МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 01 02 03 Информационные системы и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное программирование»

Тема «Программное средство «Автоматизированная система управления IT-компанией. Подсистема управления сотрудниками»

Исполнитель

Студент 2 курса группы 1 Ермаков И. А.

(Ф.И.О.)

Руководитель работы ассистент Сухорукова И.Г. (учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий  
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В Пацей\_

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированное программирование"

|  |  |
| --- | --- |
| Специальность: 1-40 01 05 03 Информационные системы и технологии  Студент:\_\_\_\_Ермаков И.А. | Группа: \_\_1\_\_ |
| **Тема:** Программное средство «Автоматизированная система управления IT-компанией. Подсистема управления сотрудниками» | |

**1. Срок сдачи студентом законченной работы**: "май 2019 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС должно:

* Управлять базой данных: добавление, редактирование, удаление данных, связанных с пользователями (сотрудниками), проектами, задачами, ролями; вести учет прогресса выполнения проектов и задач; вести учет эффективности работы сотрудников;
* Управлять сотрудниками, работающими на проекте: назначать задачи сотрудникам с возможностью редактирования; распределять по проектам;
* Позволять сотрудникам полуавтоматизированно вносить информацию о выполнении задач и выполнять прочие операции, связанные с ними: открывать новые и закрывать выполненные задачи; отмечать прогресс их выполнения; разбивать на подзадачи;
* Автоматически вести учет процесса выполнения проекта, задачи и подзадачи (если таковая имеется);
* Вести учет выполнения сотрудниками поставленных задач и построение отчетов.

**2.2.** Программное средство должно быть выполнено с использованием ООП, взаимодействовать с базой данных, реализовано под ОС Windows. Отображение, бизнес-логика и хранилище данных должны быть максимально независимы друг от друга для возможности расширения. Управление программой должно быть интуитивно понятным и удобным. Использовать архитектурные шаблоны и шаблоны проектирования.

**3. Примерное содержание расчетно-пояснительной записки**

(перечень вопросов, подлежащих разработке)

* Введение;
* Обзор литературы и аналогов (алгоритмы решения, обзор литературы);
* Постановка задачи;
* Проектирование программного средства;
* Руководство пользователя;
* Тестирование программного средства;
* Заключение;
* Список используемых источников;
* Приложения.

**4. Форма представления выполненной курсовой работы:**

* Теоретическая часть курсового проекта должны быть представлены в формате MS Word. Оформление записки должно быть согласно правилам.
* Необходимые схемы, диаграммы и рисунки допускается делать в MS Office Visio, VS или других редакторах.
* Листинги программы представляются частично в приложении.
* К записке необходимо приложить CD (DVD).

**Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 21.02.2019 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 7.03.2019 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 21.03.2019 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 04.04.2019 |  |
| 5 | Кодирование программного средства | 18.04.2019 |  |
| 6 | Тестирование и отладка программного средства | 02.05.2019 |  |
| 7 | Оформление пояснительной записки | 07.05.2019 |  |
| 8 | Сдача проекта | 20.05.2019 |  |

**5. Дата выдачи задания** 21.02.2019

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Сухорукова И.Г.*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Ермаков И.А.*

(дата и подпись студента)

**Оглавление**

[**ВВЕДЕНИЕ** 5](#_Toc8220024)

[**1.** **Обзор аналогов** 5](#_Toc8220025)

[**2.** **Анализ требований к программному средству и разработка функциональных требований** 8](#_Toc8220026)

[**3. Проектирование программного средства** 9](#_Toc8220027)

[**3.1 Концепция** 9](#_Toc8220028)

[**3.2 Доменная модель и структура** 9](#_Toc8220029)

[**3.3 Модель данных** 9](#_Toc8220030)

[**4. Реализация программного средства** 11](#_Toc8220031)

[**5. Тестирование, проверка работоспособности и анализ полученных результатов** 14](#_Toc8220032)

[**6. Руководство по установке и использованию** 15](#_Toc8220033)

[**ЗАКЛЮЧЕНИЕ** 16](#_Toc8220034)

[**СПИСОК ЛИТЕРАТУРЫ** 17](#_Toc8220035)

[**Приложение 1** 18](#_Toc8220036)

[**Приложение 2** 19](#_Toc8220037)

**ВВЕДЕНИЕ**

Грамотно организованная система управления компанией – важная составляющая ее бизнес-стратегии. Сегодня стало очевидно: среди ресурсов, которыми располагает компания, одним из самых ценных является человеческий ресурс. Вместо отделов кадров в компаниях начинают появляться HR отделы (от английского human resources – человеческие ресурсы). И это в первую очередь связано с трансформацией рынка труда. Сегодня важно не только нанять подходящего сотрудника, но и правильно использовать его способности и навыки. В список обязанностей hr-специалиста добавились помощь в адаптации специалиста в коллективе, наблюдение за его поведением, взаимоотношениями в команде и подбор подходящего проекта. Компании стремятся не только нанять лучших, но и удержать их, помочь развиться. Для достижения этого целесообразно использовать автоматизированную систему управления сотрудниками.

Современные IT­-предприятия, в которых часто работает большое число сотрудников, стремятся иметь полный контроль над их деятельностью для обеспечения максимально эффективного и, соответственно, выгодного сотрудничества как с работниками, так и с клиентами. Однако, чем больше сотрудников работает в компании, тем сложнее управлять их деятельностью, что является серьезной проблемой, ведь один из самых ценных ресурсов для компании – время.

Одним из решений данной проблемы может стать программное средство, специализирующееся на распределении задач между работниками. Программное средство сможет быть эффективно использовано при управлении проектами, составлении отчётной документации об их выполнения и многом другом. Поиску и разработке такого решения и посвящен данный курсовой проект.

Цели проекта:

* Совершенствование умений написания программного кода;
* Формирование навыков проектирования баз данных;
* Формирование навыков проектирования и разработки пользовательского интерфейса;
* Развитие навыков тестирования продукта;
* Формирование умения построения архитектуры проекта с возможностью масштабирования.

Задача проекта:

Разработать архитектуру приложения, создать интерфейс, взаимодействие с которым будет понятно, как техническому специалисту, так и сотруднику отдела управления со средним уровнем компьютерной грамотности. Использовать в разработке объектно-ориентированный язык программирования.

# **Обзор аналогов**

Среди самых распространенных систем управления сотрудниками стоит выделить систему Jira и систему YouTrack.

Подробнее рассмотрим каждую из них ниже.

Jira - коммерческая система отслеживания ошибок. По заверению разработчика, с помощью системы можно эффективно организовать взаимодействие с пользователями. Тем не менее система получила широкое распространение в сфере управления проектами. Веб-интерфейс данной системы приведен ниже (Рис 2.1).

Название системы получено путём усечения слова «Gojira» — японского имени монстра Годзилла, что, в свою очередь, является отсылкой к названию конкурирующего продукта – Bugzilla. Она создавалась в качестве замены Bugzilla и во многом повторяет её архитектуру.

Система позволяет работать с несколькими проектами. Для каждого из проектов создаёт и ведёт схемы безопасности и схемы оповещения.

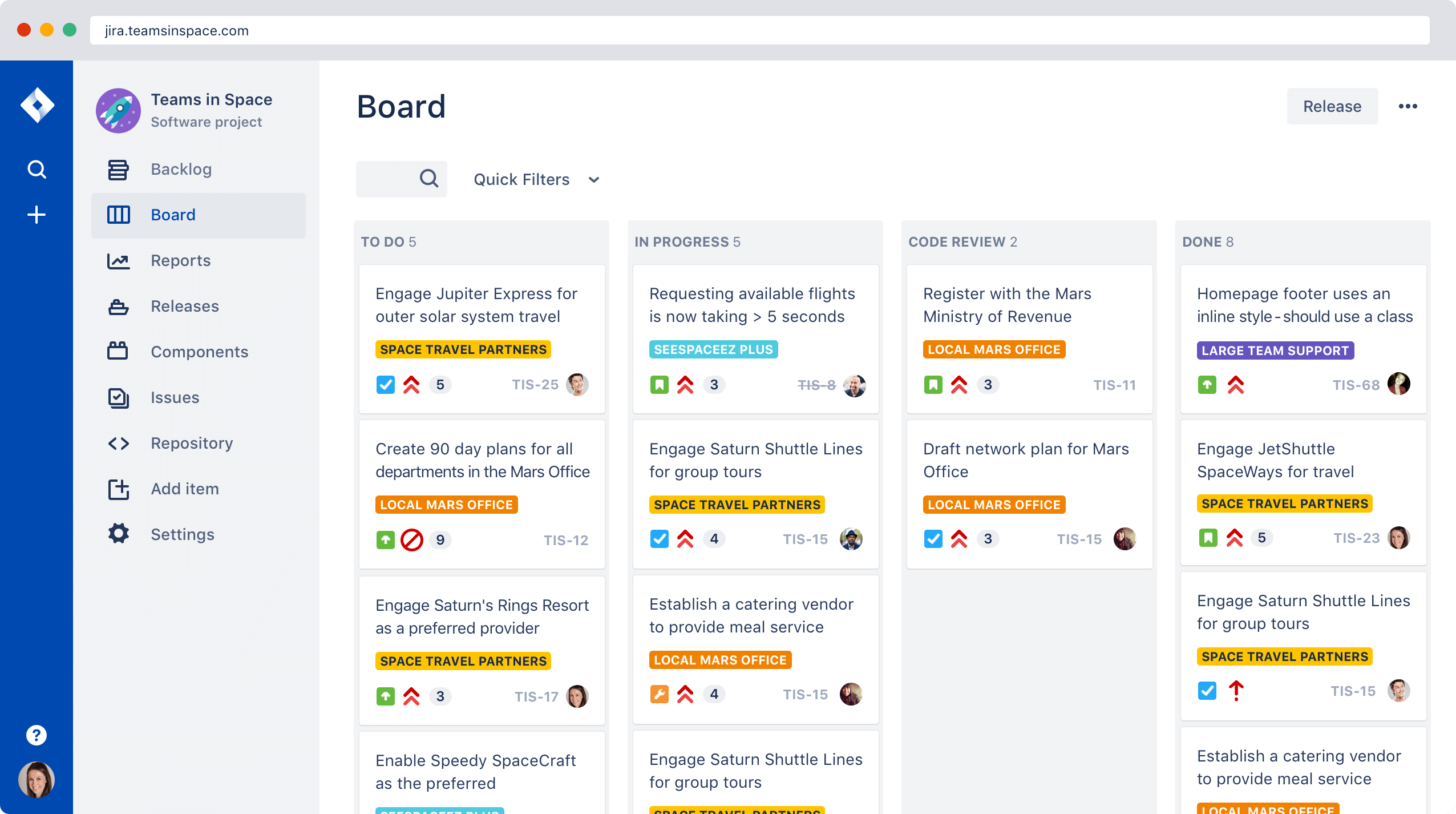


Рисунок 2.1 – интерфейс системы «Jira»

Следующая обозреваемая программа – YouTrack. Это коммерческая система отслеживания ошибок, являющаяся программным средством для управления проектами. Разработчик – JetBrains. YouTrack поддерживает поисковые запросы, автодополнение, манипуляцию с наборами задач, настройку набора атрибутов задачи, создание пользовательских рабочих процессов и реализует подход, основанный на преимущественном использовании клавиатуры.

Пользовательский интерфейс «YouTrack» (Рис. 2.2) основан на технологии AJAX и позволяет производить управление как с помощью клавиатуры, так и с помощью мыши. Поиск задач осуществляется с помощью поисковых запросов, выполняемых в единой строке поиска.

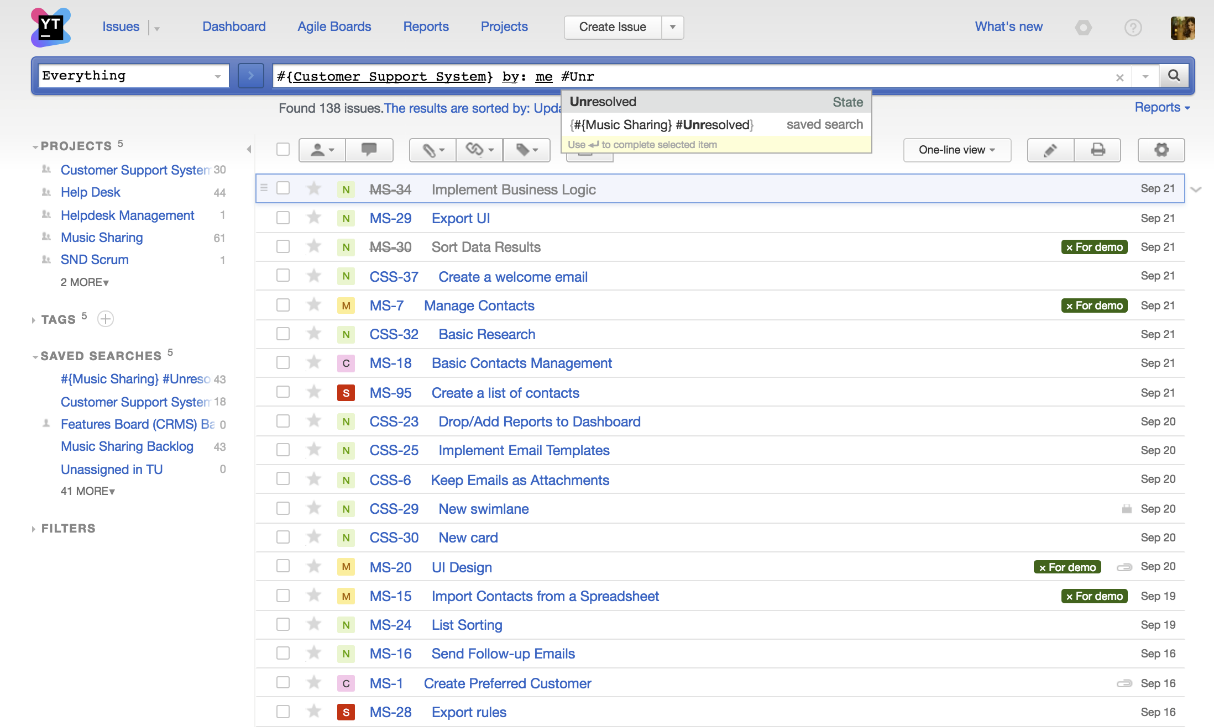


Рисунок 2.2 – интерфейс системы YouTrack

Рассмотрение аналогов показало их функционал и навело на следующие выводы:

* Интерфейс программного средства должен быть минималистичным, стоит избегать избыточности как в цвете, так и в информации. Это необходимо поскольку взаимодействие с программным средством в процессе рабочего дня будет происходить достаточно часто. Спокойный фон, логотип помогают снизить когнитивную нагрузку на пользователя, что важно для нас с точки зрения удержания клиента;
* Внедрение технологии AJAX позволяет производить управление как с помощью клавиатуры, так и с помощью мыши, что, по отзывам пользователей, удобно и повышает производительность, за которой стремятся компании, внедряющие в обиход программные средства по управлению персоналом;
* Пользователю важна функция работы с несколькими проектами. Для каждого из проектов стоит организовать создание и ведение схемы безопасности и схемы оповещения.

# **Анализ требований к программному средству и разработка функциональных требований**

**2.1 Функциональные требования**

Проведя анализ аналогов и потребностей пользователей, были выявлены следующие функциональные требования:

* Управлять базой данных:
  + Добавление, редактирование, удаление данных, связанных с пользователями (сотрудниками), проектами, задачами, ролями
  + Учет прогресса выполнения проектов и задач
  + Учет эффективности работы сотрудников
* Управлять сотрудниками, работающими на проекте:
  + Назначать и переназначать поставленные перед ними задачи
  + Распределять по проектам
* Позволять сотрудникам полуавтоматизированно вносить информацию о выполнении задач и выполнять прочие операции, связанные с ними:
  + Открывать новые и закрывать выполненные задачи
  + Отмечать прогресс их выполнения
  + Разбивать на подзадачи
* Автоматически вести учет процесса выполнения проекта, задачи и подзадачи (если таковая имеется)
* Вести учет выполнения сотрудниками поставленных задач

**3.3 Спецификация функциональных требований**

Бэкенд приложения реализован на языке программирования Java с использованием фреймворка Spring Boot для обработки REST-запросов, работы с базой данных и выполнения основной бизнес-логики приложения.

Для работы с базой данных была взята СУБД PostgreSQL и была программно связана с бэкендом с помощью ORM фреймворка Hibernate.

Веб-версия приложения была выполнена с использованием компонент-направленного фронтенд фреймворка Angular. Десктопное приложение Windows реализовано с помощью фреймворка Electron, портируя веб-версию под конкретную ОС.

# **3. Проектирование программного средства**

## **3.1 Концепция**

Так как программное средство специализируется на управлении сотрудниками, проектами и задачами, то мы можем выделить основные объекты доменной модели нашей системы, которые будут подробнее рассмотрены в подразделе «Модель и структура».

## **3.2 Доменная модель и структура**

Доменная модель и структура проекта представлена на диаграмме 3.1.

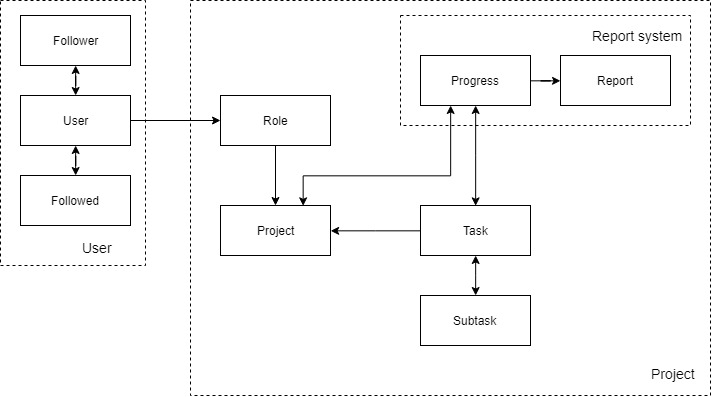
****

Диаграмма 3.1 – Доменная модель и структура проекта

**3.3 Модель данных**

Согласно разработанной доменной модели проекта была спроектирована и разработана модель данных, представленная на диаграмме 3.2.

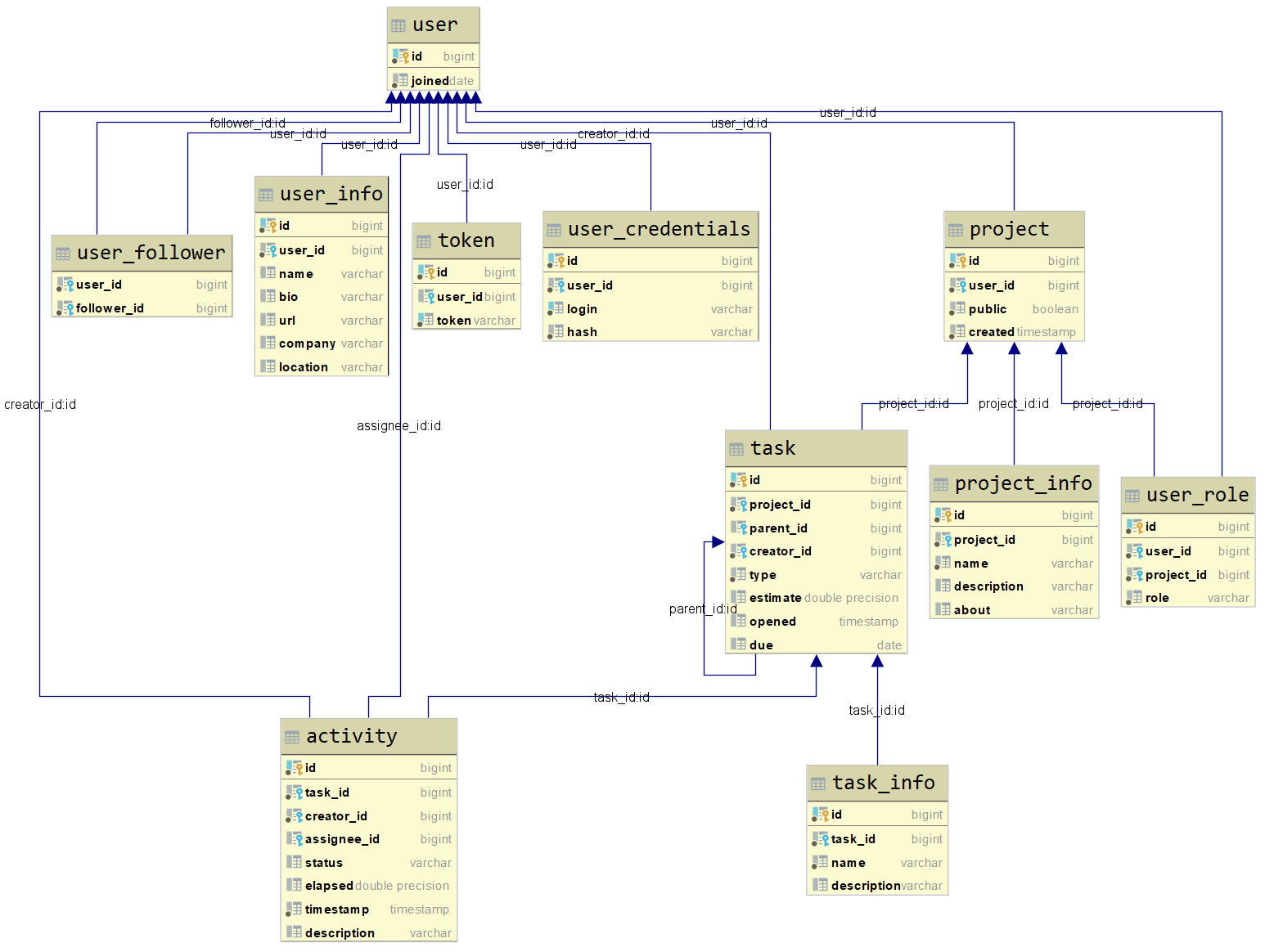
****

Диаграмма 3.2 – Модель данных проекта

# **4. Реализация программного средства**

**4.1 Разработка бэкенда**

Архитектура бэкенда завязана на паттерне «инверсия управления» (Inversion of Control - IoC), а конкретнее на паттерне «внедрения зависимости» (Dependency Injection - DI). Пример архитектуры использования паттерна «внедрения зависимости» представлен на диаграмме 4.1.

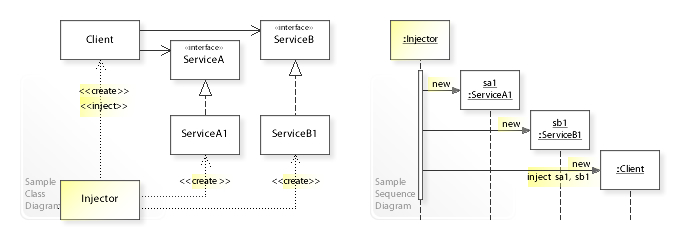


Диаграмма 4.1 – Архитектура паттерна DI

В разрабатываемом приложении данный паттерн использовался для связи друг с другом контроллеров, сервисов и репозиториев. Рассмотрим пример обращения клиента (фронтенд) к серверу (бэкенд):

1. Запрос клиента имеет определенный путь, который должен совпадать с конкретным и единственным сервлетом сервера. Сервлет – интерфейс Java, выполняющий роль сервера для единственного вида запросов. Для проверки совпадения и передачи управления сервер сверяет путь запроса с сервлетами из контейнера сервлетов
2. Контейнер сервлетов инвоцирует метод контроллера, сверяя тело запроса, параметры, заголовки и возвращаемый тип. Контроллер, произведя базовые преобразования типов передает управления сервису(-ам)
3. Сервис в свою очередь выполняет всю бизнес-логику приложения. В зависимости от поставленной задачи он может производить вычисления и/или передавать выполнения репозиторию(-ям)
4. Репозиторий является прослойкой между бизнес-логикой приложения и источником данных (сервером СУБД в нашем случае). Он преобразует программные задачи в SQL-запросы, выполняемые сервером СУБД
5. После выполнения всей иерархии компонентов (все классы, которые могут выступать в роли инжектора в паттерне внедрения зависимости называются компонентами) выполнение передается обратно вверх (отсюда и названия паттерна – «инверсия управления») и клиент либо получает ответ с требуемыми данными, или ошибку с кодом и сообщением

На диаграмме 4.2 представлена архитектура паттерна «Controller, Service, Repository», используемого в приложении и описанного выше.

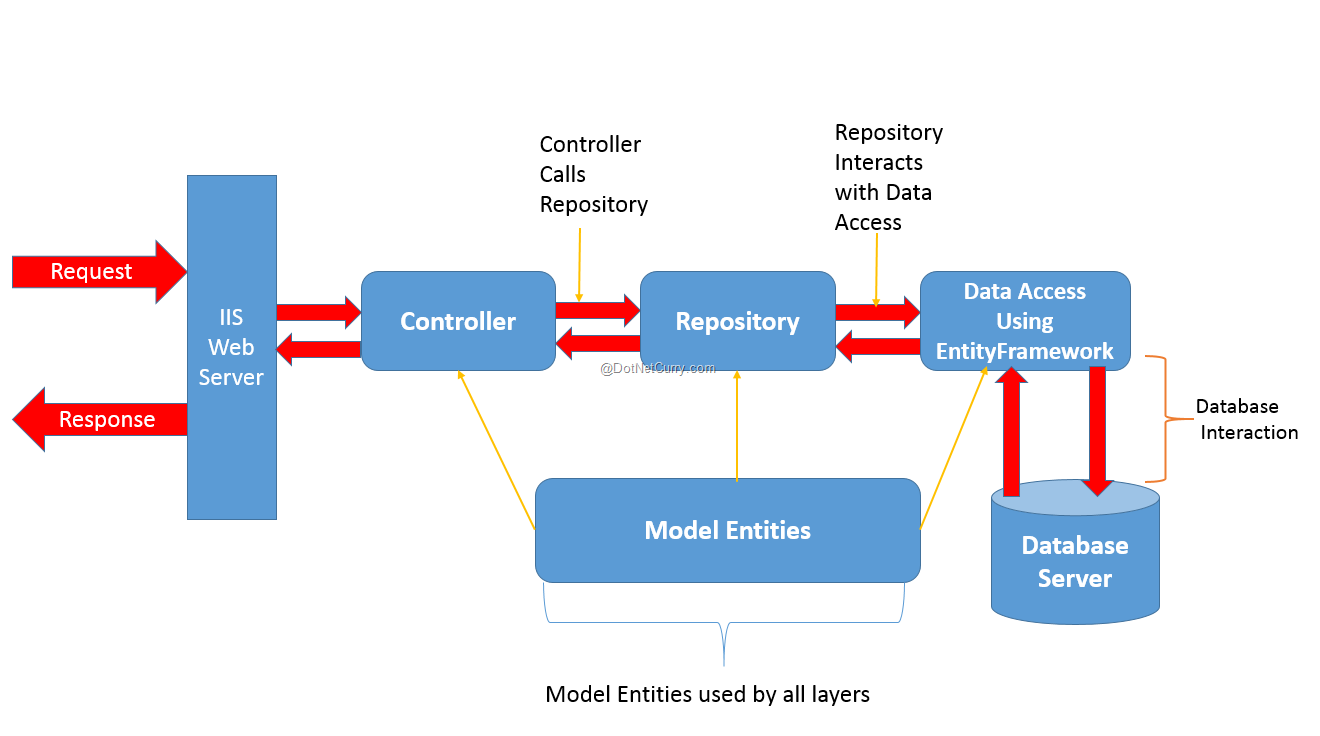


Диаграмма 4.2 – Архитектура паттерна «Controller, Service, Repository»

Пример реализации контроллера показан в Приложении 1.

**4.2 Разработка фронтенда**

Как уточнялось ранее, фреймворк Angular базируется на иерархии компонентов. Каждый компонент состоит из трех файлов:

* TypeScript код компонента
* CSS стили компонента
* HTML вид компонента

Пример компонента представлен в Приложении 2.

Особенностью фреймворка является не только модульность, но и взаимосвязь данных. Данные между компонентами передаются с помощью специальных декораторов, которые делают некоторый объект данных видимым для других компонентов. Фреймворк так же реализует паттерн «внедрение зависимости», что означает, что все связанные сервисы, необходимые для работы компонента, внедряются в этот компонент.

Сервис в фреймворке Angular представляет собой взаимодействующий элемент с контроллерами бэкенда, то есть он дублирует его функционал, но используя свои типы данных. Пример сервиса представлен в Приложении 3.

Проблема несоответствия типов в языках программирования Java и TypeScript решена с помощью встроенной библиотеки Spring Boot – «Jackson». Она автоматически сериализует и десериализует данные с обеих сторон в JSON (Нотация объектов JavaScript). DTO (объект передаваемых данных) обеих сторон представлен в Приложении 4.

# **5.** **Тестирование, проверка работоспособности и анализ полученных результатов**

**5.1 Тестирование бэкенда**

Фреймворк Spring Boot раскрывает широкие возможности для тестирования приложения. Было проведено для вида тестирования:

1. Блочное тестирование – тестирование некоторых методов компонентов, изолированно от остальной программы
2. Интеграционное тестирование – тестирование программы с точки зрения действий пользователя. Интеграционные тесты охватывают от нескольких компонентов до нескольких модулей и симулируют выполнения реальных задач
   1. **Тестирование фронтенда**

Фронтенд тестировался с применением ручного системного тестирования – функционал приложения проверялся вручную в браузере и оконном приложении.

# **6. Руководство по установке и использованию**

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения работы были достигнуты следующие результаты:

* Изучены основы объектно-ориентированного программирования и применены в практической разработке программного средства;
* Исследована сфера управления проектами с акцентом на управление сотрудниками. В работе было отмечено, что сотрудники компании и время, которое они тратят на выполнение той или иной задачи являются ключевыми факторами, влияющими на эффективность и, следовательно, на успешность компании;
* Сформулирована теория, что внедрение технологии AJAX (позволяет производить управление как с помощью клавиатуры, так и с помощью мыши) позволяет повысить продуктивность работы с программным средством. В ходе тестирования продукта респондентами было сделано аналогичное замечание, поэтому теорию можно считать подтвержденной;
* Показано;
* Разработана;
* Предложена;
* Подготовлены:
* Испытанно.

# **СПИСОК ЛИТЕРАТУРЫ**

[1] Atlassian Jira [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.atlassian.com/software/jira>.

[2] Wikipedia [Свободная энциклопедия]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Jira>.

[3] Wikipedia [Свободная энциклопедия]. – Электронные данные. – Режим доступа: <https://en.wikipedia.org/wiki/YouTrack>.

[4] JetBrains YouTrack [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.jetbrains.com/youtrack/>.

[5] draw.io [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.draw.io/>.

[6] yFiles builder [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.yworks.com/products/yfiles>;

**Приложение 1**

@RestController

//общий путь запросов для всех методов этого контроллера

@RequestMapping("auth")

public class AuthController {

// компонент-инжектор, в данном случае - сервис

private final UserService userService;

@Autowired

public AuthController(UserService userService) {

this.userService = userService;

}

// POST HTTP-запрос

@PostMapping

public String authenticate(@RequestBody AuthUserDto authUserDto /\*тело запроса\*/) {

// передача управления сервису-инжектору

return userService.authenticate(authUserDto);

}

}

**Приложение 2**

//логика компонента

// декоратор компонента

@Component({

selector: 'app-feed-project',

templateUrl: './feed-project.component.html',

styleUrls: ['./feed-project.component.scss']

})

// класс компонента

export class FeedProjectComponent implements OnInit {

// данные, приходящие извне

@Input()

project: Project;

@Input()

me: User;

// конструктор с зависимостями

constructor(

private router: Router,

private tokenProviderService: TokenProviderService,

private authService: AuthService

) {

}

}

//CSS стили компонента

.project {

display: flex;

align-items: center;

padding: 1px;

.is-open-status {

i {

font-size: 20px;

padding: 4px;

}

.public-wrapper {

color: var(--public-lock);

}

.private-wrapper {

color: var(--private-lock);

}

}

.text {

margin: 0 1px;

}

}

//HTML вид компонента

<div class="project" \*ngIf="project && me" (click)="open()">

<div class="is-open-status">

<i class="fas fa-unlock-alt public-wrapper" \*ngIf="project.isPublic === true"></i>

<i class="fas fa-lock private-wrapper" \*ngIf="project.isPublic === false"></i>

</div>

<div class="text">{{me.login}}</div>

<div class="text">/</div>

<div class="text">{{project.name}}</div>

</div>

**Приложение 3**

@Injectable({

providedIn: 'root'

})

export class AuthService {

constructor(

private http: HttpClient

) {

}

// метод, дублирующий соответствующий метод на бэкенде

authenticate(authUser: AuthUser): Observable<string> {

return this.http.post<string>(API\_URL + 'auth', authUser, {responseType: 'text' as 'json'});

}

}

**Приложение 4**

//тип данных на бэкенде

public class UserDto {

public Long id;

@JsonFormat(pattern = "yyyy-MM-dd")

public LocalDate joined;

public UserInfoDto userInfo;

public String login;

public UserDto() {

}

}

//соответствующий тип данных на фронтенде

export class User {

id: number;

login: string;

joined: Date;

userInfo: UserInfo;

}