
Спецификация на изискванията

за

TaskFlow

Версия 1.0

Изготвена от: Ангел Младенов

Пловдивски Университет „Паисий Хилендарски“

21.05.2025

Съдържание

1. Въведение	4
1.1. Цел	4
1.2. Конвенции	4
1.3. За кого е предназначен този документ	4
1.4. Обхват на продукта	4
1.5. Терминологичен речник	4
1.6. Референции	4
1.6.1. Източници:	4
1.6.2. Хранилище на Приложението:	5
2. Общо Описание	5
2.1. Обща Перспектива	5
2.2. Използвани Технологии	5
2.2.1. Java	5
2.2.2. Maven	5
2.2.3. Spring Boot	6
2.2.4. Spring Data JPA	6
2.2.5. Hibernate	7
2.2.6. MySQL	8
2.2.7. Spring Boot Starter Validation	8
2.2.8. Docker	9
2.2.9. Spring Security	9
2.2.10. JWT	10
2.2.11. Angular	10
2.2.12. JUnit	11
2.2.13. Mockito	11
2.2.14. Git	12
2.2.15. GitHub	12
2.3. Характеристики на продукта	13
2.4. Видове Потребители и Характеристики	13
2.5. Операционна среда	14
2.6. Ограничения за Дизайна и Реализацията	14
2.7. Предположения и зависимости	14
3. Външни Изисквания към Интерфейса	14
3.1. Начален екран	14
3.2. Начален екран за влезнали потребители	17
3.3. Страница за редактиране на потребители	19

3.5.	Хардуерни Интерфейси	21
3.6.	Софтуерни интерфейси	21
3.7.	Комуникационни интерфейси	21
4.	Функционалности на Системата	21
4.1.	Взимане на проектите, в които потребителя участва	21
4.1.1.	Описание и приоритет	21
4.1.2.	Последователност от действия	21
4.1.3.	Предварителни изисквания	22
4.2.	Търсене на задача или проект по име	22
4.2.4.	Описание и приоритет	22
4.2.5.	Последователност от действия	22
4.2.6.	Предварителни изисквания	23
4.3.	Създаване на нова задача	23
4.3.7.	Описание и приоритет	23
4.3.8.	Нужни данни за нова задача	23
4.4.	Промяна на данните за потребител	24
4.4.9.	Описание и приоритет	24
4.4.10.	Последователност от действия	24
4.4.11.	Предварителни изисквания	24
4.5.	Промяна на роли от администраторския панел	24
4.5.12.	Описание и приоритет	24
4.5.13.	Последователност от действия	24
4.5.14.	Предварителни изисквания	24
5.	Други Нефункционални Изисквания	24
5.1.	Архитектура на Приложението	24
5.2.	Начини за мащабиране	25
5.2.1.	Хоризонтално Мащабиране	25
5.2.2.	Вертикално Мащабиране	25
5.3.	Изисквания за Производителност	25
5.4.	Изисквания за Сигурност	25
5.5.	Атрибути за качеството на софтуера	25
6.	Други изисквания	26
6.1.	Аналитични Модели	26
6.1.1.	Class diagram на TaskFlow	26
6.1.2.	Entity relationship diagram на TaskFlow	27

1. Въведение

1.1. Цел

Този документ представя разработените функционални и нефункционални изисквания на проекта **TaskFlow** за управление на проекти като курсов проект за дисциплината *Практикум по практически проект*.

Това уеб приложение ще улесни екипната работа и проследяването на задачи. Системата предлага интуитивен интерфейс и функционалности, подходящи за организиране на както малки, така и по-големи екипни проекти.

1.2. Конвенции

За момента не се използват конвенции в документа.

1.3. За кого е предназначен този документ

Този документ е предназначен за членовете на екипа, който го разработва, студенти и преподаватели по дисциплината, за която се разработва. Може да послужи за отправна точка за по нататъшно доработване на продукта или промяна по съществуващите му функционалности.

1.4. Обхват на продукта

Целта на този софтуерен продукт е да осигури лесен и ефективен начин за управление на проекти и задачи в рамките на екипна работа. Системата позволява на потребителите да създават проекти, да организират и проследяват задачи, да задават отговорности, като по този начин подобрява координацията, планирането и проследяването на напредъка. Продуктът цели да улесни управлението на ежедневните дейности в различни типове организации – от малки екипи до по-големи корпорации.

1.5. Терминологичен речник

DB	Релационна база данни
Docker	Платформа, която позволява създаване, пакетиране и стартиране на приложения
Redis	in-memory (в паметта) ключ-стойност база данни, използвана основно за кеширане

1.6. Референции

1.6.1. Източници:

- <https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/jwt.html>
- <https://www.baeldung.com/spring-security-angular>

- <https://jwt.io/>
- <https://developer.okta.com/blog/2019/05/13/secure-spring-rest-api>
- <https://angular.dev/overview>

1.6.2. Хранилище на Приложението:

- <https://github.com/velk20/project-management-system>

2. Общо Описание

2.1. Обща Перспектива

Приложението **TaskFlow** е изготвено за дисциплината Практикум по практически проект, като идеята за него е породена от нуждата на централизирана и достъпна система за управление на проекти и задачи. В много екипи липсва ефективен инструмент за организиране на работния процес, което води до забавяния, липса на прозрачност и слаба координация. Създаването на тази система цели да предложи решение, което обединява основните функции за проектен мениджмънт в интуитивен и лесен за използване интерфейс. Освен това проектът служи като практическа демонстрация на умения в уеб разработката, архитектурното планиране и работа с бази данни.

2.2. Използвани Технологии

2.2.1. Java

Java е един от най-популярните и широко използвани програмни езици в света. Създадена от Sun Microsystems през 1995 година, Java бързо се утвърди като предпочитан избор за разработка на различни видове софтуер - от уеб приложения и мобилни приложения до корпоративни системи и интернет на нещата (IoT). Една от ключовите характеристики на Java е нейната платформа независимост, която позволява приложенията написани на този език да работят на всякакви устройства, стига те да имат Java виртуална машина (JVM). Също така Java е един от най-разпространените обектно ориентирани езици.

2.2.2. Maven

Maven е мощен инструмент за управление на проекти и автоматизация (на английски Automation Build Tool) за изграждането на Java проекти. Също така е най-използваният такъв до момента. Главните му конкуренти са "Gradle" и вече по-остарелият "Apache Ant".

Създаден е от Apache Software Foundation и предоставя стандартизиран начин за управление на проекта чрез използване на единна конфигурация, която улеснява изграждането, разположението и управлението на зависимостите.

Основната цел на Maven е да позволи на разработчика да разбере пълното състояние на усилието за разработка за най-кратък период от време. За да постигне тази цел, Maven се занимава с няколко проблемни области:

- Улесняване на процеса на изграждане
- Осигуряване на еднаква система за изграждане
- Предоставяне на качествена информация за проекта

- Насърчаване на по-добри практики за развитие

2.2.3. Spring Boot

Spring Boot е рамка (на английски: "Framework"), която улеснява разработката на приложения, производствени Spring-базирани приложения, които могат да бъдат изпълнени без нужда от външен уеб сървър. Spring Boot се основава на основната Spring Framework (която също така е най-използваната до момента рамка за Java), но включва множество допълнителни функции и автоматизации, които са вече конфигурирани и готови за използване, което значително опростяват стартирането и разработката на приложения.

Spring Framework е мощна и гъвкава платформа, която предлага множество модули за различни аспекти на приложението, като управление на данни, сигурност, уеб и други. Въпреки своята гъвкавост, Spring може да бъде сложен за конфигуриране и настройка, което може да доведе до дълъг и трудоемък процес на разработка.

Характеристики на Spring Boot:

- Създаване на самостоятелни приложения
- Директно вграждане на Tomcat, Jetty или Undertow (няма нужда от разполагане на WAR файлове)
- Осигурява самоуверени „начални“ зависимости, за да опростите конфигурацията на компилацията
- Автоматично конфигуриране и библиотеки на трети страни, когато е възможно
- Осигурява готови за производство функции като показатели, проверки на състоянието и външна конфигурация
- Не е нужно генериране на код и без изискване за XML конфигурация

2.2.4. Spring Data JPA

Spring Data JPA е мощна рамка, която опростява достъпа до базата данни в Spring Boot приложения, като предоставя слой за абстракция върху Java Persistence API (JPA). Той позволява безпроблемна интеграция с релационни бази данни, използвайки обектно-релационно преобразуване (ORM), елиминирайки необходимостта от шаблонни SQL заявки. С функции като автоматично генериране на заявки, поддръжка на JPQL и лесно управление на хранилище, Spring Data JPA подобрява ефективността при разработването на мащабируеми и високопроизводителни приложения. Това ръководство обхваща основните концепции, предимства и стъпка по стъпка внедряване на Spring Boot с JPA.

Основни функционалности на Spring Data JPA:

- Репозиториен интерфейс: Spring Data JPA предлага стандартен репозиториен интерфейс, който позволява на разработчиците да дефинират своите методи за достъп до данни чрез прости Java интерфейси. Това премахва необходимостта от писане на големи и сложни заявки към базата от данни.
- CRUD операции: Чрез дефинирането на интерфейси, които разширяват JpaRepository, Spring Data JPA автоматично генерира основните CRUD (Create, Read, Update, Delete) операции, като по този начин улеснява работата с база от данни.
- Именувани методи: Spring Data JPA позволява дефинирането на сложни заявки чрез именувани методи. Например, метод с име findByFirstName автоматично генерира заявка, която търси записи по поле firstName.

- JPQL и Native Queries: За по-сложни сценарии, Spring Data JPA поддържа използването на JPQL (Java Persistence Query Language) и native SQL заявки, които могат да бъдат дефинирани в методите на репозиторието.
- Пагинация и сортиране: Вградената поддръжка за пагинация и сортиране позволява олекотено взимане на данни, които са в голям брой.

2.2.5. Hibernate

Hibernate е инструмент за релационно картографиране на обекти (ORM) с отворен код, който предоставя рамка за обектно-ориентирани модели към релационни бази данни за уеб приложения.

Обектното релационно картографиране се основава на контейнеризирането на обекти и абстракцията, която осигурява този капацитет. Абстракцията дава възможност за адресиране, достъп и манипулиране на обекти, без да се налага да се обмисля как те са свързани с техните източници на данни.

Съпоставянето на Java класове към таблици на база данни се реализира чрез конфигурация на XML файл или чрез използване на анотации на Java. Когато използвате XML файл, Hibernate може да генерира скелетен изходен код за класовете за устойчивост. Това е спомагателно, когато се използват анотации. Hibernate може да използва XML файла или анотациите на Java, за да поддържа схемата на базата данни.

2.2.5.1. *Hibernate Query Language (HQL)*

Hibernate предоставя вдъхновен от SQL език, наречен Hibernate Query Language (HQL) за писане на подобни на SQL заявки срещу обекти с данни на Hibernate. Заявките за критерии се предоставят като обектно-ориентирана алтернатива на HQL. Criteria Query се използва за модифициране на обектите и предоставяне на ограничението за обектите. HQL (Hibernate Query Language) е обектно-ориентираната версия на SQL. Той генерира независими от базата данни заявки, така че няма нужда да пишете специфични за базата данни заявки. Без тази възможност промяната на базата данни ще изисква промяна на отделни SQL заявки, което ще доведе до проблеми с поддръжката.

2.2.5.2. *Интеграция*

Hibernate може да се използва както в самостоятелни Java приложения, така и в Java EE приложения, използващи сървлети, EJB сесийни компоненти и JBI сервисни компоненти. Може да се включи и като функция в други езици за програмиране. Например, Adobe интегрира Hibernate във версия 9 на ColdFusion (която работи на J2EE сървъри за приложения) с абстрактен слой от нови функции и синтаксис, добавен в CFML.

2.2.5.3. *Обекти и компоненти*

В жаргона на Hibernate, обектът е самостоятелен обект в постоянния механизъм на Hibernate, който може да бъде манипулиран независимо от други обекти. За разлика от това, даден компонент е подчинен на обект и може да бъде манипулиран само по отношение на този обект. Например обект „Албум“ може да представлява обект, но обектът „Песни“, свързан с обектите „Албум“, ще представлява компонент на обекта „Албум“, ако се приеме,

че „Песни“ могат да бъдат запазени или извлечени от базата данни само чрез обекта „Албум“. За разлика от J2EE, Hibernate може да превключва бази данни.

2.2.6. MySQL

MySQL е мощна и релационна база данни, която е open source (тоест, всеки може да контрибутира за имплементацията и) съгласно условията на GNU General Public License и също така се предлага под различни собственически лицензи. MySQL беше собственост и спонсориран от шведската компания MySQL AB, която беше закупена от Sun Microsystems (сега Oracle Corporation). През 2010 г., когато Oracle придоби Sun, Widenius разклони MySQL проекта с отворен код, за да създаде MariaDB.

MySQL има самостоятелни клиенти, които позволяват на потребителите да взаимодействат директно с MySQL база данни, използвайки SQL, но по-често MySQL се използва с други програми за внедряване на приложения, които се нуждаят от възможност за релационна база данни. MySQL е компонент на софтуерния стек за уеб приложения LAMP (и други), което е акроним за Linux, Apache, MySQL, Perl/PHP/Python.

MySQL се използва от много базирани на бази данни уеб приложения, включително Drupal, Joomla, phpBB и WordPress. MySQL се използва и от много популярни уебсайтове, включително Facebook, Flickr, MediaWiki, Twitter и YouTube.

Характеристики:

- **Лесен за използване:** MySQL се счита за лесен за използване сред RDBMS. Той работи с основен SQL и предвид неговата зрялост и възприемане, има изобилна налична документация.
- **Сигурен:** Зрелостта на MySQL също се поддава на сигурност. Актуализира се редовно, има жизнена общност от разработчици и поради широкото си приемане в предприятието, поради това базата данни бива постоянно актуализирана. Тези фактори се комбинират, за да направят MySQL стабилен и сигурен избор сред RDBMS.
- **Отворен код:** Изданието на общността на MySQL е готово за предприятия и се поддържа от GNU General Public License. За потребители, които искат достъп до справедлива собствена функционалност на MySQL без добавената цена, има други опции в рамките на екосистемата – като MariaDB – които могат да добавят подобни нива на функционалност и отвъд тях.
- **Мащабируеми:** MySQL е много мащабируем за RDBMS, с широк набор от опции, които не са обхванати в този блог, които позволяват настройка, персонализиране и подобряване на вашето MySQL изживяване.
- **Надежден:** MySQL е надежден - не само от гледна точка на данните, но и от гледна точка на разработката. Той е зрял, има редовни версии, корекции и утвърдена общност на разработчици, която работи с него. Това го прави безопасен избор в сравнение с по-новите, по-малко зрели опции за RDBMS.

2.2.7. Spring Boot Starter Validation

Spring Boot Starter Validation е част от екосистемата на Spring Boot, която улеснява интеграцията с Hibernate Validator, позволявайки лесно и ефективно валидиране на входните данни в Java приложенията.

Валидирането е основен аспект на всяко приложение. Независимо дали разработвате уеб API или сървис услуга, осигуряването на целостта на данните, които влизат във вашата система, е от решаващо значение. Без подходящо валидиране може да обработите непълни, непоследователни или злонамерени данни, което да доведе до грешки, уязвимости в сигурността или грешки по време на изпълнение.

Spring Boot, като удобна за разработчици рамка, се интегрира безпроблемно с Hibernate Validator, референтната реализация на стандарта Bean Validation (JSR 380). Само с няколко анотации можете да наложите стабилни правила за валидиране за вашето приложение.

Hibernate Validator е де факто стандартът за внедряване на API за валидиране на Bean. Той предоставя набор от предварително дефинирани ограничения, като @NotNull, @Size и @Email, което улеснява валидирането на потребителския вход във вашето приложение.

Тази библиотека се придържа към спецификацията JSR 380 на Java: Bean Validation 2.0, което означава, че е преносима и може да работи с всяка Java-базирана рамка. Тясната му интеграция със Spring Boot обаче го прави предпочитан избор за повечето разработчици.

2.2.8. Docker

Docker е платформа с отворен код, изградена на базата на контейнер технологията, която позволява изграждане, доставяне и стартиране разнообразни приложения, отделяйки ги от инфраструктурата.

В същността си платформата дава възможност да стартирате всякакви приложения, защитено изолирани в контейнери. Изолираната и безопасна среда позволява да работи едновременно с много контейнери на сървър. Олекотеният принцип на работа, който не изисква допълнително натоварване на хипервайзор или сложна конфигурация и настройка на среда за разработка, позволява да използвате максимума на хардуера.

Docker платформата съдържа три основни компонента:

- **Docker engine** – Процес, отговарящ за стартирането, управлението и визуализация на информация за различните контейнери, стартирани от него.
- **Docker client** – Инструмент, с който се осъществява връзката с Docker engine, предоставя интерфейс за комуникация между потребителите и Docker engine.
- **Docker Registry (Docker Hub)** – Основното хранилище на Docker. Предоставя достъп до вече готови за използване изображения на различен софтуер. При опит за създаване на контейнер от Docker engine, който е базиран на изображение, което не е налично локално, системата автоматично проверява в хранилището и ако там изображението е налично го сваля и създава контейнер от него. Съответно е възможно и обратното – при изграждане на един контейнер той да бъде запазен като изображение и да бъде качено към хранилището, за да бъде достъпно до други системи.

2.2.9. Spring Security

Spring Security е мощна и много адаптивна рамка за удостоверяване и контрол на достъпа. Това е де факто стандартът за защита на базирани на Spring приложения.

Spring Security е рамка, която се фокусира върху предоставянето както на удостоверяване, така и на оторизация на Java приложения. Подобно на всички проекти на

Spring, истинската сила на Spring Security се намира в това колко лесно може да бъде разширена, за да отговори на персонализираните изисквания.

Spring Security осигурява множество възможности за аутентикация, включително форм-базирана автентикация, автентикация с HTTP Basic и Digest, както и интеграция с OAuth и OpenID. Той поддържа ролево-базирана и гранулирана авторизация, което позволява дефинирането на сложни правила за достъп до ресурсите на приложението. Освен това Spring Security осигурява защита срещу общи уязвимости в уеб приложения, като Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), и Session Fixation.

2.2.10. JWT

JWT (на кратко от JSON Web Token) е JSON-базиран отворен стандарт за създаване на „жетони“, които съдържат определен брой твърдения. Например, един сървър може да генерира жетон, който съдържа твърдението „влязъл като администратор“ и достави тази информация до клиент. Клиентът тогава може да използва това означение да докаже, че е влязъл като администратор.

Жетоните са подписани от ключа на сървъра, така че сървърът да е в състояние да удостовери, че подписът е законен. Жетоните са проектирани да бъдат компактни, URL-безопасни и използвани най-вече в уеб браузър единично влизане контекст.

JWT обикновено се използват, за да се предаде самоличността на регистрирани потребители между доставчик на идентичност и доставчик на услуга, или всякакъв друг вид твърдения, както се изисква от бизнес процесите. Токените могат да бъдат потвърдени и криптирани.

JWT се състои от три части: заглавие (header), полезен товар (payload) и подпис (signature). Заглавието обикновено съдържа информация за типа на токена и алгоритъма на подписване. Полезният товар съдържа твърдения (claims) за енциклирана информация, която може да включва потребителски данни и други метаданни. Подписът е резултат от криптографски хеш на заглавието и полезния товар, който гарантира, че данните не са били променени.

2.2.11. Angular

Angular е една от най-популярните JavaScript технологични рамки. Тя е създадена с основната цел да опрости разработката на уеб и мобилните приложения, чрез похвати като end-to-end tooling, декларативни темплейти и dependency injection. Angular предлага богата и добра колекция от компоненти, които опростяват процеса по набиране, корекция и употреба на код. Технологичната рамка намалява количеството време, което разработчиците прекарват в писане на JavaScript код, което ускорява процеса по завършване на монотонните задачи и ви дава времето, което ви е необходимо да се фокусирате повече върху функционалностите и дизайна на вашето творение.

Angular използва в себе си TypeScript, който е строго типизиран, компилиран, ООП език за програмиране. Той се основава на добре познатия компютърен език - JavaScript. Typescript е език за програмиране от високо ниво с безплатен отворен код, който е разработен и се поддържа от Microsoft.

Този компютърен език за програмиране може да се приеме като подобрена версия на JavaScript, в която обаче са предоставени по-добри инструменти за всякаква работа и допълнителни функции.

Основната цел на Angular е създаването на Single Page приложенията, които са много популярни сред уеб разработчиците и разработките на различни сайтове. При тях няма отделни страници, а информацията се появява динамично на една уеб страница след действията на потребителите. Този тип приложения предоставят по-добро потребителско преживяване, поради което са и толкова предпочитани.

2.2.12. JUnit

JUnit е рамка за автоматизирано тестване, която се използва широко в разработката на Java приложения. Тя е инструмент за писане и изпълнение на тестов код, който проверява дали отделни части (единици) на кода работят според очакванията. JUnit позволява на разработчиците да изолират и тестват функции и методи, което осигурява надеждна и поддръжка на кода.

Ключови понятия:

- **Unit testing:** Проверка на отделни модули или функции на кода, за да се уверите, че те функционират правилно самостоятелно.
- **Test case:** Конкретна проверка, която се изпълнява по време на тестване, за да се определи дали даден входни данни водят до очаквания резултат.
- **Assertions:** Методи, които се използват в тестов код, за да се сравняват очакваните резултати с действителните резултати и да се маркират неуспешни случаи, ако те са различни.
- **Test-Driven Development (TDD):** Методология, при която тестов код се пише преди реализирането на функцията, която трябва да бъде тествана.

Плюсове от използване на JUnit:

- Повишава качеството на кода: Чрез ранно и редовно тестване с JUnit, можете да идентифицирате и отстраните бъговете, преди да се появят в продукционната среда.
- Улеснява поддръжката на кода: Тестването помага за поддържане на надеждност и структура на кода, като предотвратява внезапни промени и неизбежни грешки.
- Увеличава производителността: Автоматизирането на тестването с JUnit може да намали времето, необходимо за идентифициране на бъговете и отстраняването им.
- Осигурява надеждност: JUnit помага за гарантирането, че кода винаги е стабилен и работи правилно при различни ситуации.

2.2.13. Mockito

Mockito е популярна Java библиотека за създаване на мок обекти в процеса на тестване. Тя е предназначена да улесни писането на тестове, като позволява създаването на заместители на реалните обекти, които могат да имитират поведението на тези обекти по време на тестовете. Това позволява тестването на отделни компоненти в изолация, без нуждата от зависимост към реалните им имплементации.

Тя предоставя интуитивен и лесен за използване API, което прави процеса на създаване и конфигуриране на мок обекти бърз и ефективен. Основните функционалности на Mockito включват създаването на мок обекти, дефиниране на тяхното поведение, проверка на взаимодействията между обектите и засичане на извиквания на методи.

Mockito позволява създаването на мок обекти чрез анотацията `@Mock` или чрез статичния метод `Mockito.mock(Class<T> classToMock)`. Тези мок обекти могат да имитират поведението на реалните обекти, като се задават очаквания за извикване на методи с метода `when().thenReturn()`. Mockito предоставя възможност за проверка на взаимодействията между обектите чрез метода `verify()`, който гарантира, че определени методи са били извикани с конкретни параметри.

2.2.14. Git

Git е система за контрол на версиите, която позволява на разработчиците да проследяват и управляват промените в изходния код на проектите си. Тя е разпределена система, което означава, че всеки разработчик има пълно копие на целия проект и неговата история, съхранено локално на своята машина. Това позволява бързо и лесно възстановяване на предишни версии на проекта, работа по различни функционалности и клонове на проекта, както и колаборация между множество разработчици без нужда от постоянна връзка с централен сървър. Git предоставя инструменти за създаване на клонове, сливане на код и разрешаване на конфликти, което улеснява процеса на разработка и интеграция в екипна среда.

Git също така предлага висока степен на сигурност и надеждност, тъй като съхранява хронологията на промените и осигурява защита срещу загуба на данни или неволни грешки. Системата се използва широко в индустрията и е сърцето на много DevOps процеси и практики, като непрекъсната интеграция и непрекъсната доставка (CI/CD). Благодарение на своята гъвкавост и възможности за колаборация, Git се превърна в стандарт за управление на кода в почти всички съвременни софтуерни проекти.

2.2.15. GitHub

GitHub е платформа за хостване на код, базирана на Git, която предлага разнообразни инструменти за управление на проекти и колаборация. Тя предоставя онлайн репозитории, където разработчиците могат да съхраняват, споделят и управляват своя изходен код. GitHub комбинира мощта на Git със свои собствени функционалности, като управление на задачи, следене на проблеми (issues), обсъждания, вградени механизми за ревю на код и много други.

Една от основните силни страни на GitHub е неговата способност да улеснява колаборацията между разработчици от цял свят. Платформата предлага механизъм за "pull requests," който позволява на разработчиците да предлагат промени в даден проект и да участват в процеса на ревю и одобрение на кода. Това прави GitHub особено подходящ за екипни проекти и open-source разработки, където различни разработчици могат да допринасят със своите промени и подобрения.

Освен това, GitHub предлага интеграции с множество други инструменти и платформи за разработка и DevOps, което го прави универсален инструмент за съвременния разработчик. Платформата включва и автоматизирани работни потоци (GitHub Actions), които подпомагат автоматизацията на процеси като тестване, непрекъсната интеграция и непрекъсната доставка (CI/CD).

Избрах GitHub за този проект, защото платформата предоставя сигурна и надеждна среда за съхраняване на кода, същевременно предлага лесен и ефективен начин за управление на версиите и колаборация. Наличието на богати и в повечето случаи

безплатни функционалности, като система за следене на проблеми и интеграция с CI/CD инструменти, я прави идеална за реализирането на проекти от различен мащаб и сложност.

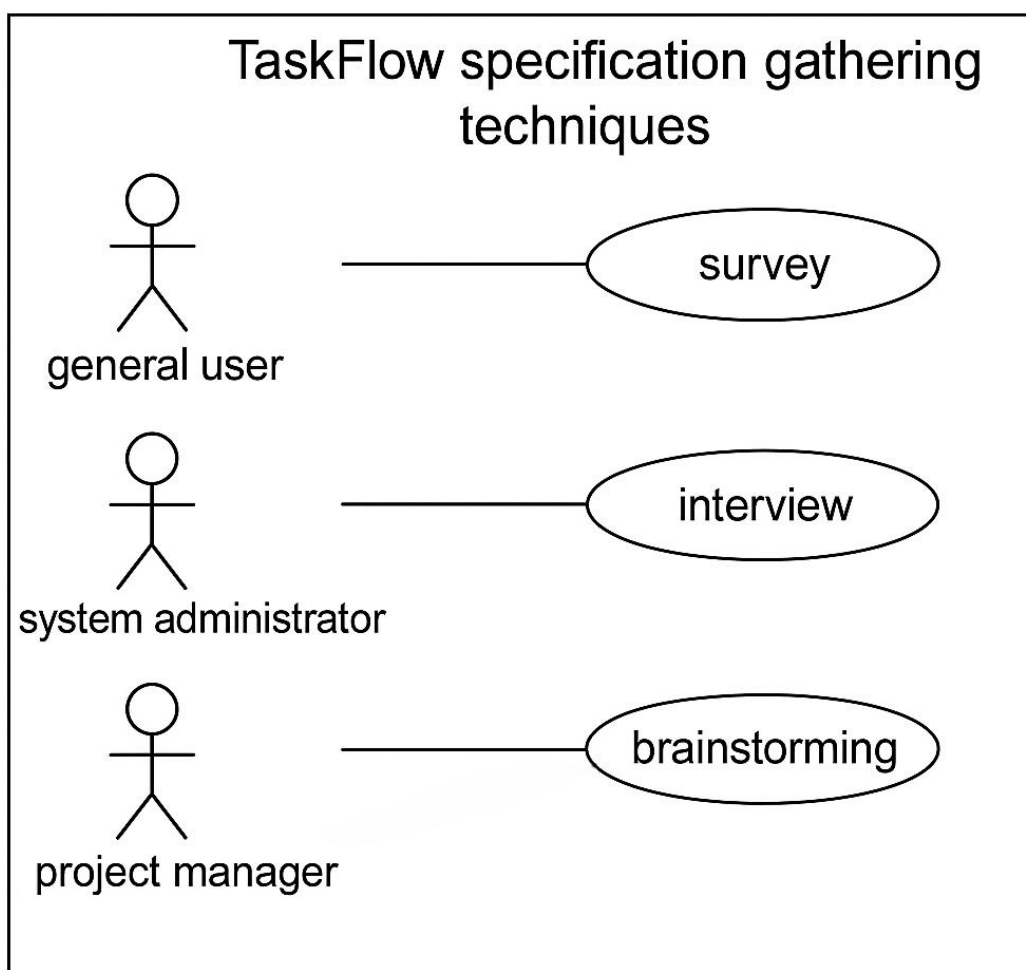
2.3. Характеристики на продукта

Системата **TaskFlow** предоставя възможност на потребителя да:

- вижда всички проекти, в които участва;
- вижда всички задачи, които са от проектите, в които участва;
- създава нови проекти;
- създава нови задачи в проекти;
- да проследява дадена задача, до какъв статус е;
- да пише коментари по дадена задача;
- да променя данните за своя профил;
- да сменя текущата си парола с нова.

2.4. Видове Потребители и Характеристики

Приложението **TaskFlow** е предназначено за организации, които се стремят да подобрят качеството си на работа. Системата за момента работи само на уеб браузъри. С диаграмата са представени всички заинтересовани страни и начините на извличане на изисквания от тях.



2.5. Операционна среда

TaskFlow е уеб базирано приложение, което е разработено за всички видове браузъри. Използва MySQL DB. Разработено е с Java, Spring Boot и Angular за потребителския интерфейс.

2.6. Ограничения за Дизайна и Реализацията

Интернет връзката е ограничение за приложението, защото без нея не може да бъдем сигурни, че разполагаме с всички ресурси, които използваме от интернет, особено в потребителския интерфейс. Такива ресурси са:

- иконки;
- изображения;
- външни скриптове за анимации;
- външни скриптове за стилизация;

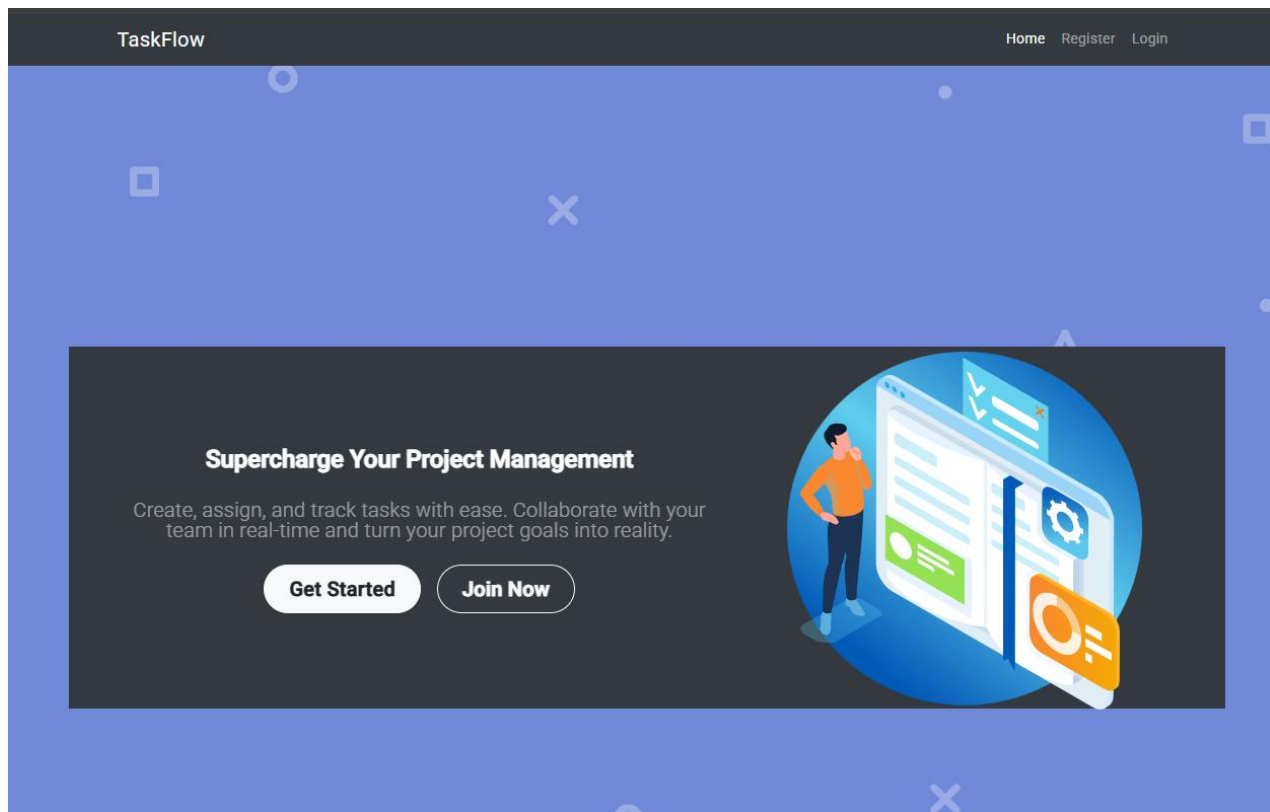
2.7. Предположения и зависимости

Една зависимост на приложението е, че е нужен браузър, за да бъде отворена платформата. Ако устройството няма инсталиран браузър, то няма да може да отвори приложение.

3. Външни Изисквания към Интерфейса

3.1. Начален екран

Когато потребителят отвори приложението *TaskFlow*, се зарежда началният екран, който няма много функционалности, докато потребителят не се регистрира и/или влезе в своя профил. Тази начална страница съдържа кратко въведение и девиз на приложението. Също така имаме бутоните “Get Started”/”Login”, които ни препращат към страницата за вход на потребители, а бутоните “Join Now”/”Register” служат за препращане на потребителя към страницата за регистрация.



Начална страница(фиг.1)

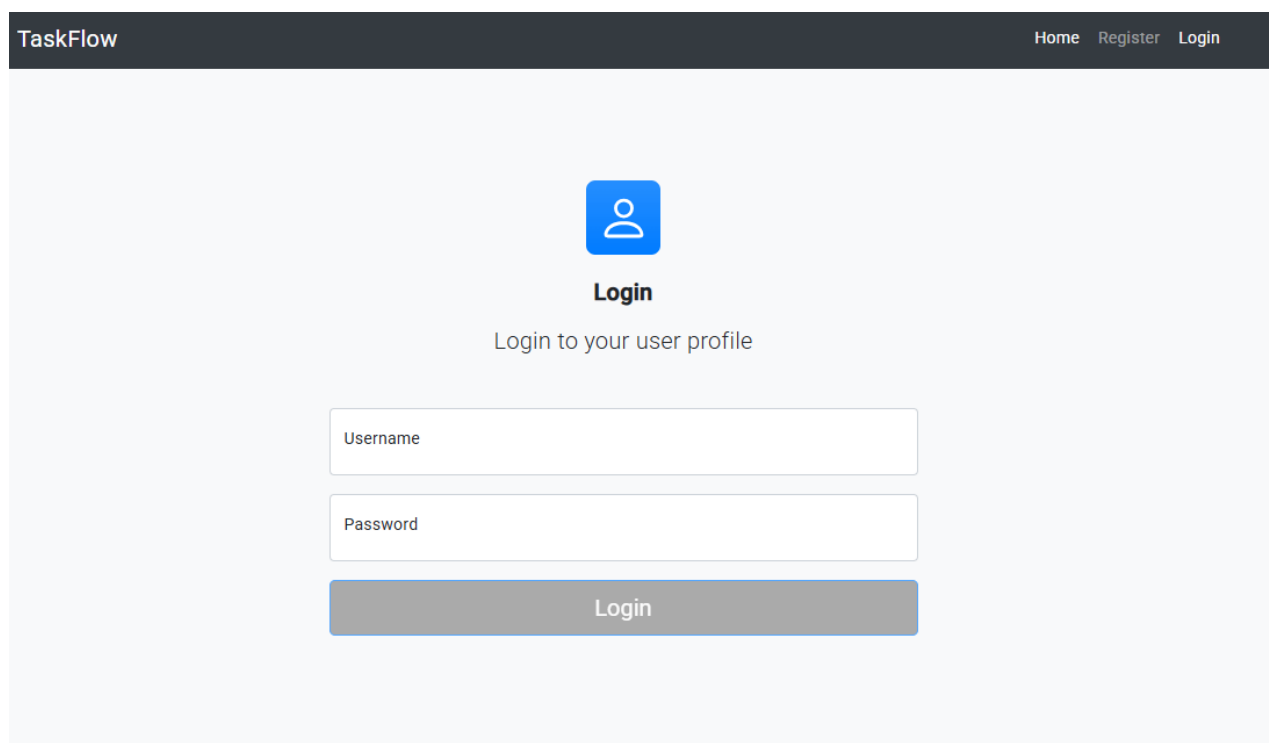
Когато гост на нашето приложение влезе за първи път в нашия софтуер е препратен директно към началната страница, която представлява съобщение за добре дошли, слоган за продукта и 2 бутона за регистрация и влизане, които съответно препращат към съответните страници за регистрация и вход.

Страница за регистрация (фиг.2)

В страницата за регистрация потребителя е задължен да попълни всички полета за създаване на профил, които са:

- Username (потребителско име): уникално име на всеки потребител, който ще служи като уникално средство за разпознаване на потребителите и вход за системата.
- Password (потребителска парола): парола на потребителя, която ще служи за вход в системата.
- Confirm Password (потвърждаване на паролата): това поле служи като добавка към паролата, че потребителя е сигурен каква трябва да бъде неговата парола.
- First Name (първо име): първото име на потребителя, който иска да се регистрира.
- Last Name (фамилно име): фамилното име на потребителя, който иска да се регистрира.
- Email (електронна поща): уникална електронна поща на потребителя.

Бутонът “Register” няма да бъде активен докато потребителя не въведе правилно всички искани от него данни за регистрация.



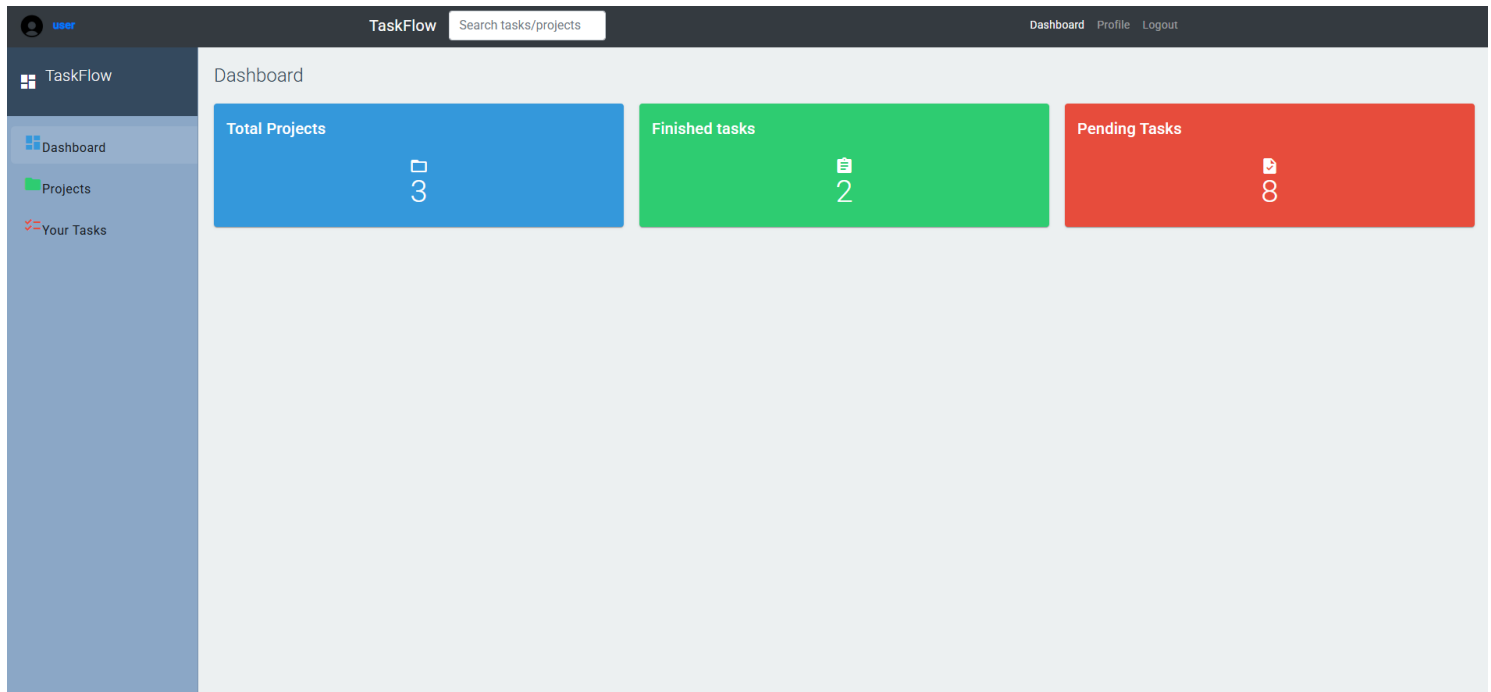
The screenshot shows the login interface of the TaskFlow application. At the top, a dark navigation bar contains the 'TaskFlow' logo and links for 'Home', 'Register', and 'Login'. The main body of the page is a light gray color. Centered on the page is a blue square icon representing a user profile. Directly below the icon is the word 'Login' in bold, followed by the text 'Login to your user profile'. Underneath this text are two white input fields with gray borders, labeled 'Username' and 'Password'. At the bottom of the form is a wide, gray button with the text 'Login' in white.

Страница за вход в системата (фиг.3)

Страницата за вход съдържа в себе си само 2 полета, които са потребителско име и парола. При опит за вход с грешни данни на потребителя ще бъде показва съобщение за грешни данни за вход.

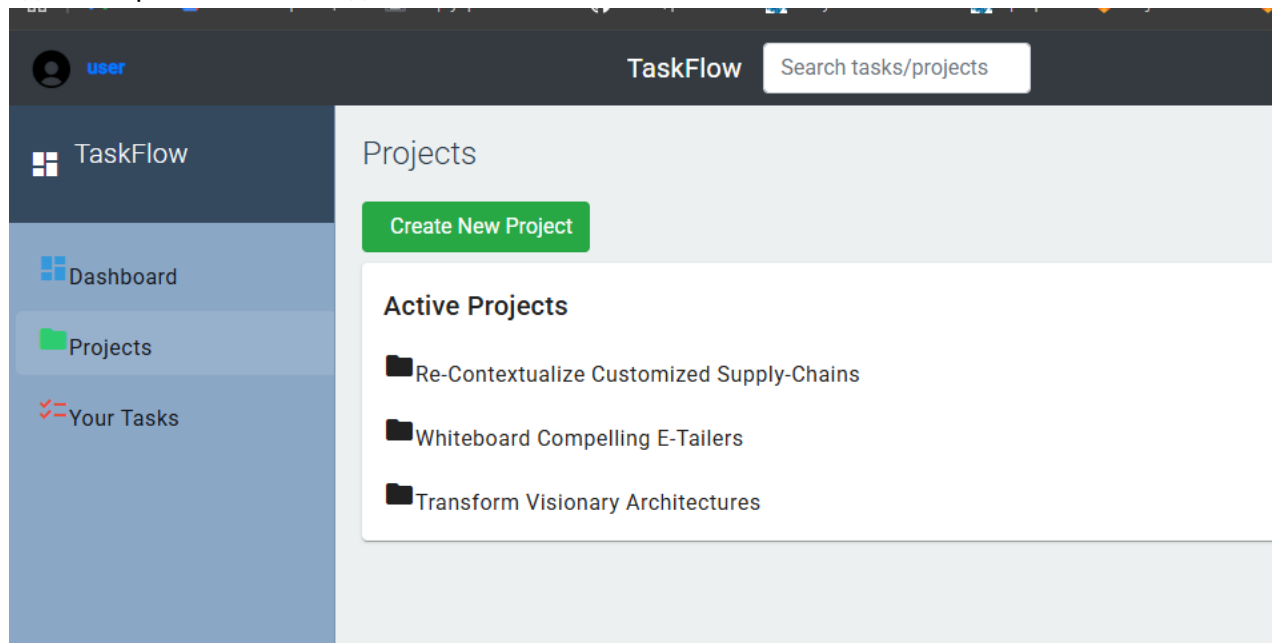
3.2. Начален екран за влезнали потребители

Когато потребителят влезе в системата със своят профил, той бива препратен към таблото за управление. Потребителят има възможност да търси задачи или проекти чрез техното име в полето за търсене, намиращо се горе в навигационния панел. А главната цел на този екран е потребителя да има лесен достъп до неговите проекти, задачи и да има бърз обзор на броя проекти, в които участва, задачите които е успял да приключи и задачите, по които все още се работи по тях.



Табло за управление (фиг. 4)

Страницата с проектите служи на потребителя, да покаже в кои проекти той е част и до кои проекти той има достъп.



Страница на проектите (фиг. 5)

ID	Subject	Type	Status	Assignee	Author	Updated On
1	Title 1	Bug	In Progress	User Userov	User Userov	2.01.25 г., 2:00 ч.
11	Title 11	Story	Resolved	User Userov	User Userov	2.01.25 г., 2:00 ч.
21	Title 21	Feature	New	User Userov	User Userov	2.01.25 г., 2:00 ч.
31	Title 31	Bug	Closed	User Userov	User Userov	22.05.25 г., 10:30 ч.
41	Title 41	Epic	Resolved	User Userov	User Userov	2.01.25 г., 2:00 ч.
51	Title 51	Bug	In Progress	User Userov	User Userov	2.01.25 г., 2:00 ч.
52	Title 52	Bug	Closed	User Userov	User Userov	22.05.25 г., 10:31 ч.
53	Title 53	Bug	In Progress	User Userov	User Userov	2.01.25 г., 2:00 ч.
54	Title 54	Bug	In Progress	User Userov	User Userov	2.01.25 г., 2:00 ч.
55	Title 55	Bug	In Progress	User Userov	User Userov	2.01.25 г., 2:00 ч.

Страница на потребителски задачи (фиг. 6)

На фиг.6 може да видим страницата със задачи за влезналият потребител. Тук ще бъдат показани всички задачи, които са обвързани с дадения потребител. Като например ако той е създател на някоя задача или му е зададена дадена задача за работа.

Back

Task Details

Updated at: 2.01.25 г., 2:00 ч.

Task ID

11

Title

Title 11

Description

Description 11

Status

Resolved

Type

Story

Assignee

user

Creator

user

Created At

1.01.25 г., 2:00 ч.

Save Changes

Comments

frankm(4.05.25 г., 0:39 ч.)
Expert off physical. Teach parent exist out result official whole. Product land save adult. Long employee indeed opportunity. Same lose bill debate force production different forget.

frankm(4.01.25 г., 6:28 ч.)
Ask subject personal think. Enough data home wall. Off world young now. Public amount dinner off ball sure. Food contain score act some against table. Want of though like unit crime.

Add a Comment

Write your comment here...

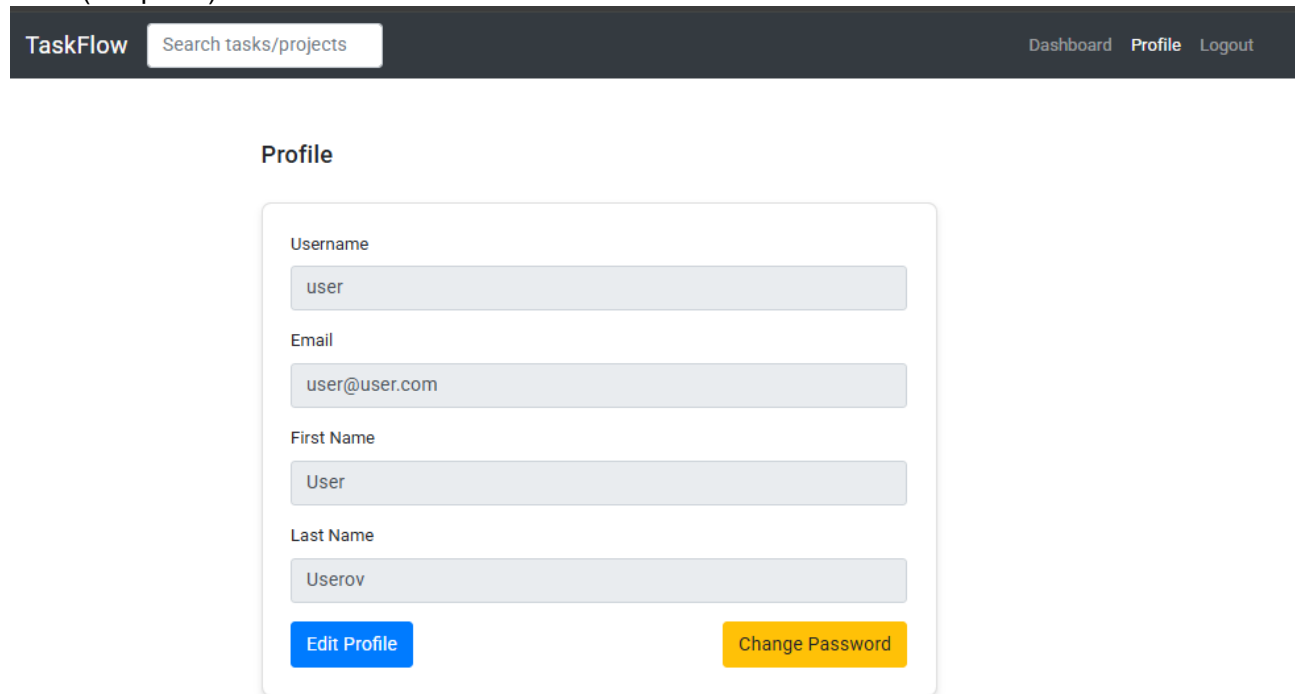
Add Comment

Подробна страница за задачи (фиг. 7)

На фиг.7 виждаме примерна визуализация на една задача, като потребителят може да променя данните за дадена задачи, като достатъчно условие е той да бъде част от проекта, в който е създадена задачата. Всеки потребител има право да добавя коментари за дадена задача. Ако потребителя е собственик на коментар, той също има право да променя съдържанието на коментара и да го изтрива.

3.3. Страница за редактиране на потребители

Потребителят има право да редактира своята информация за профила (на фиг.8), като потребителско име, имейл, първо и второ име, както и да сменя своята текуща парола с нова (на фиг.9).



TaskFlow Search tasks/projects Dashboard Profile Logout

Profile

Username
user

Email
user@user.com

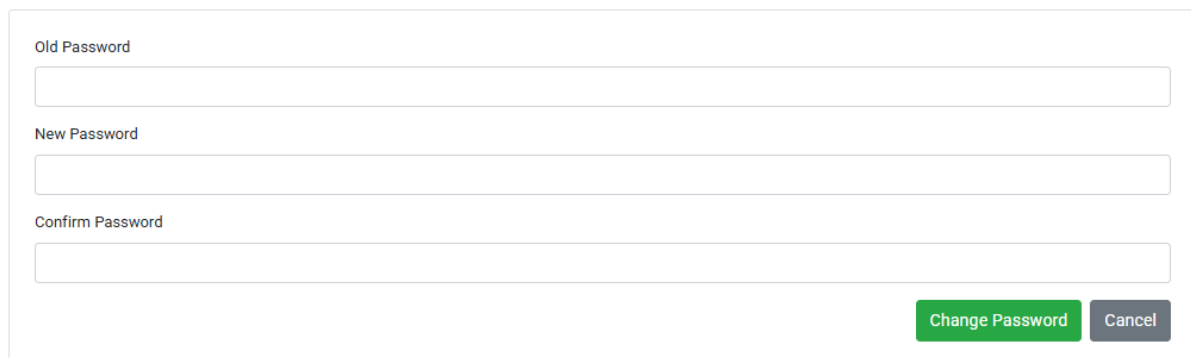
First Name
User

Last Name
Userov

Edit Profile Change Password

Страница за редактиране на потребител (фиг. 8)

Change Password



Old Password

New Password

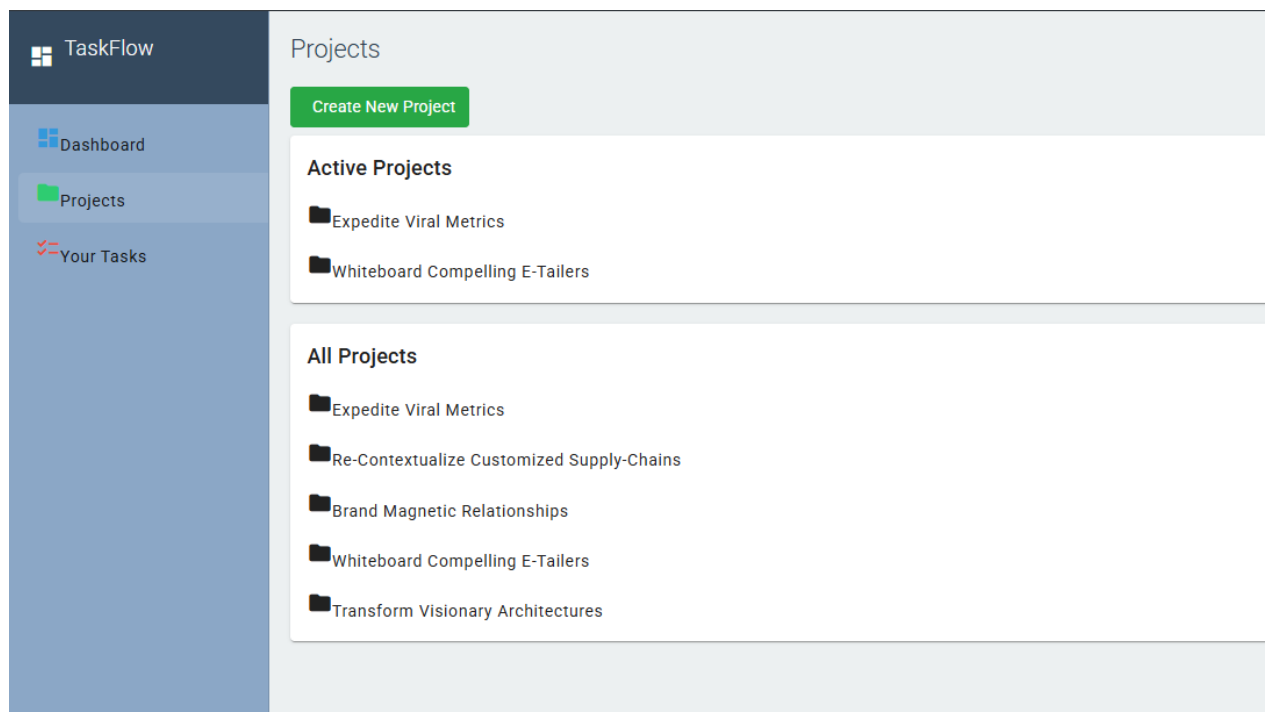
Confirm Password

Change Password Cancel

Страница за промяна на потребителска парола (фиг. 9)

3.4. Потребители с администраторски права

Потребителите, които имат администраторски права в платформата имат право да преглеждат всички проекти, дори и да не са част от тях (на фиг.10).



Страница с проекти, при админски права (фиг.10)

Администраторите имат право за достъп и до административна страница, която служи за преглеждане на всички регистрирани потребители в системата, дали техният профил е активен и с каква роля е даденият потребител (Admin/User). От тази страница админите могат да дават права на други потребители (да дават админски права), също така могат да деактивират акаунти на други потребители, след което те няма да могат да влизат в системата. Имаме и бутон за *анулиране на достъпа на всички потребители*, чиято цел е да прекрати текущия достъп на всички потребители до системата (те ще бъдат изхвърлени от акаунтите си и помолени да влезнат отново в тях).

Users Administration

[Invalidate All Users access](#)

ID	Username	Role	Status	Actions
1	admin	ADMIN	Active	Change Role Disable
2	user	USER	Active	Change Role Disable
3	alices	USER	Active	Change Role Disable
4	bobj	USER	Active	Change Role Disable
5	carolw	ADMIN	Active	Change Role Disable
6	davidb	USER	Active	Change Role Disable
7	evaj	USER	Active	Change Role Disable
8	frankm	ADMIN	Active	Change Role Disable
9	graced	USER	Active	Change Role Disable
10	henryw	USER	Active	Change Role Disable

Previous Page 1 of 1 Next

Административна страница (фиг.11)

3.5. Хардуерни Интерфейси

Приложение *TaskFlow* няма определени хардуерни изисквания, няма и директни хардуерни интерфейси.

3.6. Софтуерни интерфейси

Приложението комуникира с DB, за да съхранява информация за проекти, задачи, коментари и потребители. Приложението използва и Redis като Docker контейнер. Използван за съхранение на дата, до която потребителите имат възможност да са вътре в системата, след това време те ще бъдат изхвърлени. Това е с цел защита на приложението от външни атаки.

3.7. Комуникационни интерфейси

Комуникацията между сървърната част на приложението и клиентската не е тясно свързана. Към сървърът може и да се свържат външни клиенти (като мобилно приложения, десктоп приложения) стига те да имат достъп до HTTP протокол.

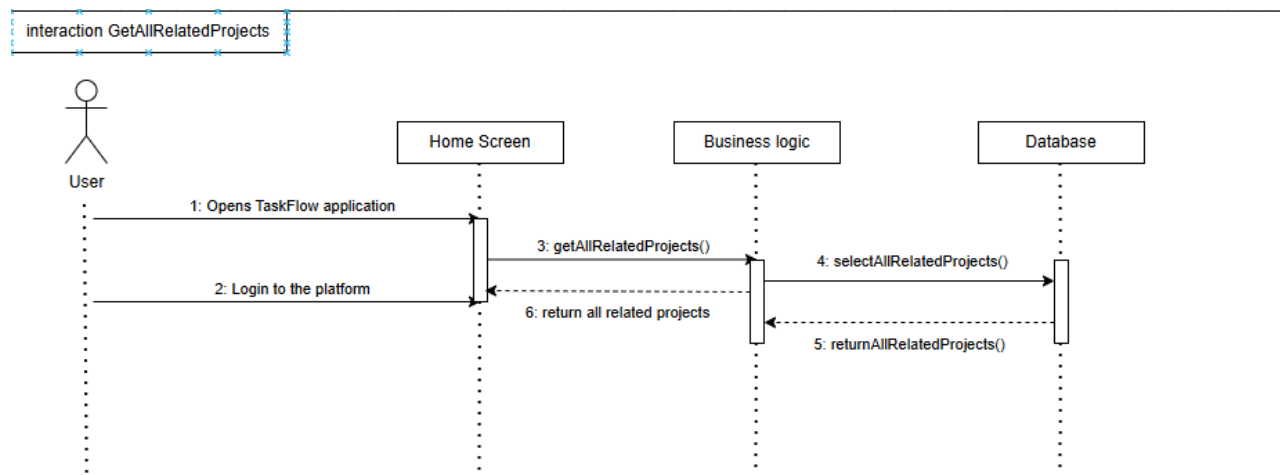
4. Функционалности на Системата

4.1. Взимане на проектите, в които потребителя участва

4.1.1. Описание и приоритет

Взимане и визуализация на потребителските проекти
Приоритет: висок

4.1.2. Последователност от действи



4.1.3. Предварителни изисквания

Изискване 1: потребителят да е влезнал в платформата

Изискване 2: потребителят да бъде на началната страница

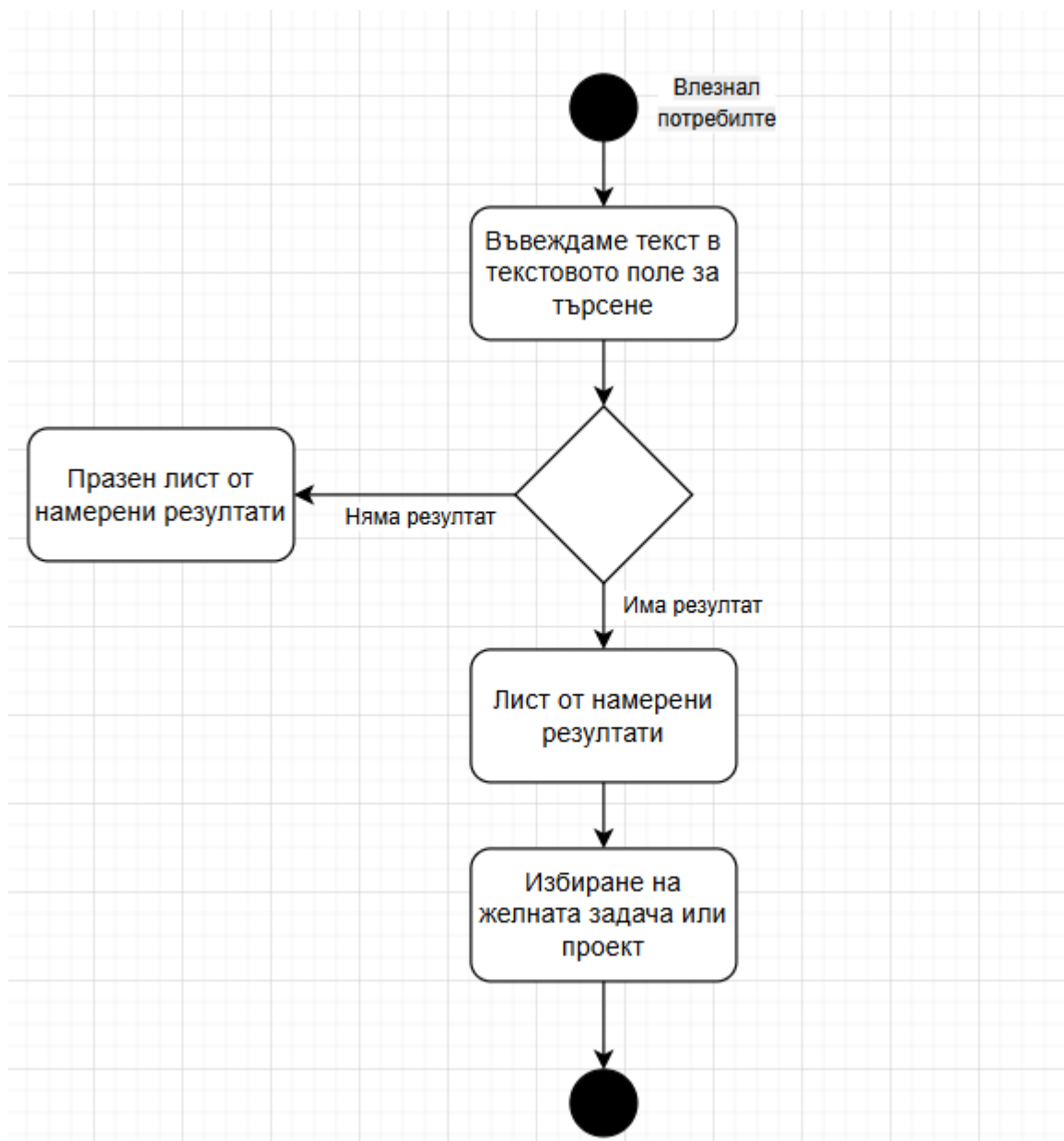
4.2. Търсене на задача или проект по име

4.2.4. Описание и приоритет

Търсене на задача или проект по име в цялата система.

Приоритет: висок

4.2.5. Последователност от действия



4.2.6. Предварителни изисквания

Изискване 1: потребителят да е отворил началният екран като влезнал протребител

Изискване 2: потребителят да напише името, което иска в текстовото поле за търсене

4.3. Създаване на нова задача

4.3.7. Описание и приоритет

Избиране на страницата със задачите и кликване на бутона „Create New Task”

Приоритет: висок

4.3.8. Нужни данни за нова задача

	Полета	Описание
1.	Заглавие	Заглавието на новата задача (задължително, уникално)
2.	Описание	Подробно описание на задачата (незадължителен)
3.	Статус	Статус на задача. Налични опции: New, Reopened, Feedback, In Progress, Resolved, Closed, Rejected, Duplicate
4.	Тип на задачата	Типа на задача. Налични опции: Feature, Story, Bug, Epic, Test, Idea
5.	Проект, в който задачата да бъде добавена	Името на проекта, в който ще бъде записана задачата. Налични опции: Списък от проектите, в които потребителя участва
6.	Потребител, който ще отговаря за задачата	Името на потребителя, който ще отговаря за задачата. Налични опции: Списък от потребители, които отговарят на търсеното име

4.4. Промяна на данните за потребител

4.4.9. Описание и приоритет

При влезнал потребител, избиране на страницата "Profile", от навигационното меню

Приоритет: среден

4.4.10. Последователност от действия

1. Потребителя се намира в профилната страница
2. Избиране на бутона "Edit Profile", за да отключи формата за редактиране
3. Промяна на данните
4. Запазване на промените от бутона „Save Changes”

4.4.11. Предварителни изисквания

Изискване 1: потребителят да е влезнал в системата

4.5. Промяна на роли от администраторския панел

4.5.12. Описание и приоритет

Промяна на роля на даден потребител от администратор

Приоритет: среден

4.5.13. Последователност от действия

1. Админа е в администраторската страница
2. Админа намира дадения потребител за редактиране
3. Избиране на бутона „Change Role”
4. Смяна на ролята и запазване на промените

4.5.14. Предварителни изисквания

Изискване 1: потребителят да е влезнал в системата

Изискване 2: потребителят е да има Админски права

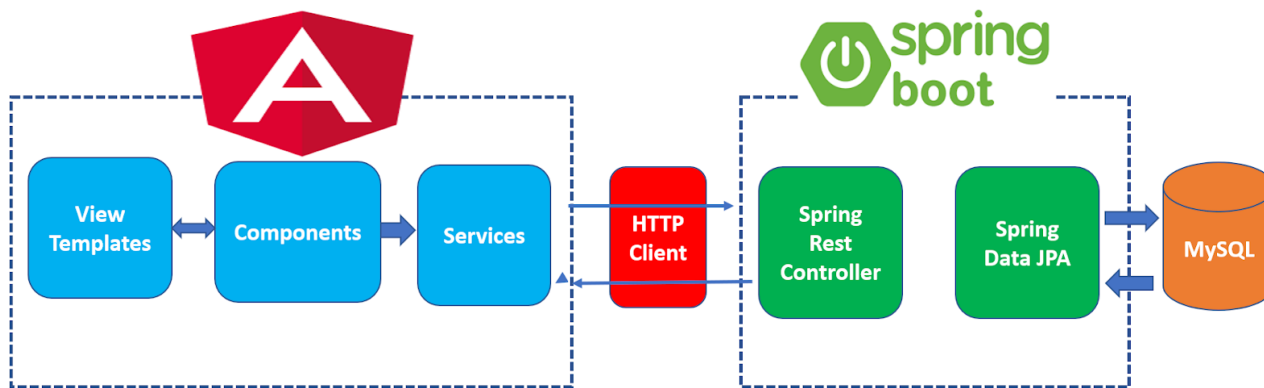
5. Други Нефункционални Изисквания

Изискванията в тази секция представят в детайли взаимодействието на потребителя с уеб приложението и показват ограниченията, поставени върху производителността.

5.1. Архитектура на Приложението

Архитектурата на приложението е от типа клиент-сървър. Първата част е Spring Boot приложение, което служи като доставчик на информацията искана от нашия клиент. Клиентите може да бъдат различни и разнообразни, стига те да поддържат HTTP протокола.

В нашето приложение имаме предоставен клиент под формата на Angular приложение, което служи на потребителите, които нямат подготвен клиент, който да взема информацията от нашия доставчик.



5.2. Начини за мащабиране

5.2.1. Хоризонтално Мащабиране

Хоризонталното мащабиране включва добавяне на повече инстанции на приложенията. Angular частта може да се хоства като статични файлове на CDN или уеб сървър и лесно да се разпространява в много сървъри или точки по света. Spring Boot приложението може да се стартира на множество сървъри или контейнери и да се разпредели трафикът между тях чрез load balancer, като например NGINX, HAProxy или Kubernetes Ingress.

5.2.2. Вертикално Мащабиране

Вертикалното мащабиране може да се постигне чрез увеличаване на ресурсите на един сървър – като процесор, памет или дисково пространство. Това е по-просто решение в началото, но не е толкова устойчиво в дългосрочен план, колкото хоризонталното мащабиране.

5.3. Изисквания за Производителност

Търсенето на задачи и проекти по име трябва да бъде бързо и ефективно. Изготвянето на нови задачи трябва да бъде лесно и удобно за потребителя. Бърз достъп до текущите задачи също е с голям приоритет.

5.4. Изисквания за Сигурност

За да се използва приложението **TaskFlow**, задължителна е нуждата от регистрация и вход в приложението. Сензитивна информация като пароли са криптирани при съхранението им в базата данни.

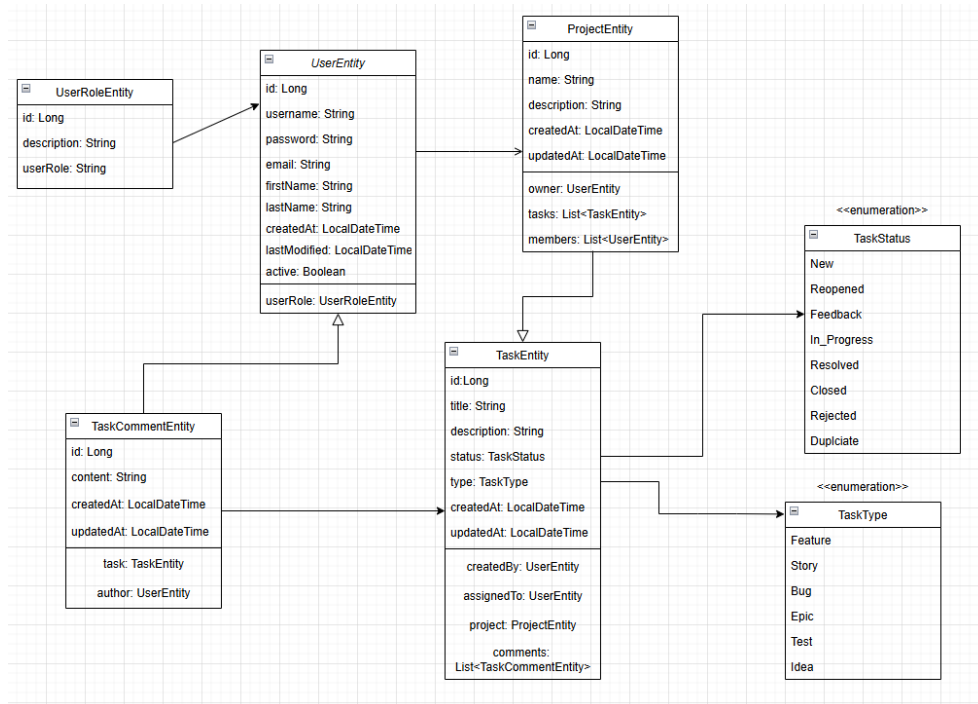
5.5. Атрибути за качеството на софтуера

Надеждността на системата трябва да се състои в това да бъде лесна за използване, с удобен потребителски интерфейс. Системата трябва да работи по всяко време, като в това не спада липса на интернет. Мерната единица за това е 1000 часа работа по време на тестване, като трябва да работи повече от 98% от времето. Трябва и да може да бъде разширявано лесно, като кода му трябва да бъде написан по начин, който е добър за лесното имплементиране на нови функционалности. За момента приложението работи само на уеб браузъри.

6. Други изисквания

6.1. Аналитични Модели

6.1.1. Class diagram на TaskFlow



Диаграмата описва главните класове, които се грижат за бизнес логиката. UserEntity представя потребителите в системата. Всеки потребител има уникално id, потребителско име, парола, имейл, собствено и фамилно име, дата на създаване и последна модификация, както и флаг дали е активен. Всеки потребител има роля, която се дефинира чрез връзка към класа UserRoleEntity. Един потребител може да създаде множество проекти, като се явява техен собственик, както и да бъде участник в различни проекти. Освен това потребител може да създава задачи и да бъде определян за изпълнител на задачи. Също така може да оставя коментари към задачите, като е автор на съответните записи в TaskCommentEntity.

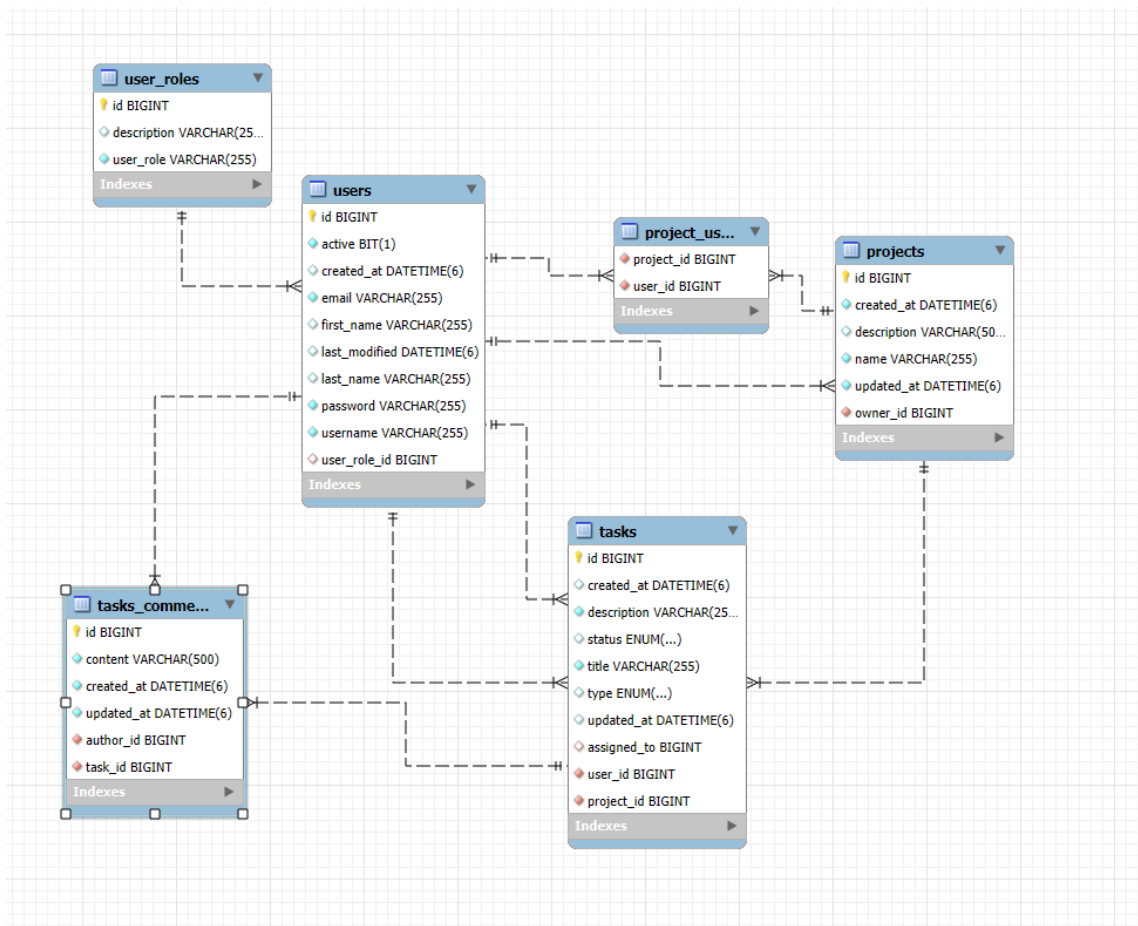
UserRoleEntity съдържа информация за потребителските роли. Всеки запис има уникално id, текстово описание и стойност на самата роля. Една роля може да бъде асоциирана с много потребители, но всеки потребител има само една роля. ProjectEntity представя един проект, който има уникално id, име, описание, дата на създаване и дата на последна промяна. Всеки проект има собственик, който е потребител и е дефиниран чрез връзка към UserEntity. Проектът съдържа списък със задачи и списък с членове, които също са обекти от тип UserEntity. Един проект може да има много задачи и много членове.

TaskEntity е сърцевината на системата и описва отделните задачи. Всяка задача има уникален идентификатор, заглавие, описание, статус и тип, както и дати на създаване и последна промяна. Връзките показват кой е създал задачата и на кого е възложена – и двете са потребители. Освен това задачата е част от конкретен проект. Една задача може да има много коментари. Статусът и типът на задачата са представени чрез enums – TaskStatus и TaskType – и са обвързани по референтен начин, без да се съдържат като обекти.

TaskCommentEntity представлява коментар към конкретна задача. Всеки коментар има идентификатор, съдържание, дата на създаване и дата на промяна. Към него има връзка към задачата, към която е написан коментарът, и връзка към потребителя, който е негов автор. Така всеки коментар знае за коя задача е и кой го е написал. TaskStatus е enum, който съдържа възможните състояния на задачата, като например New, Reopened, Feedback, In_Progress, Resolved, Closed, Rejected и Duplicate. Това са предефинирани стойности, които се използват от TaskEntity за обозначаване на текущото състояние.

TaskType е друг enum, който дефинира вида на задачата – Feature, Story, Bug, Epic, Test или Idea. Това позволява по-ясна категоризация на задачите в проекта.

6.1.2. Entity relationship diagram на TaskFlow



Съдържа 6 таблици:

- Users таблицата съдържа данните за всички потребители, като име, имейл, роля и парола. Паролата се пази в базата данни използвайки **BCrypt** хеширане.
- Users_Roles таблицата съдържа името и описанието на всички роли в нашата система.
- Project_User таблицата е допълнителна свързваща таблица за потребители и проектите, в които те участват.
- Таблицата Projects пази данните за всички проекти, като тяхното име, описание, създадел на проекта.
- Таблицата Tasks пази данните за всички задачи, като тяхното име, описание, статус, тип, за кого е предназначена задачата и в кой проект тя участва.
- Таблицата Tasks_Comments държи всички коментари в нашата система, като има външен ключ към съответната задача, на която принадлежи коментара.