ВВЕДЕНИЕ

Целью дипломной работы является разработка программного средства отслеживания задач с помощью метода управления разработкой kanban. Метод управления разработкой kanban представляет собой инструмент в виде доски с задачами, который позволяет наладить процесс работы над одним конкретным проектом. В самом простом случае kanban доска представляет собой страницу с четырьмя колонками. В каждой колонке находятся задачи с определенной степенью готовности: новая задача, задача на стадии разработки, задача в стадии тестирования и выполненная задача. В случае, если задача была выполнена некачественно, она возвращается к одному из предыдущих статусов.

В последнее время, Kanban набирает большую популярность в производстве программного обеспечения. Многие компании-разработчики программного обеспечения считают данный подход исключительно полезным. И хотя Kanban не достаточно эффективен для продуктовых команд, он очень эффективен для команд следующего вида:

* ­ группы сопровождения программного обеспечения, где важен не план, а скорость реагирования при поступлении новых задач;
* ­ группы тестирования, работающие отдельно от групп разработки;
* службы поддержки.

Также стоит отметить, что методология Kanban эффективна для различного рода стартапов, когда у команды нет четкого плана, но ведется активная работа над программным продуктом.

Однако настоящий программный продукт не является направленным исключительно на применение в IT-сфере. Данный проект может быть полезен в том числе для организации труда в сферах торговли, строительства или же просто может быть использован людьми в качестве планировщика.

В силу популярности методологий разработки с использованием kanban у настоящего программного продукта существует ряд аналогов. Наиболее популярными среди них являются Trello и Kanban flow.

На рисунке 1 представлен внешний вид рабочего стола проекта в Trello. Отсюда можно увидеть следующие особенности:

* статусы, которые может иметь задача, создаются пользователем;
* для задач можно указывать теги;
* для задач можно устанавливать крайний срок выполнения.

На рисунке 2 представлен внешний вид рабочего стола проекта в Kanban Flow. Данная доска имеет следующие особенности:

– Пользователю предлагается четыре статуса задач;

– каждую задачу можно разбить на несколько подзадач;

– имеется возможность добавления собственных статусов задач.

Все представленные аналоги являются довольно гибкими в плане настройки и довольно простыми в использовании, в том числе и для тех пользователей, которые никогда не работали с kanban досками.



Рисунок 1 – ­ Внешний вид доски Trello



Рисунок 2 ­– Внешний вид доски Kanban flow

Задачей данного дипломного проекта является разработка программного средства отслеживания задач. Данный программный продукт будет представлен в виде веб-приложения, состоящего из сайта и веб-сервиса. Планируется реализовать следующий функционал:

* организация ведения проектов с использованием доски Kanban;
* построение отчетов о затраченном времени;
* поиск задач;
* создание проектов, ролей, управление списком пользователей, правами.

**1** ОБЗОР ЛИТЕРАТУРЫ

**1.1** Обзор используемых технологий

В данном разделе выберем технологии, которые будут использованы для реализации настоящего проекта.

**1.1.1** Выбор технологий клиентской части приложения

Веб–сайт может быть представлен как одностраничное веб-приложение (в этом случае рендеринг html-страницы производится на стороне клиента) или как многостраничное приложение (рендеринг html-страницы производится на стороне веб-сервиса).

На сегодняшний день в области разработки веб-приложений широко распространена разработка одностраничных приложений (SPA – Single Page Application). Веб-сайт представлен единственной веб-страницей-оболочкой, организующей взаимодействие с пользователем посредством динамической подгрузки. При последующих запросах сервер и клиент осуществляют обмен данными (например, в формате JSON), а непосредственно рендеринг страницы производится на стороне клиента. У SPA веб-сайтов есть два неоспоримых преимущества: экономия сетевого трафика и уменьшение нагрузки на серверную часть приложения. Рассмотрим технологии разработки SPA веб-приложений.

В настоящее время наиболее пополярными фреймворками разработки SPA веб-приложени являются Angular и ReactJS.

Angular является мощным структурным фреймворком для динамических веб-приложений. В фреймворке используется шаблон проектирования MVW (Model-View-Whatever) (Модель-Отображение-Что угодно) и расширение традиционного HTML синтаксиса директивами. Другая отличительная черта фреймворка — автоматическая синхронизации модели и представления. Кроме того, Angular поддерживает внедрение зависимостей и предлагает множество других полезных функций, среди которых можно выделить маршрутизацию, асинхронные запросы, валидацию форм, шаблоны и анимацию.  
Angular имеет достаточно низкий порог вхождения благодаря обширной документации и многочисленным учебным материалам, как из официальных источников, так и созданным крупным сообществом разработчиков. Разработчик Angular – компания Google, предоставляет официальное руководство по стилю написания кода на данном фреймворке. Разработка на данном фреймфорке может осуществляться на двух языках программирования: JavaScript и TypeScript.

ReactJS — это JavaScript фреймворк разработанный компанией Facebook, который используется для построения пользовательского интерфейса. Одной из его отличительных особенностей является возможность использовать JSX язык программирования с близким к HTML синтаксисом, который транслируется в JavaScript. Разработчики могут добиваться высокой производительности приложений с помощью Virtual DOM (Объектная модель документа). C ReactJS возможно создавать изоморфные приложения, в которых данные могут подгружаться с сервера в асинхронном режиме без полной перезагрузки страницы. Созданные компоненты могут быть с легкостью изменены и повторно использованы в новых проектах. Высокий процент переиспользования кода повышает покрываемость тестами, что, в свою очередь, приводит к более высокому уровню контроля качества.

Из-за низкого порога вхождения и наличия обширной документации выбор был сделан в пользу фреймворка Angular. В связи с ограниченным сроком, на реализацию настоящего проекта, эти факторы приобрели ключевое значение.

**1.1.2** Выбор технологий для разработки веб-сервиса

Поскольку для языка Java существует множество технологий, упрощающих разработку серверной части веб-приложений, рассмотрим некоторые из них. Наиболее распространенными фреймворками для разработки веб-приложений являются: Spring Framework, JSF, Grails.

Spring Framework — это универсальный Java фреймворк с открытым исходным кодом. Spring состоит из множества модулей, которые разработчик может подключать к своему приложению по необходимости. Подавляющее большинство этих модулей может работать независимо друг от друга. По умолчанию Spring предоставляет две основных возможности: инверсия управления (каждый объект получает свои зависимости, а не создает их самостоятельно) и внедрение зависимостей. Spring имеет низкий порог вхождения благодаря обширной документации и большому количеству различных статей в сети интернет.

Фреймворк JSF (Java Server Faces) был разработан компанией Oracle в рамках спецификации J2EE. Данный фреймворк не подходит для быстрой разработки веб-приложений, однако он легок в использовании благодаря наличию качественной документации и отсутствию внешних зависимостей, при использовании данного фреймворка не происходит выход за рамки спецификации J2EE. JSF славится своей богатой инфраструктурой и различными модулями, которые подходят на для решения широкого круга задач: разработки пользовательского интерфейса, взаимодействия с базами данных и др. Главным недостатком данного фреймворка является сложность его внутреннего устройства.

Данный фреймворк, в отличие от предыдущих, написан не на языке Java, а на языке Groovy, однако он является полностью совместимым с языком программирования Java. Вообще по своей сути Grails можно считать не фреймворком, а своеобразной платформой для интеграрации иных фреймворков. Это позволяет Grails взаимодействовать со многими фреймворками, такими как Spring, Hibernate, Struts 2 и др. Вместе с вышеупомянутыми фреймворками Grails содержит в себе различные плагины для интеграции с контейнерами сервлетов, базами данных и средами разработки.

Среди рассмотренных фреймворков в качестве инструмента для разработки настоящего проекта будет использован Spring Framework, поскольку на сегодняшний день этот фреймворк является самым популярным среди Java разработчиков. Также Spring содержит набор стандартных конфигураций Spring Boot, воспользовавшись которыми отпадает необходимость настройки проекта вручную.

**1.1.3** Выбор СУБД

Выберем стредство управления базой данных. По модели данных базы данных бывают: иерархические, объектные, объектно-реляционными, реляционными, сетевыми, функциональными. В настоящее время наиболее популярными моделями баз данных являются иерархические и реляционные.

Однако, в настоящей предметной области данные хорошо структурированы (задача, проект, пользователь и др), что хорошо проецируется на реляционные базы данных. В связи с этим фактом, а также в связи с имеющимся опытом работы с реляционными СУБД, выбор пал в их сторону.

SQLite по своей сути является файловой базой данных: все таблицы хранятся в виде файла на локальном компьютере. Когда приложение использует SQLite, их связь производится с помощью функциональных и прямых вызовов файлов, содержащих данные (например, баз данных SQLite), а не интерфейса, что повышает скорость и производительность операций.

Типы данных, которые возможно хранить в базе данных SQLite: NULL, INTEGER, REAL (числовое значение с плавающей точкой, хранится в формате 8-байтного числа IEEE с плавающей точкой), TEXT (значение строки текста, хранится с использованием кодировки базы данных (UTF-8, UTF-16BE или UTF-16LE)), BLOB (значение бинарных данных, хранящихся точно в том же виде, в каком были введены)

### Преимущества SQLite:

* файловая структура - вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины;
* используемые стандарты - хотя может показаться, что эта СУБД примитивная, но она использует SQL. Некоторые возможности SQL в ней недоступны (RIGHT OUTER JOIN или FOR EACH STATEMENT), но основные все-таки поддерживаются;
* в процессе разработки приложений часто появляется необходимость масштабирования. SQLite предлагает всё что необходимо для этих целей, так как состоит всего из одного файла и библиотеки написанной на языке C.

Недостатки SQLite:

– отсутствие системы пользователей;

### отсутствие возможности увеличения производительности;

### вероятность повреждения данных при асинхронном доступе к ним.

### MySQL в настоящее время является одной из самых популярных СУБД, используемых при разработке высоконагруженных приложений. Для нее существует огромное количество документации и различных статей, помогающих начать работу с данной СУБД.

### MySQL поддерживает большое количество типов данных: TINYINT (очень малые целочисленные значения), SMALLINT (малые целочисленные значения), MEDIUMINT (средние целочисленные значения), INTEGER, BIGINT (большие целочисленные значения), DOUBLE, DECIMAL (распакованное значение с плавающей точкой, всегда знаковое), DATE (дата), DATETIME (дата и время в одном значении), TIMESTAMP (временная отметка timestamp), TIME, YEAR (год, 2 или 4 числа), CHAR, VARCHAR (строковое значение переменной длины), FLOAT, SET (множество), ENUM (перечисление), BLOB (значение типа BLOB 65535 (2^16 - 1) символов - максимальная длина), TINYBLOB, MEDIUMBLOB, LONGBLOB.

### Преимущества MySQL:

* простота использования;
* поддержка большинства функций языка SQL;
* реализовано большое количество средств, обеспечивающих безопасность;
* высокая производительность.

### Недостатки MySQL:

### не все возможности SQL реализованы;

### невысокая надежность, связанная со способами обработки данных;

### медленная разработка: новые версии СУБД выходят с большой задержкой.

От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного. Например, полной поддержкой транзакций. Акцент разработчиков был уделен производительности СУБД. Параллельность достигается не за счет блокировки операций чтения, а благодаря реализации управления многовариантным параллелизмом. PostgreSQL легко дополняется хранимыми процедурами.

Типы данных поддерживаемые PostgreSQL: bigint (знаковое 8-ми битное целочисленное значение), bigserial (автоматически инкрементируемое 8-ми битное целочисленное значение), bit (строка постоянной длины), bit varying (строка переменной длины), boolean, box (прямоугольник на плоскости), bytea (массив байт), character varying (строковое значение переменной длины), character (строковое значение постоянной длины), cidr (IPv4/IPv6 сетевой адрес), circle (круг на плоскости), date (календарная дата), double, inet (IPv4/IPv6 адрес хоста), integer, interval (отрезок времени), macaddr (MAC адрес), mony (валютное значение), numeric (точное численное значение с выбранной точностью), path (геометрическая кривая на плоскости), point (геометрическая точка на плоскости), polygony (прямоугольик на плоскости), real (число с плавающей точкой одинарной точности), smallint (знаковое целочисленное значение), serial (автоматические инкрементируемое целочисленное значение), text (строковое значение переменной длины), time (время суток (без часового пояса)), time with time zone (время суток (включая часовой пояс)), timestamp (дата и время (без часового пояса)), timestamp with time zone (дата и время (с часовым поясом)), tsquery (текстовый поисковый запрос), tsvector (документ текстового поиска), txid\_snapshot (пользовательский снимок транзакции с ID), uuid (универсальный уникальный идентификатор).

Достоинства PostgreSQL:

* бесплатная СУБД с качественной реализацией SQL;
* многочисленное сообщество разработчиков позволяет оперативно найти решение некоторых проблем;
* наличие большого количества дополнений, которые позволяют разрабатывать данные для этой СУБД и управлять ими;
* реализация объектно-ориентированной модели.

### Недостатки PostgreSQL:

* слабая производительность (особенно при чтении);
* низкая популярность;
* мало хостингов поддерживает данную СУБД.

В настоящем проекте будет использоваться база данных MySQL. Именно эта СУБД будет использована, потому что она является одной из самых популярных среди всех ныне существующих реляционных баз данных, а так же в связи с имеющимся опытом разработки на данной СУБД. Также выбор в сторону MySQL пал, потому что данная база является бесплатной, кроссплатформенной и хорошо подходит для хранения структурированных данных.

**1.2** Обзор аналогов

Как было упомянуто ранее, поскольку методологии с использованием Kanban досок широко распространены в настоящее время, уже существует несколько популярных продуктов в этой области. Рассмотрим некоторые из них более детально.

**1.2.1** Trello

Trello представляет собой доску с задачами, которые распределены по колонкам (см. рис. 1). В каждой колонке содержится задача принадлежащая той или иной группе.

Каждая задача представлена в виде карточки. С карточками можно проводить большое количество манипуляций (см. рис. 1.1): комментирование, прикрепление файлов, назначение меток и др.



Рисунок 1.1 – Экран управления задачей.

В Trello для каждого проекта создается своя рабочая группа (см. рис. 1.2). Члены данной группы имеют возможность просматривать в режиме реального времени состояние задач, касающихся проекта. Также возможно получать уведомления по электронной почте.



Рисунок 1.2 –­ Экран рабочей группы проекта

В общем случае trello представляет собой web-приложение, однако имеется и приложение для мобильных устройств под управлением операционной системы IOS.

По мнению некоторых разработчиков Trello не годится для разработки крупномасштабных проектов, а больше подходит для различого рода стартапов или в качестве ежедневника. Использование всех версий Trello в настоящий момент полностью бесплатно.

**1.2.2** Kanban flow

Данный сервис является абсолютно бесплатным и позволяет создать несколько досок, при необходимости работы сразу над несколькими проектами.

Изначально пользователю предоставляется доска с четырьмя колонками (см. рис. 2), в каждой колонке задачи с определенным статусом:

— новые задачи,

— задачи на сегодня,

— задачи, которые выполняются,

— выполненные задачи.

При добавлении новой задачи в проект можно указывать: время, которое потребуется для решения задачи, комментарий к задаче и исполнителя задачи (см. рис. 1.3). При работе над задачей можно указать фактическое время ее выполнения.



Рисунок 1.3 — Создание новой задачи в kanban flow.

Смена статуса задачи осуществляется методом перетаскивания, а при нажатии на задачу правой кнопкой мыши, появляется контекстное меню со списком некоторых действий над задачей.

Данная доска является полностью настраиваемой, при необходимости пользователь может изменить порядок колонок, изменить название колонок, добавить или удалить колонки (см. рис. 1.4).

Также существует возможность добавления участников в проект и коллективной работы над ним. Приглашенному пользователю придет уведомление на электронную почту.



Рисунок 1.4 –­ Экран редактирования колонок Kanban flow.

В итоге можно сказать, что доска Kanban flow довольно функциональна и удобна, одним из недостатков, для некоторых пользователей, является отсутствие мобильного приложения Kanban flow.

**1.3** Выводы

Резюмируя всё вышесказанное можно сказать, что настоящий дипломный проект будет состоять из серверной и клиентской частей. Серверная часть будет написана на языке программирования Java, с использованием фреймворка Spring. Клиентская часть будет написана с использованием фреймворка Angular на языке JavaScript. В качестве СУБД была выбрана MySQL.

**2.** СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Перед разработкой структуры приложений необходимо более глубоко изучить требования к разрабатываемой системе.

На чертеже 666 изображена диаграмма использования разрабатываемого программного продукта. В системе можно выделить четыре актера: пользователь, руководитель проекта, администратор системы и непосредственно программное средство.

Все процессы, происходящие в системе сводятся к запросу к программному средству. Пользователь может управлять задачей и заполнять отчеты о затраченном времени. Руководитель кроме вышеупомянутых действий может так же: создавать задачи, назначать проектую группу, назначать права для группы, отслеживать статистику по задачам. Администратор системы обладает всеми правами.

Исходя из функций пользователей системы, была проведена декомпозиция программного средства на блоки.

**2.1** Структура приложения

После изучения теоретических аспектов разрабатываемой системы и постановки требований к ней, систему необходимо разбить на функциональные блоки. Таким образом достигается гибкость архитектуры, что позволяет изменять сами модули без изменения всей системы в целом.

В настоящей системе на стороне веб-сервиса можно выделить следующие функциональные блоки:

* модуль API веб-сервиса;
* модуль аутентификации;
* модуль отправки уведомлений;
* модуль построения отчетов;
* модуль маршрутизации фреймворка Spring;
* модуль управления задачами;
* модуль администрирования и авторизации;
* модуль доступа к данным;
* база данных веб-сервиса.

На стороне веб-сайта можно выделить следующие функциональные блоки:

* модуль рендеринга html-страниц;
* модуль взаимодействия с веб-сервисом;
* фреймворк Angular;

Структурная схема, где представлены все вышеперечисленные блоки приведена на чертеже ГУИР.400201.136 С1.

В системе каждый модуль выполняет свою задачу и взаимодействует с другими модулями посредством интерфейсов.

Рассмотрим функциональные модули разрабатываемой системы.

*Модуль API веб-сервиса.* Этот блок является своего рода «ядром» настоящего программного продукта. Данный модуль принимает запросы от модуля маршрутизации фреймворка Spring, затем обращается, по необходимости, к другим модулям веб-сервиса, производит предварительную обработку ответа и посылает ответ модулю маршрутизации.

*Модуль аутентификации.* Данный модуль отвечает за авторизацию пользователей. Авторизация производится с помощью логина и пароля, причем пароль хранится на стороне веб-сервиса в зашифрованном виде, используется метод хеширования bcrypt. Данный метод хеширования является «односторонним», это значит что не существует способа по ключу получить исходный пароль. Во время авторизации из базы данных извлекается хеш пароля, который впоследствии сравнивается с хешем пароля, который ввел пользователь. При успешной авторизации пользователь перенаправляется на главную страницу приложения, иначе происходит перенаправление на страницу авторизации. Блок аутентификации реализуется с помощью модуля Spring Security фреймворка Spring.

*Модуль отправки уведомлений.* Данный модуль предназначен для отправки сообщений по электронной почте пользователям. Модуль может использоваться для отправки уведомлений при смене пароля, восстановлении пароля, изменениях в проекте и др. Данный модуль планируется разработать с помощью библиотеки JavaMail и ее обертки spring-boot-starter-mail.

*Модуль построения отчетов.* Модуль построения отчетов необходим для получения статистики по задачам, имеющимся на проекте и для отправки отчетов пользователем о проделанной работе.

*Модуль маршрутизации фреймворка Spring.* Данный модуль является своего рода «точкой входа» в веб-сервис для веб-сайта. Модуль представляет собой контроллер, который принимает запросы по протоколу http в формате json и делегирует данный запрос модулю API веб-сервиса. Получив ответ от модуля API веб-сервиса, модуль маршрутизации отправляет клиенту ответ.

*Модуль управления задачами.* Данный модуль предназначен для создания задач, смены статуса задачи поиска задач и т.д. Модуль будет представлен в виде набора классов, позволяющих управлять задачей.

*Модуль администрирования и авторизации*. В данном модуле сосредоточены функции получения прав пользователя, назначения прав пользователям а так же для администрирования системы в целом (создание проекта, назначение руководителя проекта). Авторизация производится с использованием модуля Spring Security фреймворка Spring. После создания проекта его руководитель создает проектную группу. Для каждой группы указываются её права: возможность создавать задачу, право на смену статуса задач, может ли пользователь выбирать задачу самостоятельно, может ли пользователь назначать задачи другим пользователям.

*Модуль доступа к данным.* Данный модуль является своего рода адаптером между базой данных и непосредственно веб-сервисом. Модуль будут реализовывать шаблон проектирования DAO (Объект для доступа к данным)(Data Access Object). Используя такой подход можно с лёгкостью поменять схему базы данных или тип хранилища данных (например с SQL на NoSQL), при этом не затронув логику самого веб-сервиса. Данный модуль реализуется с помощью модуля Spring Jdbc фреймворка Spring.

*База данных веб-сервиса.* В настоящем программном продукте хранилище данных представлено в виде базы данных SQL. В качестве базы данных была выбрана база MySQL с движком MariaDB.

*Модуль рендеринга html-страниц.* Поскольку клиентская часть приложения основана на концепции SPA, рендеринг html-страниц происходит непосредственно на стороне клиента (в браузере). В таком случае клиент получает ответ от сервера в формате json и на его основе отрисовывает html-страницу. Такой подход позволяет сократить объем трафика между веб-сайтом и веб-сервисом.

*Модуль взаимодействия с веб-сервисом.* Данный модуль работает на стороне веб-сайта и производит передачу запроса к веб-сервису по протоколу http. После обработки запроса веб-сервисом данный модуль получает ответ по тому же протоколу. Запросы отправлятся на веб-сервер в асинхронном режиме с использованием ajax.

*Фреймворк Angular.* Данный модуль по сути явдяется фреймворком, упрощающим разработку клиентской части веб-приложения. С помощью этого фреймворка осуществляется рендеринг html-страниц и отправка http-запросов веб-сервису.

В данной подглаве были рассмотрены функциональные блоки разрабатываемой системы, установлены их предназначения и связи между ними.

**2.2** Структура базы данных

Следующим этапом разработки программного продукта является постороение базы данных. На чертеже 667 изображена структура базы данных программного средства отслеживания задач.

Далее будут рассмотрены таблицы базы данных разрабатываемой системы.

*User.* Данная таблица предназначена для хранения информации о пользователях системы. Первичным ключом здесь является поле *us\_username* (логин пользователя). Внешний ключ *us\_company\_id* ссылается ссылается на таблицу company и указывает в какой компании работает пользователь. Таблица *user* состоит в отношении многие ко многим с таблицей *role* и с таблицей *project*.

*Project.* В этой таблице хранится информация о проектах внутри системы. В качестве первичного ключа здесь выступает поле *pr\_id* (синтетически сгенерированное число). Внешний ключ *pr\_lead* ссылается на таблицу *user* и указывает руководителя проекта. Внешний ключ *pr\_company\_id* ссылатся на таблицу company и указывает в рамках какой компании разрабатывается проект.

Таблица *task* используется для храненения проектных задач. В качестве первичного ключа используется поле *ta\_id* (искусственное число). Внешний ключ *ta\_status* ссылается на таблицу *task\_status*, тем самым указывая текущий статус задачи. Внешний ключ *ta\_creator* ссылается на таблицу user и указывает создателя задачи. Внешний ключ ta\_executor указывает текущего исполнителя задачи. Внешний ключ *ta\_project* ссылается на таблицу *project* и указывает проект, которому принадлежит задача. Таблица *task* состоит в отношении многие ко многим с таблицей *tag*.

Таблица *tag* представляет собой набор тегов для задач в рамках проекта. По тегам осуществляется поиск задач. В качестве первичного ключа выступает поле *ta\_id*. Таблица состоит в отношении многие ко многим с таблицей *task*.

Таблица *task\_status* используется для хранения статуса задачи в рамках проекта. Поле *ts\_id* используется в качестве первичного ключа. Внешний ключ *ts\_project* ссылается на таблицу *project* и указывет проект в рамках которого задача может иметь данный статус.

*Comment.* Данная таблица используется для хранения комментариев к задаче. Поле *co\_id* здесь является первичным ключом. Внешний ключ *co\_task\_id* ссылается на таблицу *task*, тем самым указывая задачу, которой принадлежит комментарий. Внешний ключ *co\_creator* ссылается на таблицу *user*, тем самым указывая автора комментария.

Таблица *attachment* предназначена для хранения прикрепленных к задаче файлов. Поле *at\_id* является первичным ключом. Внешний ключ *at\_task\_id* ссылается на таблицу *task*, тем самым указывая задачу к которой прикреплен файл. Внешний ключ *at\_owner* ссылается на таблицу user и указывает пользователя, прикрепившего файл.

Таблица *company* используется для хранения компаний внутри системы. В качестве первичного ключа используется поле *co\_id*. Внешний ключ *co\_owner* ссылается на таблицу *user*, тем самым указывая владельца компании.

Таблица *role* используется для хранения ролей пользователей в рамках проекта. Поле *ro\_id* является первичным ключом. Внешний ключ *ro\_project\_id* ссылается на проект, в рамках которого используется та или иная роль. Таблица *role* состоит в отношении многие ко многим с таблицей *permission* и таблицей *user*.

Таблица *permission* используется для хранения прав пользователя в рамках роли. Поле *pe\_id* является первичным ключом. Внешний ключ *pe\_project\_id* ссылается на проект, в рамках которого используется данный вид права. Таблица permission состоит в отношении многие ко многим с таблицей *role*.

Таблица *change\_status\_permission* предназначена для хранения прав пользователей для смены статуса задачи. Поле *csp\_perm\_id* (идентификатор записи из таблицы *permission*) используется в качестве первичного ключа. Внешний ключ *csp\_status\_from* ссылается на таблицу *task\_status*, тем самым указывая статус задачи в каком статусе может быть изменен. Внешний ключ *csp\_status\_to* ссылается на таблицу *task\_status*, тем самым указывая на какой статус может быть изменен статус задачи.

Таблица *logged\_work* используется для хранения отчетов о затраченном времени по конкретной задаче. Поле *lw\_id* является первичным ключом. Внешний ключ *lw\_username* ссылается на таблицу user, тем самым указывая пользователя составившего отчет.

Таблица *task\_history* используется для хранения истории изменения статуса задачи. Поле *th\_id* является первичным ключом. Внешний ключ *th\_task\_id* ссылается на таблицу task, тем самым указывая задачу. Внешний ключ *th\_username* ссылается на таблицу *user* и указывает пользователя, который внес изменения в статус задачи. Внешний ключ *th\_task\_status* указывает новый статус задачи.

В данной главе была спроектирована схема базы данных разрабатываемой системы. Были рассмотрены ключевые таблицы в схеме и установлены связи между ними.

ЗАКЛЮЧЕНИЕ

В ходе работы над дипломным проектом было реализовано веб-приложение состоящее из веб-сайта и веб-сервиса. Был реализован следующий функционал:

* организация ведения проектов с использованием доски Kanban;
* построение отчетов о затраченном времени;
* поиск задач;
* создание проектов, ролей, управление списком пользователей, правами.

Серверная часть разработанной системы реализована на языке Java, с использованием фреймворка Spring. Клиентская часть написана с использованием фрейворка Angular на языке JavaScript. В качестве СУБД использовалась MySQL.

ПРИЛОЖЕНИЕ А

(обязательное)

Программное средство Kanban board. Плакат.

Приложение Б

(справочное)

Программное средство kanban board. Схема структурная.