**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 пользовательского интерфейса;
* блок модели базы данных;
* блок реляционной базы данных;
* блок базы данных для визуализации;
* блок веб-сервера.

В мобильном приложении можно выделить следующие блоки:

* блок пользовательского интерфейса;
* блок модели базы данных;
* блок реляционной базы данных;
* блок взаимодействия с веб-сервером.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.064 C1.

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 пользовательского интерфейса* является клиентской частьювеб-приложения. Данный блок представляет собой совокупность средств, при помощи которых пользователь взаимодействует с приложением через браузер. Для построения интерфейса используется технология Reactjs.Reactjs - фреймворк для создания интерфейсов, созданный компанией Facebook. Он отвечает за представление данных, получение и обработку ввода пользователя. Reactjs построен на парадигме реактивного программирования. Этот декларативный подход предлагает описывать данные в виде набора утверждений или формул. Изменение одного из параметров ведёт за собой автоматический пересчёт всех зависимостей.

*Блок модели базы данных*представляет из себя набор моделей определенных библиотекой ActiveRecord в фреймворкеRubyOnRails, 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.

*Блок реляционной базы данных*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 базы данных для визуализации*включает данные, используемые веб-приложением для хранения и выборки статистических и аналитических данных для дальнейшего построения диаграмм и графиков.

При реализации использовалась база данныхInfluxDB. Это база данных для хранения временных рядов, метрик и информации о событиях.В числе преимуществ InfluxDB в первую очередь нужно выделить следующие:

* отсутствие зависимостей (следствие того, что она написана на Go);
* возможность работы в том числе и в кластерном режиме;
* наличие библиотек для большого числа языков программирования (Python, JavaScript, PHP, Haskell и других);
* SQL-подобный язык запросов, с помощью которого можно производить различные операции с временными рядами (объединение, слияние, разбиение на части);
* удобный графический интерфейс для работы с БД;
* широкие возможности интеграции с другими программными продуктами.

*Блок веб-сервиса*представляет встроенноеRubyOnRails приложение с использованием фреймворкаGrape, который предназначен для удобного построения API веб-сервисов.

Фреймворк RubyOnRails написан с помощью языка программирования Ruby[4].

Ruby – [интерпретируемый](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80)[язык программирования высокого уровня](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Обладает независимой от [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) реализацией [многопоточности](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C), строгой [динамической типизацией](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [«сборщиком мусора»](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D1%89%D0%B8%D0%BA_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0) и многими [другими возможностями](https://ru.wikipedia.org/wiki/Ruby#.D0.92.D0.BE.D0.B7.D0.BC.D0.BE.D0.B6.D0.BD.D0.BE.D1.81.D1.82.D0.B8_Ruby), поддерживающими много разных [парадигм программирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), прежде всего [объектно-ориентированную](https://ru.wikibooks.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

RubyonRails предоставляет из себя архитектуру [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller)для [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), а также обеспечивает их интеграцию с [веб-сервером](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) и сервером базы данных. RubyonRails определяет следующие принципы разработки приложений, помогающие разработчику в создании элегантных программных решений, усвоенные сообществом разработчиков:

* предоставляет механизмы [повторного использования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0), позволяющие минимизировать дублирование кода в приложениях (принцип [Don’trepeatyourself](https://ru.wikipedia.org/wiki/Don%E2%80%99t_repeat_yourself));
* [по умолчанию](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE_%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E) используются соглашения по конфигурации, типичные для большинства приложений (принцип Conventionoverconfiguration);
* основными компонентами приложений RubyonRails являются модель, представление и контроллер;
* RubyonRails использует [REST](https://ru.wikipedia.org/wiki/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](https://ru.wikipedia.org/wiki/ActiveRecord). Конкурирующий аналог – DataMapper. Существуют плагины для работы с [нереляционными базами данных](https://ru.wikipedia.org/wiki/NoSQL), например Mongoid для работы с [MongoDB](https://ru.wikipedia.org/wiki/MongoDB).

Представление создает [пользовательский интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) 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](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD) ERB. Они представляют собой файлы [HTML](https://ru.wikipedia.org/wiki/HTML) с дополнительными включениями фрагментов кода [Ruby](https://ru.wikipedia.org/wiki/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](https://ru.wikipedia.org/wiki/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](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD) – подключаемых «[гемов](https://ru.wikipedia.org/wiki/RubyGems)» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) gem), некоторые из них со временем были включены в базовую поставку Rails, например [Sass](https://ru.wikipedia.org/wiki/Sass) и [CoffeeScript](https://ru.wikipedia.org/wiki/CoffeeScript), другие же, хотя и не были включены в базовую поставку, являются фактическим стандартом для большинства разработчиков, например, средство [модульного тестирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)RSpec.

Теперь рассмотрим функциональные блоки мобильного приложения.

*Блок пользовательского интерфейса*является основным блоком взаимодействия пользователя и мобильного приложения. Данный блок был разработан с помощью приложения InterfaceBuilder [5]. Interface Builder – является частью Xcode, специальной системы инструментов для разработчиков Apple Developer Connection. Interface Builder позволяет Cocoa и Carbon разработчикам создавать графические интерфейсы для приложений. Результат разработки хранится в файле с расширением .nib, сокращение от NeXT Interface Builder, хотя в последнее время чаще используется, .xib.

InterfaceBuilder предоставляет палитры, или коллекций, объектов пользовательского интерфейса для Objective-C разработчиков. Эти объекты пользовательского интерфейса содержат такие элементы, как текстовые поля, таблицы данных, слайдеры и всплывающие меню. Палитры InterfaceBuilder являются полностью расширяемыми, то есть любой разработчик может разрабатывать новые объекты и добавлять их к палитре IB.

Для создания интерфейса, разработчик просто перетаскивает элементы интерфейса с палитры на окно или меню. Конкретные объекты, которые получают сообщения указываются в коде приложения. Таким образом все инициализации происходят до выполнения, что ведет к повышению производительности и делает процесс разработки более упорядоченным.

*Блок модели базы данных* представляет из себя модуль для взаимодействия с базой данных, которая находиться в приложении. Было решено использовать технологию CoreData, которая обеспечивает инфраструктуру для управления изменениями и для сохранениями объектов и извлечения их из хранилища. Она позволяет получать данные, организованные в виде отношения модели сущность-атрибут для сериализации в XML, бинарное или SQLiteхранилище. CoreData не идет ни в какое сравнение по скорости и количеству кода с другими реализациями. Основные возможности CoreData:

* операции со сложными графами объектов;
* операции с множественными связями;
* операции с валидацией.

*Блок реляционной базы данных*включает данные, используемые мобильным приложением. Для ее реализации была выбрана база данных SQLite. SQLite - компактнаявстраиваемаяреляционнаябазаданных,  запросы к которой можно осуществлять при помощи языка запросов SQL. База данных не поддерживает все особенности SQL и уступает в функциональности другим развитым СУБД, но вполне подходит для хранения и извлечения информации.

SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются функции (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени.

Благодаря архитектуре движка возможно использовать SQLite как на встраиваемых системах, так и на выделенных машинах с гигабайтными массивами данных.

*Блок взаимодействия с веб-сервером* – это модуль, основной задачей которого является создание и выполнение запросов к RESTful серверу веб-приложения, а также получение ответа и его дальнейшая обработка. Для его реализации была ислользована библиотека AFNetworking. Это библиотека, котораяпредставляетсобойоберткунадвстроеннымиинструментамивзаимодействиямобильногоприложения с сетью, расширяя их возможности и упрощая интерфейс их использования.