за изчисление на стойността на отговора.

#### Ранк

Това е ранк на потребител в системата, спечелен чрез отговаряне на въпроси.

* Име на ранк
* Изображение
* Опит
* Потребител [1 : N към потребител]

#### Организация

Организация от потребители на системата които отговарят организирано на въпроси. Предназначено за корпорации.

* Име - име на организацията
* IBAN
* Телефон за контакт
* Лого
* Членове [1 : N от Потребител] - членове на организацията
* Създател [1 : 1 от Потребител] - създател на организацията
* Администратори [1 : N oт Потребител] - членове с административни права над организацията
* Групи [1 : N от Групи] - групи в организацията
* Потребителски аккаунт на групата [1 : 1 потребители]

#### Група

Група от потребители, принадлежаща на оргарнизация.

* Име - име на групата
* Публична - определя дали въпросите към групата се виждат публично
* Външно Задаване - определя дали потребители извън групата могат да задават въпроси към нея
* Организационно Задаване - определя дали потребители извън групата но в организацията могат да задават въпроси към групата
* Групово задаване - определя дали потребители от групата освен администраторите могат да задават въпроси към нея
* Членове [N : M от Потребител]
* Администратори [N : M от Потребител]
* Организация [N : 1 към Организация] - организация към която принадлежи групата

#### Постижения

Постижения на потребителите

* Име - име на постижението
* Съдържание - текст към постижението
* Наградени [N : M към Потребител] - потребители получили постижението

### Aутентикация

Аутентикационната схема на системата се състои от онлайн логин форма предоставена с anti-forgery token против Replay, системата ползва SSL , за да предотврати Man in the Middle атаки.

**Алгоритъм за записване на пароли:**

* При създаване на потребител / смяна на парола на потребител:
* Генерираме случаен 16 символен salt
* Запазлваме salt във записа на потребителя
* Добавяме salt към края на паролата.
* Ползваме SHA256 алгоритъм за генериране на хеш на получения стринг
* Запазваме хеш-а в записа на потребителя
* При аутентикиране на потребител:
* Взимаме 16 символния salt
* Добавяме го към края на подадената парола за потребителя
* Ползваме SHA256 алгоритъм за генериране на хеш на получения стринг
* Сравняваме получената стойност със тази в записа на потребителя.

### Архитектура за разговори

Разговорите се имплементират ползвайки добре известна архитектура при работа с конферентни разговори ,чрез канали

#### Канали

Един канал съществува докато собственика му е в него , при напускането на канала от страна на собственика му , канала се dispose-ва и освобождава ресурсите които ползва. Каналите в архитектурата на HelpMeOut ползват websocket-i за комуникация

* + - * 1. Мениджмънт на канал

Потребителите които са задали въпрос , могат да създадат канал за въпроса като навигират до страницата за разговор , така автоматично се създава канал със потребителя като собственик. Отваря се websocket към браузъра на създаващия канала , и изпочва асинхронно изпращане на данни от него към сървера , като могат да се присъединяват допълнителни потребители като започват собсвена websocket връзка със сървера.

* + - * 1. Типове канали

Каналите не се различават като структура , а само и единствено по данните които те разменят. Видовете канали са следните :

Чат

Потребителите изпращат съобщения по своя сокет , към сървера , който го броудкаства към всеки потребител който е във канала , освен изпращача.

Видео

При видео разговори , каналите се ползват за размяна на SDP(Session description Protocol) [SDP] съобщения , така че да може да се останови peer to peer WebRTC връзка между тях . SDP пакетите се разменят между първите 2 потребителя които влезнат в канала. Канала остава отворен за да могат да се разменят допълнителни съобщения за оценяване , както и за прекъсване на разговора с нотификация към сървера.

# Библиография

[RFCEM] http://tools.ietf.org/html/rfc2822#section-3.4.1 (3/5/2014)

[THIN] http://code.macournoyer.com/thin/usage/ (4/5/2014)

[Jan00] http://andersjanmyr.github.io/jquery\_sinatra/#1 (4/5/2014)

[JQRUI] http://api.jqueryui.com/1.10/category/widgets/ (4/5/2014)

[RBYDC] http://www.ruby-doc.org/core-2.1.2/ (4/5/2014)

[SNTRA] http://www.sinatrarb.com/intro.html (4/5/2014)

[SNTWS] https://github.com/simulacre/sinatra-websocket (4/5/2014)

[SQLIT] http://www.sqlite.org/docs.html (4/5/2014)

[SQLRB] https://github.com/sparklemotion/sqlite3-ruby (4/5/2014)

[CTKRB] https://github.com/ankane/chartkick (4/5/2014)

[ATCRP] https://github.com/attr-encrypted/attr\_encrypted (4/5/2014)

[SQEL] http://sequel.jeremyevans.net/ (4/5/2014)

[HAML] http://haml.info/tutorial.html (4/5/2014)

[WKLXM] http://en.wikipedia.org/wiki/Linux\_Mint (5/5/2014)

[RCPT] https://github.com/ambethia/recaptcha/ (5/5/2014)

[BCRPT] https://github.com/codahale/bcrypt-ruby (5/5/2014)

[WBSOK] http://en.wikipedia.org/wiki/WebSocket (5/5/2014)

[TWBST] http://getbootstrap.com/2.3.2/getting-started.html#examples (5/5/2014)

[RACK] http://rack.github.io/ (5/5/2014)

[RERUN] https://github.com/alexch/rerun (5/5/2014)

[WKMVC] http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93contro

ller (5/5/2014)

[SHA] http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf (достъп 5/5/2014)

[WBRTC] http://www.webrtc.org/ (5/5/2014)

[ECMSC] http://en.wikipedia.org/wiki/ECMAScript (5/5/2014)

[CSRF] http://en.wikipedia.org/wiki/Cross-site\_request\_forgery (5/5/2014)

[CSSCR] http://en.wikipedia.org/wiki/Cross\_Site\_Scripting (5/5/2014)

[ICESR] http://en.wikipedia.org/wiki/Interactive\_Connectivity\_Establishment (5/5/2014)

[RBGMS] https://rubygems.org/ (5/5/2014)

[CSSTU] http://www.w3schools.com/css/css3\_intro.asp (5/5/2014)

[HTMLT] http://www.w3schools.com/html/html5\_intro.asp (5/5/2014)

[SDP] http://tools.ietf.org/html/rfc4566.html (5/5/2014)