**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 совершать следующие действия: управление задачей и заполнение отчетов о затраченном времени. Руководитель проекта наследует дейтсвия, которые может совершать пользователь и добавляет следующие: создание задач, назначение проектной группы, назначение прав для группы, отслеживание статистики по задачам. Администратор системы наследует действия, которые может совершать руководитель проекта и добавляет следующие: создание проекта, назначение руководителя проекта.

**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 взаимодействут с другими модулями посредством интерфейсов.

Рассмотрим функциональные модули разрабытываемой системы.

*Модуль API веб-сервиса.* Этот блок является своего рода «ядром» настоящего прораммного 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 алгоритм шифрования bcrypt. Данный алгоритм шифрования является «односторонним», это значит что не существует способа по ключу получить исходный пароль. При авторизации из базы данных извлекается ключ пароля, затем пароль введенный пользователем шифруется с помощью bcrypt и сравниваются два полученных ключа. При успешной авторизации пользователь перенаправляется на главную страницу приложения, в противном случае происходит перенаправление на страницу авторизации. Блок аутентификации реализуется с помощю модуля Spring Security фреймворка Spring.

*Модуль отправки уведомлений.* Данный модуль предназначен для отправки сообщений по электронной почте пользователям. Данный модуль может использован для отправки уведомлений при смене пароля, при восстановлении пароля, при измениях в проекте и др. Данный модуль реализуется с использованием библиотеки JavaMail и ее обертки spring-boot-starter-mail.

*Модуль построения отчетов.* Модуль построения отчетов необходим для получения статистики по задачам, имеющимся на проекте, а так же для отправки отчетов пользователем по 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 виде json и делигирует данный запрос модулю API веб-сервиса. Получив ответ от модуля API веб-сервиса модуль маршрутизации отправляет клиенту ответ в формате json, html, jpeg или другом формате.

*Модуль управления задачами.* Данный модуль предназначен для создания задач, смены статуса задачи поиска задач и т.д.

*Модуль администрирования и авторизации.* Данный модуль используется для выяснения прав, которыми обладает пользователь, 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: право на создание задач, право на смену статуса задач, может ли пользователь выбирать задачу спмостоятельно, может ли пользователь назначать задачи другим пользователям.

*Модуль доступа к данным.* Данный модуль является своего рода адаптером между базой данных и непосредственно веб-сервисом. Такой шаблон проектирования называется DAO (Data Access Object) (Объект для доступа к данным). 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 фрейсворка осуществляется рендеринг html-страниц и отправка http-запросов веб-сервису.

**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.