**2** СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

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

В при изображена диаграмма использования разрабатываемого программного продукта. В системе можно выделить четыре актера: пользователь, исполнитель запросов и непосредственно программное средство.

Все процессы, происходящие в системе сводятся к запросу к программному средству. Пользователь может отправлять запросы, ранжировать проблемы и предложения, оценивать исполнителей запросов. В обязанности исполнителя запросов входит обработка запросов и удовлетворение запросов пользователей.

Исходя из функций пользователей системы, была проведена декомпозиция программного средства на блоки, что позволяет реализовать концепцию слабой связанности элементов приложения.

**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 прямой и обратной связи;

– модуль ранжирования проблем и предложений;

– модуль маршрутизации;

– модуль авторизированной системы учёта заявок от клиентов;

– модуль администрирования и авторизации;

– модуль доступа к данным;

– база данных веб-сервиса.

На стороне веб-сайта можно выделить следующие функциональные блоки:

– модуль пользовательского интерфейса;

– модуль взаимодействия с веб-сервисом;

– модуль визуализации;

Структурная схема, где представлены все вышеперечисленные блоки приведена на чертеже ГУИР.400201.187 С1. В системе каждый модуль выполняет свою задачу и взаимодействует с другими модулями посредством интерфейсов. Рассмотрим функциональные модули разрабатываемой системы.

*Модуль API веб-сервиса.* Данный модуль является самым важным модулем программного продукта. Данный модуль принимает запросы от модуля маршрутизации, 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 хешированном 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 прямой и обратной связи.* Данный модуль во многом раскрывает суть данного дипломного проекта: обеспечить связь между разработчиком программного средства и пользователем, у которого есть вопросы и предложения по данному продукту. Данный модуль предполагает обеспечить клиентам связь с разработчиками в публичном виде, где запрос и ответ будут видеть все пользователи. Также будет реализован способ связи по электронной почте, при котором запрос и ответ на него будет видеть пользователь, который отправил данный запрос.

*Модуль ранжирования проблем и предложений.* Данный модуль позволяет решать наиболее актуальные проблемы продукта. Ранжирование обратной связи пользователей заключается в том, что клиенты видят, какие жалобы и предложения и предложения появляются по программному продукту и могут их отмечать, как наиболее актуальные. Чем более актуальна проблема, тем выше она будет находиться в списке проблем, что позволит разработчикам программного обеспечения исправлять в первую очередь наиболее важных недочётов

*Модуль маршрутизации.* 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 авторизированной системы учёта заявок от клиентов.* Данный модуль предназначен для реализации helpdesk — ресурса, предназначенного для предоставления клиенту или конечному пользователю информации и поддержки, связанных с продуктами и услугами компании или учреждения.

*Модуль администрирования и авторизации*. В данном модуле сосредоточены функции получения прав пользователя, назначения прав пользователям, а также 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 под названием «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 визуализации.* 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 отправляются на веб-сервер в асинхронном режиме с использованием ajax.

*Модуль пользовательского интерфейса.* Данный модуль по сути является фреймворком AngularJS, упрощающим разработку клиентской части веб-приложения. С помощью этого фреймворка осуществляется рендеринг html-страниц и отправка http-запросов веб-сервису.

В данном разделе 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. В приложении Г представлена 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 *user\_id* (id пользователя). Также присутствуют поля *f\_name* (имя пользователя)*, s\_name* (фамилия пользователя)*,* *user\_login* (логин пользователя)*, user\_password* (пароль пользователя)*, user\_role* (роль пользователя).

Таблица *Request* используется для хранения проблем и предложений, которые интересуют клиентов некоторого программного продукта. В качестве первичного ключа используется поле *req\_id* (искусственное число). Внешний ключ *req\_status* ссылается на таблицу *request\_status*, тем самым указывая текущий статус задачи. Внешний ключ *req\_priority* ссылается на таблицу *request\_priority*, тем самым указывая текущий приоритет задачи. Внешний ключ *req\_executor* ссылается на таблицу *executor* и указывает исполнителя задачи. Внешний ключ *creator\_id* ссылается на таблицу *User* и указывает текущего создателя задачи. Внешний ключ *req\_answer* ссылается на таблицу *answer* и указывает ответ на текущий запрос. Также присутствуют поля *req\_name* (имя запроса), *req\_description* (описание запроса), *req\_mark* (оценка запроса, с помощью которой выполняется ранжирование проблем).

Таблица *request\_status* используется для хранения статуса запроса в рамках проекта. Поле *status\_id* используется в качестве первичного ключа. Ключ *status\_name* используется для того, чтобы указать текущий статус данного запроса. Существует 5 видов текущего статуса запроса: open, distributed, processing, checking, close.

Таблица *request\_priority* используется для хранения приоритета запроса в рамках проекта. Поле *priority\_id* используется в качестве первичного ключа. Ключ *priority\_name* используется для того, чтобы указать текущий приоритет данного запроса. Существует 4 вида текущего приоритета запроса: low, medium, high, critical.

*Lifecycle.* Данная таблица используется для хранения жизненного цикла запроса. Поле *req\_id* здесь является первичным ключом. Также присутствуют поля, описывающие сам жизненный цикл определенного запроса: opened (дата открытия запроса), distributed (дата распределения запроса), processing (дата обработки запроса), checking (дата проверки запроса), close (дата закрытия запроса).

Таблица *answer* предназначена для хранения прикрепленных к запросу ответов. Внешний ключ *req\_id* ссылается на таблицу *Request* и является первичным ключом. Также присутствует поле answer\_content, которое хранит сам ответ на данный запрос.

Таблица *executor* используется для хранения информации об исполнителе запросов. Внешний ключ *executor\_id* ссылается на таблицу *User* и является первичным ключом. Поле *req\_closed* позволяет хранить информацию о том, сколько запросов закрыл исполнитель. Поле *executor\_rating* хранит рейтинг исполнителя, который выставляется на основе времени и качества решения проблем клиентов.

Таблица *role* используется для хранения ролей пользователей в рамках проекта. Внешний ключ *role\_id* является первичным ключом. Также присутствует поле *role\_name*, которое определяет роль пользователя (администратор, модератор, исполнитель, простой пользователь).

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

**2.3** Инфраструктура приложения

В приложении Г представлена диаграмма развертывания разрабатываемого программного средства.

Для работы приложения со стороны веб-сайта (клиент приложения) необходимо устройство с установленным веб-браузером. На серверной части приложения необходимы: контейнер сервлетов Tomcat, платформа Java SE Runtime Environment а так же СУБД MySQL.

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