2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Изучив теоретические аспекты разрабатываемой системы и выработав список требований необходимых для разработки системы, разбиваем систему на функциональные блоки(модули). Это необходимо для обеспечения гибкой архитектуры. Такой подход позволяет изменять или заменять модули без изменения всей системы в целом.

В разрабатываемом веб-приложении можно выделить следующие блоки:

- блок интерфейса пользователя;
- блок модели базы данных;
- база данных веб-приложения;
- блок регистрации;
- блок авторизации;
- блок ядра веб-приложения.

В управляющей программе можно выделить следующие блоки:

- блок ядра управляющей программы;
- блок определения движения;
- блок работы с данными на SD-карте;
- блок получения изображения с камеры;
- блок взаимодействия с веб-приложением.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.047 С1.

Каждый модуль выполняет свою задачу. Чтобы система работала каждый модуль взаимодействует с другими модулями путем обмена данными, используя различные форматы и протоколы.

Рассмотрим функциональные блоки веб-приложения.

Блок интерфейса пользователя является клиентской частью веб-приложения. Данный блок представляет собой совокупность средств, при помощи которых пользователь взаимодействует с приложением через браузер. Для построения интерфейса используется технология Haml. Haml – язык разметки для упрощенной генерации HTML, который компилируется в HTML. Он позволяет представлять данные, а также обрабатывать ввод пользователя. Официальная реализация этого языка написана на Ruby.

Блок модели базы данных представляет из себя набор моделей определенных библиотекой ActiveRecord в фреймворке RubyOnRails, по сути является ключевым модулем на стороне сервера, содержит в себе схему базы данных и отвечает за генерацию запросов к СУБД с последующим «маппингом» полученных данных в привычные и удобные объекты языка Ruby.

База данных веб-приложения включает данные, используемые веб-приложением. При реализации использовалась база данных PostgresSQL. Это свободно распространяемая объектно-реляционная система управления

базами данных наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

Основными достоинствами PostgresSQLявляются:

– 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;

– богатый набор типов данных;

– простота использования;

– безопасность данных.

Блок регистрации отвечает за регистрацию пользователей и сохранение их данных в базу данных веб-приложения.

При реализации использовался плагин Devise. Этот плагин предоставляет решение с весьма гибкими настройками для задач регистрации и аутентификации пользователей.

Блок авторизации отвечает за авторизацию пользователей и разделение прав доступа к ресурсам веб-приложения.

При реализации использовался плагин Devise, рассмотренный выше, а также плагин Cancancan. Cancancan позволяет устанавливать различные уровни доступа для пользователей, таким образом ограждая определнные ресурсы от несанкционированного доступа.

Блок ядра веб-приложения представляет встроенное RubyOnRails приложение.

Фреймворк RubyOnRails написан с помощью языка программирования Ruby[7].

Ruby – интерпретируемый язык программирования высокого уровня. Обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, «сборщиком мусора» и многими другими возможностями, поддерживающими много разных парадигм программирования, прежде всего объектно-ориентированную.

RubyOnRails предоставляет из себя архитектуру MVC для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером базы данных. RubyOnRails 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

yourself);

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

– основными компонентами приложений RubyonRails являются модель, представление и контроллер;

– RubyOnRails использует REST-стиль построения веб-приложений.

Модель предоставляет остальным компонентам приложения объектно-ориентированное отображение данных (таких как каталог продуктов или список заказов). Объекты модели могут осуществлять загрузку и сохранение данных в реляционной базе данных, а также реализуют бизнес-логику.

Для хранения объектов модели в реляционной СУБД по умолчанию в Rails использована библиотека ActiveRecord. Конкурирующий аналог – DataMapper. Существуют плагины для работы с нереляционными базами данных, например Mongoid для работы с MongoDB.

Представление создает пользовательский интерфейс с использованием полученных от контроллера данных. Представление также передает запросы пользователя на манипуляцию данными в контроллер (как правило, представление не изменяет непосредственно модель).

В RubyonRails представление описывается при помощи шаблонов ERB. Они представляют собой файлы HTML с дополнительными включениями фрагментов кода Ruby (EmbeddedRuby или ERB). Вывод, сгенерированный встроенным кодом Ruby, включается в текст шаблона, после чего получившаяся страница HTML возвращается пользователю. Кроме ERB возможно использовать ещё около 20 шаблонизаторов, в том числе Haml и Slim.

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), некоторые из них со временем были включены в базовую поставку Rails, например Sass и CoffeeScript, другие же, хотя и не были включены в базовую поставку, являются фактическим стандартом для большинства разработчиков, например, средство модульного тестирования RSpec.

Теперь рассмотрим функциональные блоки управляющего приложения.

Блок ядра управляющей программы является основным блоком управляющей программы. Данный блок представляет собой приложние, написанное на языке Python

Python – это интерпретируемый язык программирования с динамической типизацией. Язык поддерживает несколько парадигм программирования и имеет богатую стандартную библиотеку.

Блок ядра управляющей программы отвечает за инициализацию переменных, а также управляет остальными функциональными блоками управляющей программы.

Блок определения движения отвечает за обработку сигнала, поступающего с датчика движения, также уведомляя об этом блок получения изображения с камеры.

Блок получения изображения с камеры позволяет послать сигнал камере, при котором она сделает снимок. Этот снимок обрабатывается данным блоком и пересылается в блок работы с данными на SD-карте.

Блок работы с данными на SD-карте реализует операции записи данных на SD-карту и чтения с нее.

Блок взаимодействия с веб-приложением позволяет отправлять фотографии, полученные с камеры и хранящиеся на SD-карте на сервер нашего веб-приложения.