**2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ**

Проанализировав теоретическую часть разрабатываемой системы, был получен список требований, который необходим для эффективного, стабильного функционирования разрабатываемого веб-приложения. В ходе анализа требований было принято решение разделить программный продукт на функциональные блоки. Распределение функционала по блокам обеспечит удобство в разработке, а также предоставит возможность проверки основных характеристик блока без внешнего воздействия. Выделенные функциональные блоки:

- блок базы данных;

- блок авторизации пользователя;

- блок взаимодействия с корзиной;

- блок работы сервера;

- блок управления позициями;

- блок взаимодействия с позициями;

- блок администрирования;

- блок пользовательского интерфейса.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.078 C1.

Рассмотрим каждый структурный блок подробнее.

**2.1 Блок базы данных**

Блок базы данныхвключает данные, используемые веб-приложением. При реализации используется реляционная база данных PostgreSQL. От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, т.е. атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability (ACID). Существует обширный список типов данных, которые поддерживает PostgreSQL. Кроме числовых, с плавающей точкой, текстовых, булевых и других ожидаемых типов данных (а также множества их вариаций), PostgreSQL имеет преимущество в качестве поддержки uuid, денежного, перечисляемого, геометрического, бинарного типов, сетевых адресов, битовых строк, текстового поиска, xml, json, массивов, композитных типов и диапазонов, а также некоторых внутренних типов для идентификации объектов и местоположения логов.

Основными достоинствами PostgreSQL являются:

- надежность (полное соответствие принципам ACID - атомарность, непротиворечивость, изолированность, сохранность данных);

- производительность (основывается на использовании индексов, интеллектуальном планировщике запросов, тонкой системы блокировок, системе управления буферами памяти и кэширования, превосходной масштабируемости при конкурентной работе);

- расширяемость (означает, что пользователь может настраивать систему путем определения новых функций, агрегатов, типов, языков, индексов и операторов);

- поддержка SQL;

- поддержка JSON;

- богатый набор типов данных;

- простота использования.

**2.2 Блок авторизации пользователя**

Блок авторизации пользователяявляется частью системы, которая отвечает за проверку существования пользователя и в случае его существования в системе, генерирует авторизационный токен для этого пользователя. Для данных целей будет использоваться популярная в Rails среде библиотека Devise. Данная библиотека состоит из следующих модулей:

- Database Authenticatable – хэширует и сохраняет пароль в базе данных для проверки подлинности пользователя при входе в систему;

- Confirmable – отправляет электронные письма с инструкциями по подтверждению аккаунта и проверяет подтверждена учётная записи или нет;

- Recoverable – при необходимости сбрасывает пароль и отправляет инструкции по восстановлению;

- Registerable – предоставляет механизм регистрации нового пользователя;

- Trackable – запоминает количество входов, время и IP-адрес;

- Timeoutable – закрывает открытые сессии пользователя если они не были активны заданный период времени;

- Validatable – проверяет при входе корректность введённых данных так же можно определить свои валидации;

- Lockable – блокирует аккаунт после определенного количества неудачных попыток входа. Может быть разблокирован по электронной почте или по истечении указанного периода времени;

- Rememberable – добавляет возможность «запомнить аккаунт»;

- Omniauthable – поддержка авторизации через социальные сети.

В разрабатываемом программном продукте будут использоваться только некоторые из этих модулей, а именно: Database Authenticatable, Recoverable, Rememberable, Validatable, Registerable. Так же присутствуют методы, упрощающие работу с пользователями, которые будут подробно описаны в следующих главах.

**2.3 Блок администрирования**

Блок администрирования –это часть системы, отвечающая за управление всеми ресурсами сайта. В Rails для этого есть несколько решений, в данном проекте используется плагин Rolify. Rolify предоставляет возможность назначать зарегистрированным пользователям роли, с помощью которых каждому отдельному пользователю предоставляются доступные для данной роли действия. Для роли admin в пользовательском интерфейсе открываются дополнительные опции для доступа ко всем ресурсам приложения, а также появляется возможность для проведения любых манипуляций с данными. Кроме того, стоит отметить, что Rolify гибко настраивается в зависимости от потребностей проекта.

**2.4 Блок пользовательского интерфейса**

Блок пользовательского интерфейса является клиентской частью веб-приложения. Данный блок представляет собой совокупность средств, при помощи которых пользователь взаимодействует с приложением через браузер. Для построения интерфейса будет использоваться несколько технологий: JavaScript, JQuery. Также в Rails есть расширение HTML файлов «.erb», которое позволяет использовать Ruby-код непосредственно в HTML-разметке и предоставляет множество заготовленных хелперов для отрисовки элементов. JQuery – используется для Ajax-запросов на сервер. Ajax – это технология, которая позволяет обновлять данные не перезагружая страницу полностью.

**2.5 Блок взаимодействия с корзиной**

Блок взаимодействия с корзиной – это блок, основной задачей которого является хранение и помещение в него товаров пользователем. В дальнейшем из продуктов, которые хранятся в корзине, формируется заказ. Для реализации данного блока используется Redis. Redis – это быстрое хранилище данных типа «ключ-значение». Оно используется как для баз данных, так и для реализации кэша из-за быстрой скорости обработки операций (хранит базу данных в оперативной памяти).

**2.6 Блок веб-сервера**

Блок веб-сервера *–* это блок, который обрабатывает запросы, отправленные клиентом, в качестве серверной части будет использоваться фреймворк Ruby On Rails.

Фреймворк Ruby On Rails написан с помощью языка программирования Ruby.

Ruby – интерпретируемый язык программирования высокого уровня. Язык динамический, объектно-ориентированный, реализует многопоточность, которая не зависит от операционной системы, имеет «сборщик мусора».

Ruby on Rails предоставляет из себя архитектуру MVC для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером базы данных. Ruby on Rails определяет следующие принципы разработки приложений, помогающие разработчику в создании элегантных программных решений, усвоенные сообществом разработчиков:

- предоставляет механизмы повторного использования, позволяющие минимизировать дублирование кода в приложениях (принцип Don’t repeat your self);

- по умолчанию используются соглашения по конфигурации, типичные для большинства приложений (принцип Convention over configuration);

- основными компонентами приложений Ruby on Rails являются модель, представление и контроллер;

- Ruby on Rails использует REST-стиль построения веб-приложений.

Модель используется для отображения данных остальным компонентам приложения. Объекты модели производят загрузку и сохранение данных в базе данных, а также реализуют бизнес-логику.

Для хранения объектов модели в СУБД по умолчанию в Rails использована библиотека Active Record.

Представление создает пользовательский интерфейс с использованием полученных от контроллера данных. Представление также передает запросы пользователя на взаимодействие с данными в контроллер (само представление обычно не изменяет модель). Контроллер в Rails – это набор логики, запускаемой после получения HTTP-запроса сервером. Контроллер отвечает за вызов методов модели и запускает формирование представления.

Вокруг Rails сложилась большая экосистема плагинов – подключаемых «гемов» (англ. gem). «Гем» – это библиотека, составленная определенным образом. То есть это набор кода (модули, классы, представления и так далее) которые решают определенную задачу. Утилита gem занимается тем, что управляет этими библиотеками. Например 'gem install colorize' скачает из интернета библиотеку (далее "гем") в каталог, в который прежде был установлен Ruby. После чего в коде можно написать require ‘colorize’ и пользоваться методами, которые данный гем предоставляет. Гем может требовать для установки другие гемы. Для того чтобы не ставить/обновлять гемы по-одному каждый раз, был создан Bundler. который сам по себе является гемом. Работает это следующим образом: в Gemfile описываются требуемые в данном конкретном проекте гемы. После чего запускается команда bundle install. Bundler просматривает Gemfile проекта и устанавливает (с помощью утилиты gem) нужные гемы, а также создаёт файл Gemfile.lock, в котором описывает установленный гем и, если понадобились при установке, то дополнения к данному гему.

**2.7 Блоки управления и взаимодействия с позициями**

Блоки управления объявлениями и взаимодействия с позициями призваны отвечать за работу с позициями. Через эти блоки проводятся базовые CRUD-операции с позициями. Аббревиатура CRUD в переводе с английского означает C(create) – создать, R(read) – читать, U(update) – обновлять, D(delete) – удалять. Это основные операции, используемые для реализации приложений с постоянный хранением данных и приложений реляционных баз данных.

CRUD постоянно используется для всего, что связано с базами данных и их проектированием. Разработчики программного обеспечения ничего не могут сделать без операций CRUD. Например, при разработке веб-сайтов используется REST (передача репрезентативного состояния), который является надмножеством CRUD, используемого для ресурсов HTTP.

С другой стороны, CRUD не менее важен для конечных пользователей. Без него были бы невозможны такие базовые операции как регистрация пользователя, размещение новых записей. Большинство приложений, которые мы используем, позволяют нам добавлять или создавать новые записи, искать существующие, вносить в них изменения или удалять их.

Create позволяет вносить новые строки в таблицу базы данных. Сделать это можно с помощью команды INSERT INTO. Команда начинается с ключевого слова INSERT INTO, за которым следует имя таблицы, имена столбцов и значения, которые нужно вставить.

Функция чтения соизмерима с функцией поиска, поскольку позволяет извлекать определенные записи и считывать их значения. Она относится к команде SELECT.

Обновление – это изменение существующей записи в таблице. Это используется для внесения изменений в существующие записи в таблице базы данных. При выполнении команды UPDATE определяется целевая таблица и столбцы, которые необходимо обновить.

Удаление используется для удаления записи из таблицы базы данных. SQL имеет встроенную функцию удаления для одновременного удаления одной или нескольких записей из базы данных. Некоторые приложения реляционных баз данных могут разрешать жесткое удаление (безвозвратное удаление) или мягкое удаление (обновление статуса строки).