МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет комп’ютерних наук

Кафедра програмної інженерії

ЗВІТ З ПЕРЕДДИПЛОМНОЇ ПРАКТИКИ

Місце проходження практики «ЕПАМ СИСТЕМЗ»

у період з «18» квітня 2016 р. по «14» травня 2016 р.

Тема індивідуального завдання:

Програмна система «Мобільна прикладна програма для проведення екскурсій»

Студент гр. ПІ-12-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Войтенко М.В.

Керівник з практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ас. Ляпота В.М.

Роботу захищено «\_\_»\_\_\_\_\_\_2016р.

з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник випускної кваліфікаційної роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Турута О.П.

Рекомендована оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Харків 2016

РЕФЕРАТ

Пояснювальна записка: 30 сторінок, 2 розділи, 11 рисунків, 10 джерел та 1 додаток.

Метою роботи є аналіз існуючих стану сучасної туристичної галузі, проектування прикладної програми для проведення екскурсій, програмна реалізація та порівняння результатів розробки та існуючих рішень.

Метод розробки – об'єктно-орієнтований підхід до розробки програмного додатка, застосування паттернів розробки, написання тестів для перевірки правильності роботи системи.

В результаті була розроблена програмна система для проведення екскурсій, яка складається з серверу, веб-клієнта та мобільного клієнта та проведений порівняльний аналіз розробленого продукту з аналогами.

ВИПУСКНА РОБОТА БАКАЛАВРА, ПРОГРАМНА СИСТЕМА, ТУРИЗМ, МОБІЛЬНИЙ ДОДАТОК, JAVA, ANDROID, КЛІЄНТ, СЕРВЕР, API.

Explanatory note: 30 pages, 2 sections, 11 pictures, 10 sources and 1 application.

The aim is to analyze the current state of the modern tourism industry, design the application for excursions, software implementation and results comparison of this solution and existing solutions.

Method of the development - an object-oriented approach to application software development, applying of design patterns, writing tests to verify the system.

As a result, a software system for the tours has been developed, consiststing of the server, Web client and mobile client and comparative analysis of product developed counterparts.

FINAL WORK OF BACHELOR, SOFTWARE SYSTEM, TOURISM, MOBILE APPLICATION, JAVA, ANDROID, CLIENT, SERVER, API.

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Вступление | 4 |
| 1 Анализ предметной области и постановка задачи | 5 |
| 1.1 Анализ предметной области разработки | 5 |
| 1.2 Анализ аналогов | 7 |
| 1.3 Постановка задачи | 10 |
| 2 Проектирование программного обеспечения | 11 |
| 2.1 Программные средства | 11 |
| 2.2 Архитектура ПО | 13 |
| 2.3 UML-моделирование программной системы | 16 |
| 2.4 Проектирование базы данных | 23 |
| Выводы | 27 |
| Перечень ссылок | 28 |
| Приложение А Описание предприятия | 29 |

ВСТУПЛЕНИЕ

В последние десятилетия количество туристических поездок по всему миру существенно увеличилось. Об этом говорят не только известные статистические порталы и исследовательские организации как Statista и Global Wellness Institute, но и государственные либо туристические сайты отдельных государств. Туризм стал самым быстрорастущим экономическим сектором мира. Туризм стал одним из главных игроков в международной торговле, и представляет собой в то же время один из основных источников дохода для многих развивающихся стран. По всем данным причинам, за 2015 год рост количества туристов в мире вырос на несколько процентов.

При выборе места для отдыха многие интересуются, какие достопримечательности есть в том или ином месте. Как правило, если страна имеет маленькую площадь, люди пытаются посетить ее целиком, но, к сожалению, памятные или интересные места могут быть расположены друг от друга и не иметь достаточного количества информации о себе на различных языках. Зачастую люди заказывают всевозможные туры, которые проводятся опытным экскурсоводом с помощью транспортного средства, но стоит это недешево для любого бюджетного туриста. Таким образом, чтобы хорошо познакомиться с культурой и историей страны, он оформляет несколько таких туров, стоимость которых может составлять большую часть его бюджета. Стоит заметить, что на данный момент существует огромное количество информации о достопримечательностях в Интернете, а также разнообразные мобильные приложения для туризма, но все они рассчитаны на использование перед поездкой, а не во время посещения места. Такого рода программы содержат только набор достопримечательностей, их расположение на карте и краткую информацию о них, что не дает возможности прослушать более подробную лекцию, которая сможет заменить экскурсовода, и сэкономить многим туристам время и деньги.

Таким образом, задача предоставления актуальной и полной информации о достопримечательностях во время пребывания на экскурсии является актуальной проблемой современного туризма, которая будет рассмотрена в данной работе.

Целью данной работы является разработка мобильного клиента для определения местонахождения достопримечательности и предоставления полного записанного материала для прослушивания о ней, разработка сервера для хранения данных и анализ полученного результата.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ

* 1. Анализ предметной области разработки

Понятие туризма определяется как временные выезды людей в другую страну или местность, отличную от места постоянного жительства, на срок от 24 часов до 6 месяцев в течение одного календарного года или с совершением не менее одной ночевки в развлекательных, оздоровительных, спортивных, гостевых, познавательных, религиозных и иных целях без занятия деятельностью, оплачиваемой из местного источника[1]. Таким образом, туризм является важным аспектом жизни многих людей. Туризм дает возможность выйти из обычного окружения и открыть для себя другие части мира. Он помогает объединить людей из разных культур и способствует взаимному уважению между различными сообществами как внутри, так и в разных странах или на разных континентах.

Актуальность темы туризма растет с каждым годом. Все большое количество направлений начинают открываться для поездок, что является причиной социально-экономического прогресса посредством создания рабочих мест и предприятий, получения доходов от экспорта и развития инфраструктуры.

Несмотря на периодические кризисные ситуации, туристическая отрасль показала непрерывный рост за последние шесть десятилетий. Международные туристические поездки увеличились с 25 миллионов в 1950 году, до 278 миллионам в 1980, 527 миллионам в 1995 и 1133 миллионам в 2014 году. Увеличение количества туристов стало причиной такого же динамичного роста прибыли, полученной от туризма. Например, прибыль от международного туризма выросла с 2 миллиардов долларов США в 1950 году, до 104 миллиардов в 1980, 415 миллиардов в 1995 и 1245 миллиардов в 2014 [2]. Ожидается, что рост числа туристов будет увеличиваться по правилу арифметической прогрессии на 3,3% ежегодно с 2010 по 2030 годы, что благоприятно отразится на экономике стран, в которые будут совершаться туристические путешествия. Такой рост можно объяснить все большей доступностью путешествий, так как многие направления сейчас предоставляют все возможности как для бюджетного, так и для дорогого отдыха.

По статистическим данным 2013 года туристическую индустрию можно разделить на следующие направления (с учетом общей суммы в 3,2 триллиона долларов США) [3]:

* культурный (800 миллиардов – 1,1 триллион долларов США);
* оздоровительный (439 миллиардов долларов США);
* кулинарный (350 – 550 миллиардов долларов США);
* поддерживающий экологию (325 – 480 миллиардов долларов США);
* спортивный (250 - 375 миллиардов долларов США);
* приключенческий (115 – 150 миллиардов долларов США);
* агро-туризм (60 – 160 миллиардов долларов США);
* духовный (37 – 47 миллиардов долларов США);
* волонтерский (10 – 20 миллиардов долларов США).

В зависимости от выбранного направления цели туризма могут также варьироваться от культурных до оздоровительных и так далее. На целях поездки основаны и общие затраты на предоставление услуг. Таким образом, основными затратами являются (с учетом мирового рынка оздоровительного туризма в 438,6 миллиардов американских долларов) [3]:

* проживание – отели, мотели, курорты и групповые лагери (93,4 миллиарда долларов США);
* транспорт – авиакомпании, аренда автомобилей, публичные перевозки, поезда и такси (91,8 миллиардов долларов США);
* еда и напитки – рестораны, бары и различные магазины закусок (71,9 миллиард долларов США);
* шоппинг – сувениры, подарки, одежда и предметы искусства (64,8 миллиарда долларов США);
* активности и экскурсии – музеи, туры и театры и посещение достопримечательностей (61,4 миллиарда долларов США);
* другие сервисы – телекоммуникации, страхование, туристические агентства и консьержи (55,3 миллиардов долларов США).

Исходя из этих данных, можно сказать, что туризм является дорогим удовольствием для среднестатистического человека, поэтому так много стран и туристических организаций предоставляет возможность бюджетного отдыха. Стоимость туристической поездки будет основана на ее типе: внутреннем (более дешевом варианте) либо заграничном, который будет более затратным по средствам. Экономить можно на любой из такого рода затрат во время поездки, особенно, на различных активностях и экскурсиях, ведь информацию можно взять не только из туров, но и из свободных источников вроде Интернета и различных приложений.

Наиболее удобным средством получения такого рода информации считаются мобильные приложения, так как рынок мобильных приложений сейчас находится на очень высоком уровне и динамично развивается, а их разработка считается несложным заданием. Поэтому, решение задачи для проведении экскурсий лежит в разработке мобильного приложения.

* 1. Анализ аналогов

В настоящее время существует довольно большое количество сайтов и приложений, посвященных туризму. В основном, это государственные либо относящиеся к государству сайты, цель которых кратко и доступно объяснить туристу хорошие стороны поездки в данную страну и завлечь его своими достопримечательностями и известными событиями. Они содержат большое количество медиа для наглядной демонстрации того, что увидит турист по прибытии в эту страну или конкретное место. Примером такого сайта может быть ireland.com, интерфейс которого показан на рисунке 1.1.

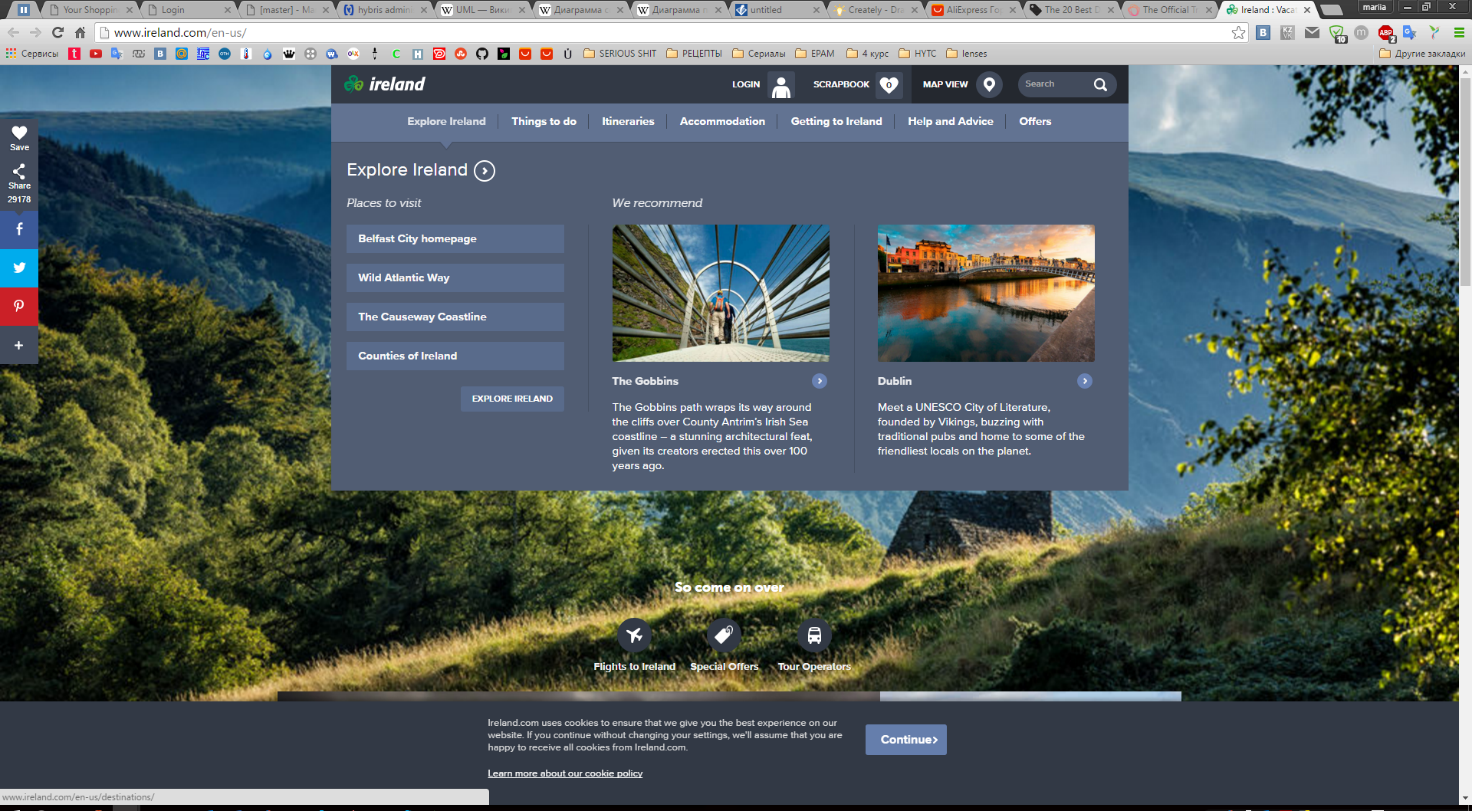


Рисунок 1.1 – Интерфейс сайта, предоставляющего информацию о туризме в Ирландии

В целом, такие ресурсы описывают самые популярные места и также могут содержать отзывы людей либо ссылки на социальные сети, контент которых связан с определенной тематикой сайта. Таким образом, формируется рейтинг достопримечательностей, который позволит туристу спланировать свою поездку, если тот ограничен во времени либо финансах.

Такого рода информационные источники разделяют область туризма на подобласти, предоставляя краткую либо более детальную информацию о каждом из них. Эта информация содержит картинки, карты и, возможно, некоторые исторические данные, связанные с определенным объектом. Некоторые сайты предоставляют запись на различные экскурсии и туры, доступные сразу со страницы с информацией о музеях, выставках и так далее.

Большинство контента сгруппировано по периоду событий либо по интересам туриста, что является удобным в пользовании ресурсом. Также контент может быть оформлен в виде маршрутов посещения интересных мест либо предоставления необычных активностей для туристов, однако, такого рода сайты являются скорее исключением, чем правилом. Примером такого сайта является triphobo.com, интерфейс которого представлен на рисунке 1.2.

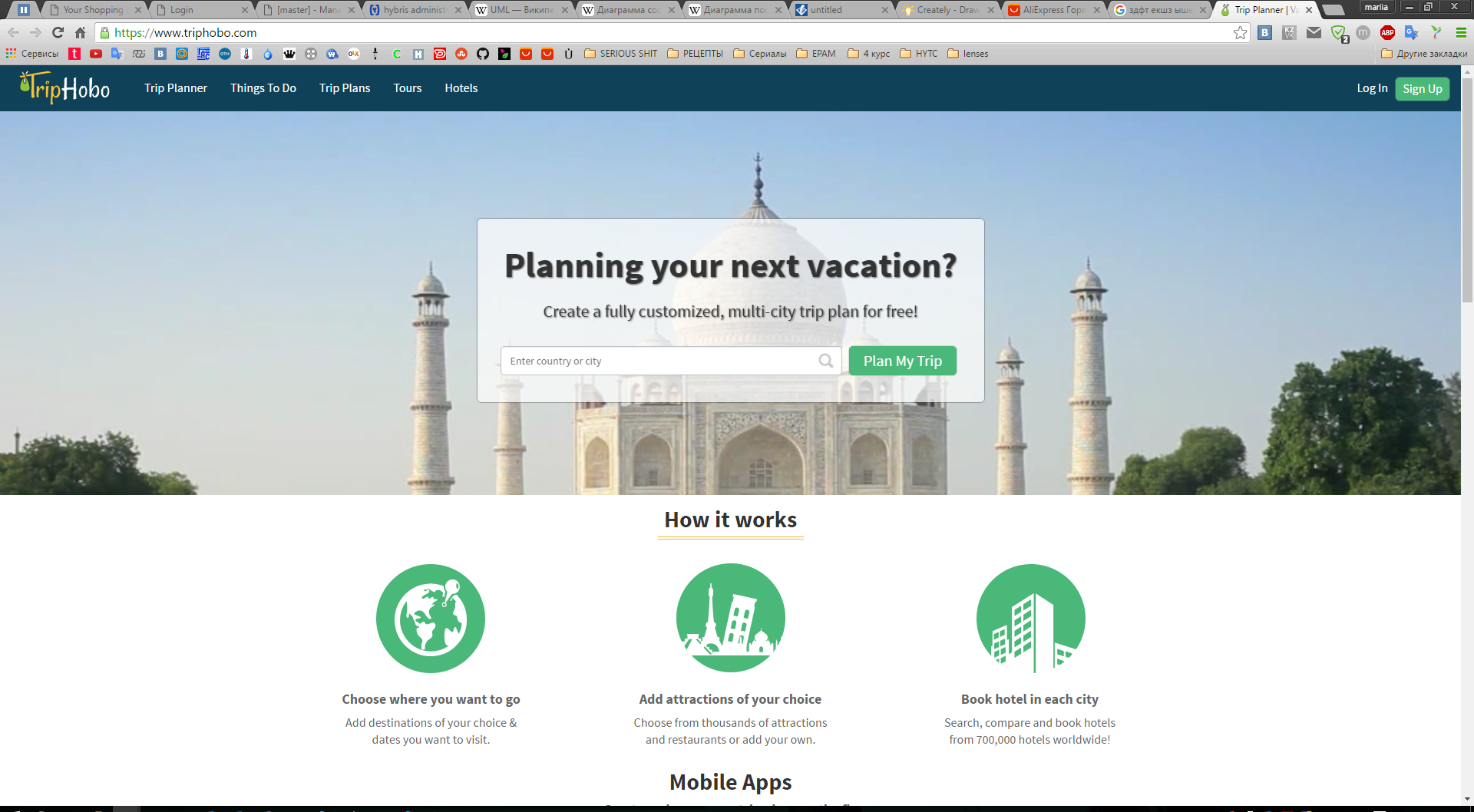


Рисунок 1.2 – Сайт для планирования туристические поездок

Другим способом получить желанную информацию перед поездкой являются мобильные приложения, которые предоставляют различную информацию о конкретных странах либо знаменитых местах. Обычно достопримечательности расположены по пользовательскому рейтингу, основанному на голосовании либо на отзывах о конкретном месте. Приложения также имеют большое количества медиа, но описание места довольно сжатое. Может предоставляться адрес и карта достопримечательности. Некоторые такие приложения рассчитаны на добавление пользовательских фотографий и отзывов с места, которое было выбрано для них данным приложением автоматически. Также возможны поиск отелей и хостелов, ресторанов и кафе, а также развлечений по конкретным разнообразным критериям либо по областям. Примером такого приложения является TripAdvisor, который является одним из самых популярных приложений в разделе «Путешествия» в Android Play Market с рейтингом 4,5 из 5 очков. Графический интерфейс данного приложения представлен на рисунке 1.3.

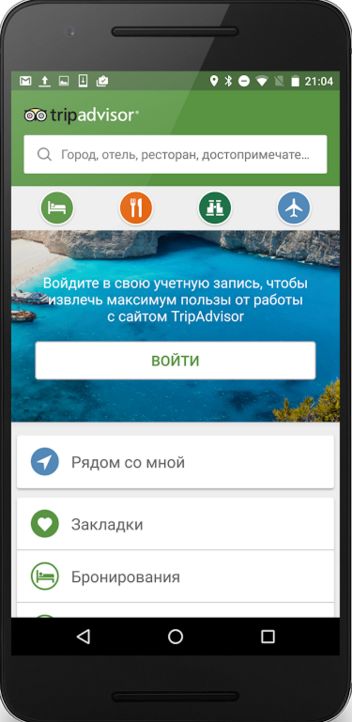


Рисунок 1.3 – Интерфейс TripAdvisor

Большим плюсом таких приложений является функция построения маршрута либо планирования поездки путем сохранения мест в избранные. Главным отличием таких мобильных приложений от сайтов является то, что они больше рассчитаны на персонификацию, нежели на получение большого количества информации об объекте.

Таким образом, проведя анализ доступных источников информации для туриста, было выяснено, что большинство ресурсов рассчитаны на этап планирования поездки, но никак не на период ее проведения, что иногда делает невозможным изменить заранее составленный план.

* 1. Поставновка задачи

Целью выпускной работы является разработка прикладного мобильного приложения для проведения экскурсий, разработка сервера для хранения информации о достопримечательностях, анализ существующих решений и разработка их улучшеной мобильной версии, а также ручное тестирование программной системы. Создание программной системы состоит из таких заданий:

* Анализ существующих решений;
* Разработка базы данных для хранения информации;
* Проектирование архитектуры сервера для предоставления информации;
* Разработка серверной части и разработка открытого API для его использования мобильным приложением;
* Проектирование мобильного клиента;
* Разработка мобильного клиента;
* Заполнение данными серверной части для тестирования программной системы.

Для написания серверной части и предоставления API был выбран фреймворк JHipster, который основан на множестве других фреймворков: Spring, Yeoman, Gulp, Bower, AngularJs, Twitter Bootstrap, Karma. Для разработки мобильного клиента была выбрана платформа Android.

1. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
   1. Программные средства

Для написания данной выпускной работы, а именно ее серверной части с WEB-клиентом сервера, был выбран фреймворк JHipster, который позволяет быстро и качественно написать как серверную, так и клиентскую сторону программной системы. Он позволяет сделать красивое одностраничное приложение, которое динамически подгружает фрагменты страниц, уменьшая время отклика. Фреймворк использует такие технологии для своего WEB-клиента:

* Yeoman – предоставляет экосистему генераторов приложений для создания правильной структуры приложения, основанной на лучших практиках разработки. Он генерирует модульную архитектуру и предоставляет возможность тестирования и минификации;
* Gulp – выполняет задачи Javascript. Он позволяет редактировать код без пересборки проекта, прогоняет тесты в фоновом режиме, сжимает ресурсы и оптимизирует их. Gulp, как система, построена вокруг концепции потоковой передачи данных, которая избавляет разработчика от промежуточных файлов и папок;
* Bower – пакетный менеджер для активов JavaScript, который предоставляет огромное количество всевозможных библиотек. Менеджеры пакетов упрощают установку и обновление зависимостей проекта;
* Karma – средство для запуска Junit-тестов для Javascript, которое предоставляет их быстрый запуск в фоновом режиме. Эта технология, которая выполняет исходный код тестов для каждого браузера. Результаты каждого теста для каждого браузера проверяются и отображаюся в командной строке разработкика;
* AngularJS – фреймворк для JavaScript, который является популярным для разработки одностраничных приложений и предоставляет много функций из коробки. Его цель — расширение браузерных приложений на основе MVC шаблона, а также упрощение тестирования и разработки. Фреймворк работает с HTML, содержащим дополнительные пользовательские атрибуты, которые описываются директивами, и связывает ввод или вывод области страницы с моделью, представляющей собой обычные переменные JavaScript. Значения этих переменных задаются вручную или извлекаются из статических или динамических JSON-данных;
* Twitter Bootstrap – большой CSS/JavaScript фреймворк для предоставления адаптивных WEB-приложений, делающий фронтенд-разработку быстрее и проще за счет готовых шаблонов стилей.

Для серверной части используются следующие технологии:

* Maven – популярный сборщик приложений, написанных на Java. Он содержит примерную структуру приложения и цели для сборки, интегрируется со всеми популярными средами разработки, имеет большую экосистему, включая Spring Boot и Yeoman;
* Spring Boot – является де-факто стандартом для Java приложений, который помогает создавать автономные Spring приложения, содержит встроенный контейнер сервлетов, автоматическую конфигурацию Spring и конфигурацию POM-файла для Maven. Он поддерживает Inverstion of Control паттерн, аспектно-ориентированное программирование и другие удобные шаблоны разработки;
* Liquibase – управляет обновлением схемы базы данных, отлично интегрируется с Spring и JPA. Эта технология предназначена для разработки базы данных многими разработчиками, поддерживает XML, YAML, JSON и SQL форматы ввода данных и не требует живого соединения с базой данных во время внесения изменений;
* JPA – стандартный ORM для Java приложений. Это API, входящий с версии Java 5 в состав платформ Java SE и Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных. Он использует платформенно-независимый объектно-ориентированный язык запросов Java Persistence Query Language;
* Spring Data JPA – добавляет дополнительный синтаксис для JPA и генерирует JPA репозитории автоматически;
* Spring Security – фреймворк для предоставления безопасности, который защищает URL и Spring Beans, работает с cookies и не имеющими состояния архитектурами WEB-приложений;
* Spring MVC REST – REST фреймворк для Java, который поддержуется технологией Spring и может быть протестирован. REST - это архитектурный стиль для проектирования распределенных систем. Он не является стандартом, но определяет ограничения, такие как отсутствие состояний, клиент-серверная взаимосвязь и унифицированный интерфейс. REST не связан строго с HTTP, но чаще всего его ассоциируют именно с ним;
* Thymeleaf – фреймворк для предоставления шаблонов для серверной стороны приложения, который убирает JSP и JSTL на страницах и преобразует их в более понятных синтаксис.

Для разработки мобильного клиента была выбрана технология написания под операционную систему Android. На сегодняшний день число разработанных приложений для данной ОС является наибольшим[4]. Статистика количества приложений предоставлена на рисунке 2.1.

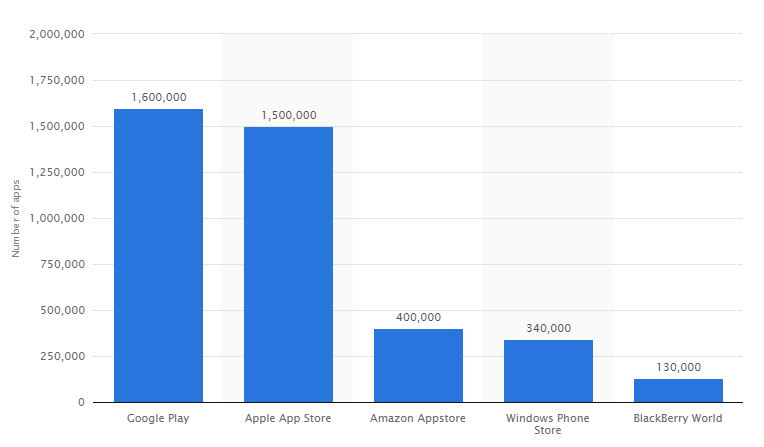


Рисунок 2.1 – Статистика доступных приложений к скачиванию в наиболее популярных магазинах, основанная на данных, полученных в июле 2015

* 1. Архитектура ПО

При проектировании проекта была выбрана классическая трехслойная архитектура, предполагающая наличие в ней трёх компонентов: WEB-клиента и клиента на мобильной платформе Android, сервера приложений (к которому подключены клиентские приложения) и сервера баз данных (с которым работает сервер приложений).

Выбор такого стиля архитектуры обусловлен тем, что трехслойная архитектура обеспечивает, как правило, большую масштабируемость (за счёт горизонтальной масштабируемости сервера приложений и мультиплексирования соединений), большую конфигурируемость (за счёт изолированности уровней друг от друга). Реализация приложений, доступных из веб-браузера или из тонкого клиента, как правило, подразумевает развёртывание программного комплекса в трёхуровневой архитектуре [5].Схема такой архитектуры представлена на рисунке 2.2.

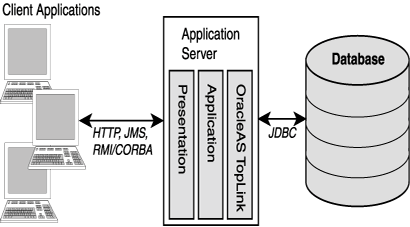


Рисунок 2.2 – Трехслойная архитектура приложения

WEB-сайт построен на классической системе трех модулей – MVC, которая позволяет модифицировать компоненты с минимальным влиянием их друг на друга.

Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

* модель - предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать;
* представление - отвечает за отображение информации (визуализацию). Часто в качестве представления выступает форма (окно) с графическими элементами;
* контроллер - обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

Основная цель применения этой концепции состоит в разделении бизнес-логики (модели) от её визуализации (представления, вида) [6]. За счет такого разделения повышается возможность повторного использования. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения.

При работе с таким видом архитектуры к одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. При использовании MVC можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных), не затрагивая реализацию видов, для этого достаточно использовать другой контроллер. Также ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

Шаблон MVC используется в качестве архитектурной основы во многих фреймворках и CMS, которые создавались для того, чтобы иметь возможность разрабатывать качественно более сложные решения за более короткий срок. На рисунке 2.3 представлен шаблон MVC.



Рисунок 2.3 – Шаблон архитектуры WEB-клиента

Для реализации мобильного клиента была выбрана платформа Android, архитектура которой опирается на шаблон MVVM. Логика пользовательского интерфейса реализуется как компонент ViewModel. Функциональные связи между пользовательским интерфейсом и ViewModel реализуются через биндинги (bindings) [7].

MVVM используется для разделения модели и её представления, что необходимо для изменения их отдельно друг от друга. Шаблон MVVM делится на три части: модель, представление и модель представления.

Модель так же, как в классической MVC, представляет собой фундаментальные данные, необходимые для работы приложения.

Представление — это графический интерфейс. Представление является подписчиком на событие изменения значений свойств или команд, предоставляемых моделью представления. В случае, если в модели представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и представление, в свою очередь, запрашивает обновленное значение свойства из модели представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, мредставление вызывает соответствующую команду, предоставленную моделью представления. На рисунке 2.4 представлена схема модели.



Рисунок 2.4 – Шаблон архитектуры мобильного клиента

Таким образом, программная система имеет сложную, но гибкую и качественно продуманную архитектуру, которая позволит вносить любые изменения налету, без большого изменения структуры приложения.

* 1. UML – моделирование программной системы

UML — язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур [8].

UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML позволяет также разработчикам достигнуть соглашения в графических обозначениях для представления общих понятий и больше сконцентрироваться на проектировании и архитектуре.

UML обладает такими преимуществами:

* объектно-ориентированная структура, в результате которой методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках, что позволяет быстро переходить от создания диаграмм к созданию кода;
* возможность описать систему практически со всех возможных точек зрения и разные аспекты поведения системы;
* простота диаграмм для понимания;
* возможность расширения и ввода собственных текстовых и графических стереотипов, что способствует его применению не только в сфере программной инженерии;
* получил широкое распространение и динамично развивается.

В данной выпускной работе представлены основные диаграммы для проектирующейся программной системы из поведенческой группы, так как диаграммы именно этой группы являются наиболее важными. Одной из самых важных диаграмм является диаграмма прецендентов, диаграмма состояний и диаграмма развертывания.

Диаграмма прецедентов — это диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. В свою очередь прецедент — это возможность моделируемой системы, благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой [9].

Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится пояснение варианта использования в словесном виде в форме глагола. Цель прецендента – определить конечный аспект либо фрагмент поведения некоторой сущности без раскрытия ее внутренней структуры. В качестве такой сущности может выступать система или любой другой элемент модели, который обладает собственным поведением.

В UML существует несколько стандартных видов отношений между актерами и прецендентами:

* ассоциация - отражается линией между актером и вариантом использования (без стрелки). Отражает связь актера и варианта использования.;
* обобщение или наследование - показывает, что потомок наследует атрибуты и поведение своего прямого предка. Может применяться как для актеров, так для вариантов использования. Обозначается как стрелка с незакрашеным треугольником, указывающая на элемент, который будет наследован;
* включение - показывает, что вариант использования включается в базовую последовательность и выполняется всегда. Обозначается как пунктирная стрелка с надписью <<include>>;
* расширение показывает, что вариант использования расширяет базовую последовательность действий и вставляет собственную последовательность. При этом в отличие от типа отношений "включение" расширенная последовательность может осуществляться в зависимости от определенных условий. Обозначается как пунктирная стрелка с надписью <<extend>>, входящая в расширяемый прецедент, в дополнительном разделе которого может быть указана точка расширения и, возможно в виде комментария, условие расширения.

Для данной системы были описаны возможности как WEB-клиента, так и действия пользователя в рамках мобильного приложения. Use Case диаграмма представлена на рисунке 2.5.

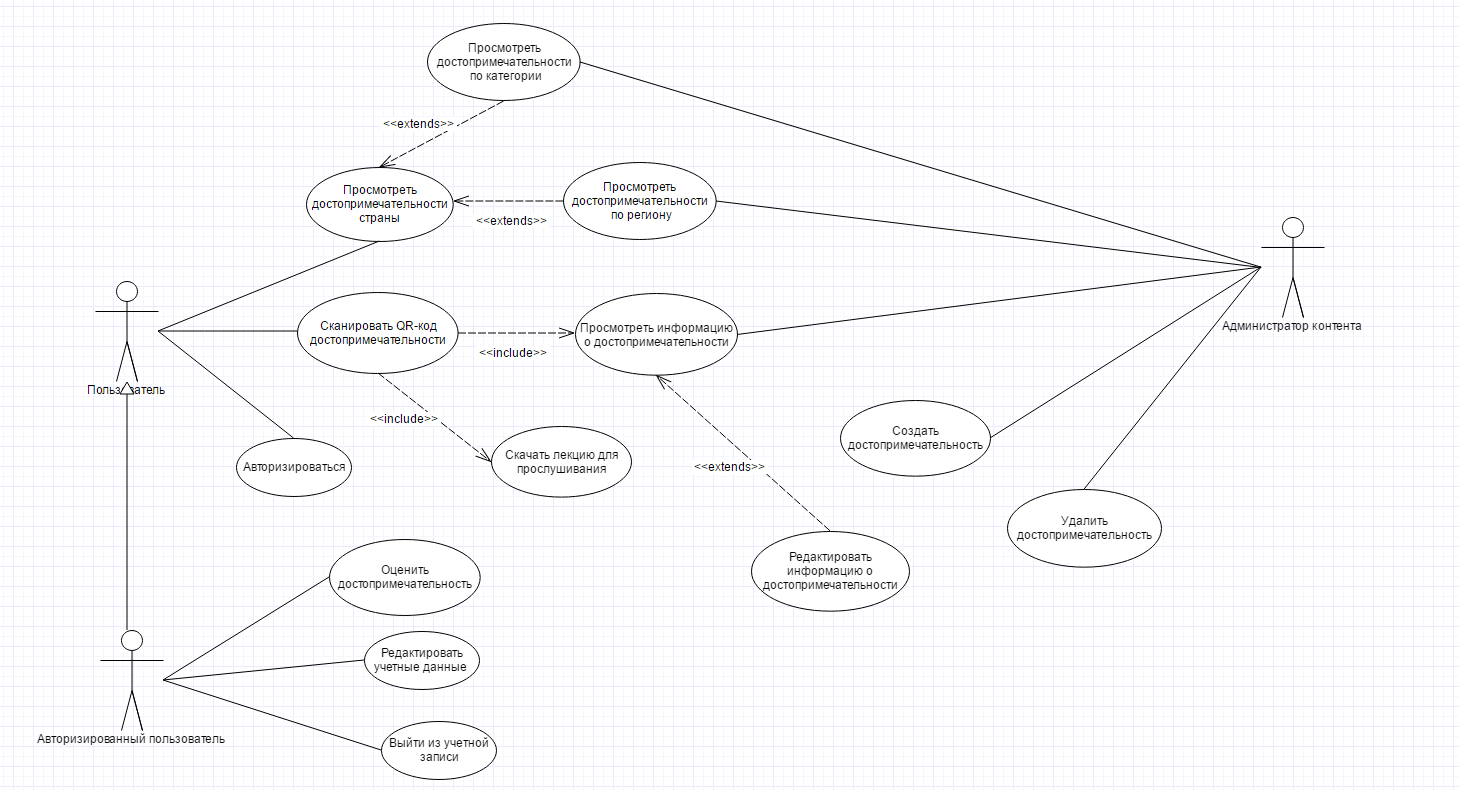


Рисунок 2.5 – Use Case диаграмма программной системы

В данной программной системе существует 3 актера: пользователь, зарегистрированный пользователь и администратор контента. Зарегистрированный пользователь наследует поведение обычного пользователя.

Для каждого пользователя существует большой набор возможных действий. Обычный пользователь может авторизироваться, сканировать QR-код места и скачать лекцию для прослушивания или просто посмотреть информацию о выбранной достопримечательности, просмотреть достопримечательности страны, просмотреть достопримечательности по региону или категории.

Зарегистрированный пользователь может оценить достопримечательность, редактировать учетные данные и выйти из учетной записи, а также все действия, которые доступны обычному пользователю.

Администратор контента может создать и удалить достопримечательность, просмотреть информацию о достопримечательности, просмотреть достопримечательность по странам, регионам и категориям.

Диаграммы состояний применяются для того, чтобы объяснить, каким образом работают сложные объекты. Состояние - ситуация в жизненном цикле объекта, во время которой он удовлетворяет некоторому условию, выполняет определенную деятельность или ожидает какого-то события. Состояние объекта определяется значениями некоторых его атрибутов и присутствием или отсутствием связей с другими объектами.

Перечень меток действия имеет фиксированные значения, которые не могут быть использованы в качестве имен событий. Эти значения следующие:

* entry - эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент входа в данное состояние (входное действие);
* exit - эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент выхода из данного состояния (выходное действие);
* do - эта метка специфицирует выполняющуюся деятельность («do activity»), которая выполняется в течение всего времени, пока объект находится в данном состоянии, или до тех пор, пока не закончится вычисление, специфицированное следующим за ней выражением действия. В этом случае при завершении события формируется соответствующий результат;
* include - эта метка используется для обращения к подавтомату, при этом следующее за ней выражение действия содержит имя этого подавтомата.

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий (псевдосостояние). В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания на диаграмме графической области, от которой начинается процесс изменения состояний. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка, из которого может только выходить стрелка, соответствующая переходу.

Конечное состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояние). В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени. Оно служит для указания на диаграмме графической области, в которой завершается процесс изменения состояний или жизненный цикл данного объекта. Графически конечное состояние в языке UML обозначается в виде закрашенного кружка, помещенного в окружность, которую может только входить стрелка, соответствующая переходу.

Простой переход представляет собой отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния объекта другим. Если пребывание моделируемого объекта в первом состоянии сопровождается выполнением некоторых действий, то переход во второе состояние будет возможен только после завершения этих действий и, возможно, после выполнения некоторых дополнительных условий, называемых сторожевыми условиями.

Для разрабатываемой программной системы диаграмма состояний имеет одну начальную и две конечные точки. На диаграмме состояний отображены состояния мобильного приложения, так как оно является ключевым для использования конечным пользователем. Работа программы будет начинаться с запуска, отображения главного экрана, на котором будет доступно две главных функции – сканирование QR-кода места или поиск достопримечательностей по стране. После сканирования будет доступна информация о месте, которую зарегистрированый пользователь сможет оценить. После поиска по стране можно произвести поиск по категориям либо регионам. При выборе достопримечательности будет предоставлена полная информация о ней. Диаграмма приведена на рисунке 2.6.

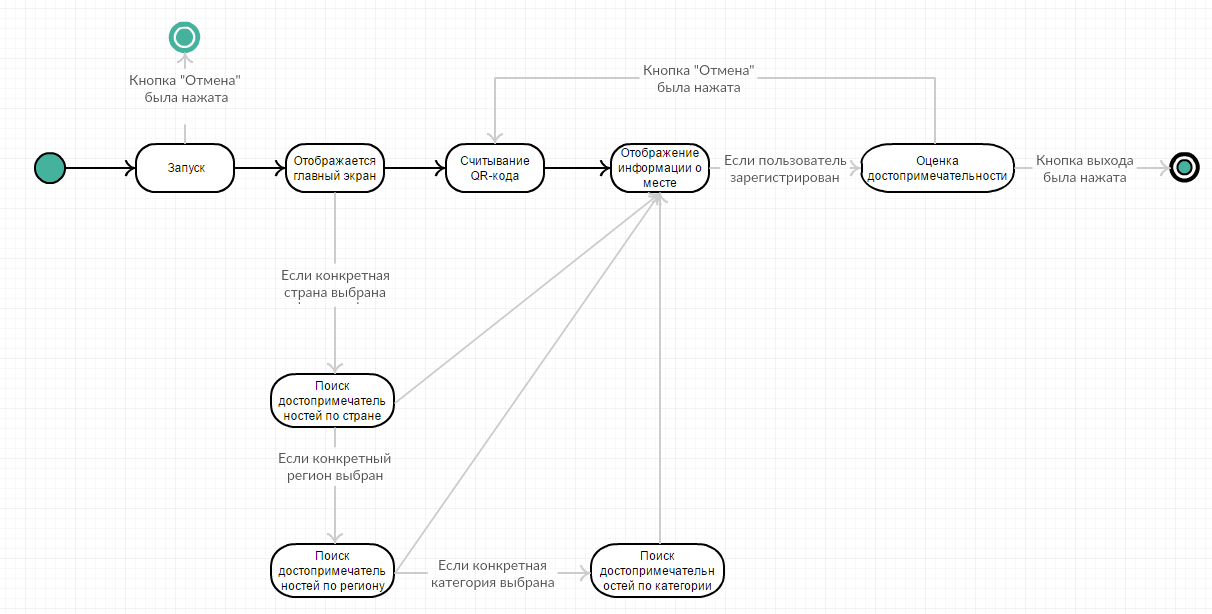


Рисунок 2.6 – Диаграмма состояний

Таким образом, при создании диаграммы состояний была смоделирована динамическая составляющая мобильного приложения.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. При этом представляются только компоненты-экземпляры программы, являющиеся исполняемыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются. Так, компоненты с исходными текстами программ могут присутствовать только на диаграмме компонентов. На диаграмме развертывания они не указываются [10].

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации. Разработка диаграммы развертывания, как правило, является последним этапом спецификации модели программной системы.

При разработке диаграммы развертывания преследуют следующие цели:

* определить распределение компонентов системы по ее физическим узлам;
* показать физические связи между всеми узлами реализации системы на этапе ее исполнения;
* выявить узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности.

В диаграмме присутствуют такие элементы:

* узел - представляет собой некоторый физически существующий элемент системы, обладающий определенным вычислительным ресурсом. Графически на диаграмме развертывания узел изображается в форме трехмерного куба. Узел имеет собственное имя, которое указывается внутри его графического символа. Сами узлы могут представляться как в качестве типов, так и в качестве экземпляров.
* соединения - физические соединения между узлами и зависимости между узлами и компонентами, изображения которых тоже могут присутствовать на диаграммах развертывания. Соединения являются разновидностью ассоциации и изображаются отрезками линий без стрелок. Наличие такой линии указывает на необходимость организации физического канала для обмена информацией между соответствующими узлами. Характер соединения может быть дополнительно специфицирован примечанием, помеченным значением или ограничением.

Диаграмма развертывания представляет собой схему, состоящую из символов узлов, соединенных маршрутами коммуникационных ассоциаций. Внутри символов узлов могут находиться экземпляры компонентов. Это указывает на то, что компонент существует или выполняется на данном узле. Символы компонентов, в свою очередь, могут содержать в себе символы объектов. Это обозначает, что объект является частью компонента. Компоненты связаны между собой пунктирными стрелками зависимостей (возможно, через интерфейсы). Такие связи обозначают, что один компонент использует услуги другого. Когда на диаграмме нужно точно указать вид зависимости, используют стереотип.

Диаграммы развертывания во многом напоминают диаграммы объектов. Как правило, на них изображаются отдельные экземпляры узлов, задействованных в системе. Куда реже встречаются диаграммы развертывания, на которых показаны существующие типы узлов и их отношения с другими типами узлов, как это делается на диаграмме классов.

При разработке диаграммы развертывания было создано три узла: WEB-сервер, сервер баз данных и клиент. Артифактом WEB-сервера является сайт, сервера баз данных – база данных, клиента сайта – браузер, артифактом которого есть HTML5. Между WEB-сервером и клиентом обозначена связь по протоколу http на порту 80. WEB-сервер и сервер баз данных соединяются без какого-либа протокола. На рисунке 2.7 представлена диаграмма развертывания для разрабатываемой программной системы.

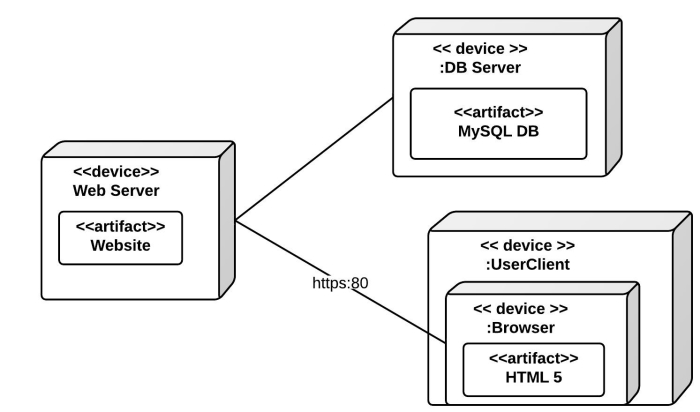


Рисунок 2.7 – Диаграмма развертывания приложения

Таким образом, после этапа UML-моделирования было создано две диаграммы поведения: диаграмма прецендентов и диаграмма состояний и одна структурная диаграмма в виде диаграммы развертывания.

* 1. Проектирование базы данных

Так как программная система служит, в основном, для получения информации, то надежно спроектированная база данных играет ключевую роль в разработке. Для данной работы нужна максимально простая схема, так как вся нагрузка будет приходиться именно на нее. При неправильном проектировании схемы время отклика будет увеличиваться, а удобство ее использования – снижаться.

Для данной работы была выбрана реляционная база данных, так как эта модель хранения данных является наиболее простой для проектирования, а также существует возможность ее изменения.

Реляционная база данных — это совокупность взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа. Строка таблицы содержит данные об одном объекте, а столбцы таблицы описывают различные характеристики этих объектов — атрибутов. Записи имеют одинаковую структуру — они состоят из полей, хранящих атрибуты объекта. Каждое поле описывает только одну характеристику объекта и имеет строго определенный тип данных. Все записи имеют одни и те же поля, только в них отображаются различные информационные свойства объекта.

В реляционной базе данных каждая таблица должна иметь первичный ключ — поле или комбинацию полей, которые единственным образом идентифицируют каждую строку таблицы. Если ключ состоит из нескольких полей, он называется составным. Ключ должен быть уникальным и однозначно определять запись. По значению ключа можно отыскать единственную запись. Ключи служат также для упорядочивания информации в базе данных.

Таблицы реляционной БД должны отвечать требованиям нормализации отношений. Нормализация отношений — это формальный аппарат ограничений на формирование таблиц, который позволяет устранить дублирование, обеспечивает непротиворечивость хранимых в базе данных, уменьшает трудозатраты на ведение базы данных.

Разрабатываемая база данных была нормализирована до третьей нормальной формы, которая определяется следующими свойствами:

* в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов;
* каждый атрибут, который не принадлежит ни одному из потенциальных ключей атрибут неприводимо зависит от её потенциального ключа;
* ни один неключевой атрибут не находится в транзитивной функциональной зависимости от потенциального ключа.

Дальнейшей нормализации эта база данных не требует, так как третьей нормальной формы достаточно для нормальной работы системы, а также для быстрого и простого изменения ее схемы.

Основными сущностями являются «Роль», «Пользователь», «Язык», «Контент», «Страна», «Регион», «Категория», «Код» и «Достопримечательность». В ходе концептуального моделирования были определены их атрибуты и связи.

Сущность «Роль» создана для хранения ролей пользователей и поддержания ролевого разграничения доступности контента WEB-сайта. Таким образом, обычный пользователь не сможет редактировать информацию на сервере в то время, как у администратора будут на это определенные права. Сущность содержит следующие атрибуты:

* уникальный идентификатор;
* имя.

Сущность «Язык» создана для хранения данных о родном языке пользователя для локализации контента сайта и мобильного приложения. Сущность содержит следующие атрибуты:

* уникальный идентификатор;
* имя.

Сущность «Пользователь» была создана для хранения информации о пользователе и настройках его учетной записи. Эта сущность имеет отношение «один ко многим» с таблицей «Язык», поэтому принимает внешним ключом ее уникальный идентификатор. Также она имеет отношение «один ко многим» с таблицей «Роль», также принимая ее идентификационный номер в качестве внешнего ключа. Сущность содержит такие атрибуты:

* уникальный идентификатор;
* имя;
* уникальный идентификатор языка (внешний ключ на таблицу «Язык»);
* уникальный идентификатор роли (внешний ключ на таблицу «Роль»);
* логин;
* пароль;
* фотография.

Сущность «Контент» спроектирована для хранения единиц записей экскурсий, доступных для скачивания пользователям. Сущность содержит следующие атрибуты:

* уникальный идентификатор;
* ссылка на файл;
* имя.

Сущность «Страна» создана для того, чтобы хранить в себе регионы, по которым могут быть найдены достопримечательности. Эта сущность также имеет отношение «один ко многим» с таблицей «Язык», поэтому принимает внешним ключом ее уникальный идентификатор. Сущность содержит следующие атрибуты:

* уникальный идентификатор;
* уникальный идентификатор языка (внешний ключ на таблицу «Язык»);
* имя.

Сущность «Регион» создана для того, чтобы хранить в себе достопримечательности. Она имеет отношение «один ко многим» с сущностью «Страна», поэтому содержит уникальный идентификатор в качестве внешнего ключа. Сущность содержит следующие атрибуты:

* уникальный идентификатор;
* уникальный идентификатор страны (внешний ключ на таблицу «Страна»);
* имя.

Сущность «Категория» была спроектирована для того, чтобы хранить в себе категории достопримечательностей, по которым их можно будет найти. Сущность содержит следующие атрибуты:

* уникальный идентификатор;
* имя.

Сущность «Достопримечательность» создана для того, чтобы хранить в себе информацию о достопримечательности. Она имеет отношения «один ко многим» с таблицами «Категория» и «Регион», поэтому содержит внешние ключи на эти таблицы. Сущность содержит следующие атрибуты:

* уникальный идентификатор;
* уникальный идентификатор категории (внешний ключ на таблицу «Категория»);
* уникальный идентификатор региона (внешний ключ на таблицу «Региона»);
* информацию;
* широту;
* долготу;
* фотографию;
* имя.

Из-за того, что таблицы «Контент» и «Достопримечательность» находятся в отношении «многие ко многим» образуется новая таблица «Контент достопримечательности», которая разбивает это отношение на два: «один ко многим» и «многие к одному». Эта таблица содержит два внешних ключа и уникальный идентификатор.

Сущность «Код» была спроектирована для того, чтобы хранить в себе QR-коды для каждой достопримечательности для того, чтобы при его считывании можно было получить ссылку на определенный контент для скачивания. Сущность содержит следующие атрибуты:

* уникальный идентификатор;
* картинку QR-кода;
* - уникальный идентификатор достопримечательности (внешний ключ на таблицу «Достопримечательность»).

Проанализировав предметную область, была создана ER-модель данных, позволяющая описывать концептуальные схемы предметной области. Во время проектирования баз данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели данных. В результате получилась схема базы данных, которая состоит из десяти таблиц. На рисунке 2.8 представлена схема базы данных.

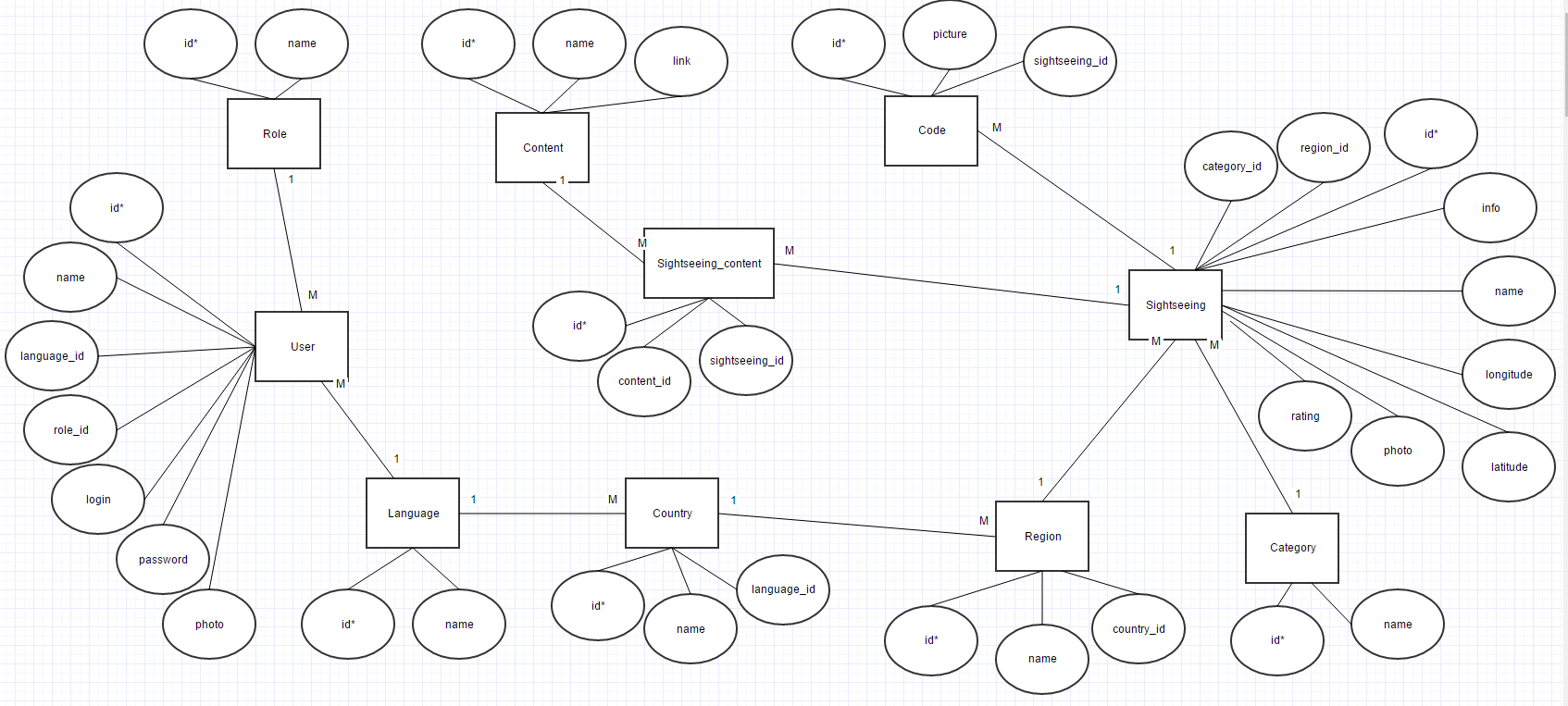


Рисунок 2.8 – Схема базы данных

В качестве средства управления базы данных был выбран MySQL, так как он прост в использовании, имеет открытый код и является надежным средством поддержки и администрирования базы данных.

ВЫВОДЫ

Я, Войтенко Мария Валерьевна, проходила преддипломную практику на базе компании «ЕПАМ Системз» в период с 18 апреля 2016 года по 14 мая 2016 года. Во время прохождения практики я получила опыт работы на производстве, получила новые знания и закрепила их на при выполнении индивидуального задания, научилась применять теоретические знания во время проектирования, разработки и тестирования программного обеспечения на практике. Во время выполнения индивидуального задания, выданого на период преддипломной практики, я закрепила и увеличила количество изученного материала по технологиям Java, Spring, AngularJS, Twitter Bootstrap, HTML, CSS, JavaScript, Spring Boot, Spring Security, Liquibase, JPA, Spring MVC REST и Android.

В результате выполнения индивидуального задания была проанализирована предметная область и спроектирована архитектура WEB-сервера, WEB-клиента и мобильного клиента. Был разработан сервер и клиент для него, а также мобильное приложение.

Результаты выполненной работы полностью удовлетворяют поставленные требования. В ходе разработки была разработана система для проведения екскурсий, которая позволяет считывать QR-код, находить по нему информацию достопримечательности, а также находить достопримечательности в разных странах и регионах. Также пользователи приложения имеют возможность просматривать достопримечательности по категориям и оценивать их для предоставления другим пользователям рейтинга выбранного места.

В ходе разработки индивидуального задания были использованы архитектурные шаблоны, практики написания «чистого кода» и лучшие практики разработки на всех этапах построения приложения.

Мобильная прикладная программа для проведения экскурсий является актуальной в всязи с распространением туристического направления по всему миру. Это приложение позволяет сэкономить средства, которые могли быть потраченными на гидов и платные туры по достопримечательностям разных стран, а также значительно облегчает выбор маршрутов и планирование поездки.

ПЕРЕЧЕНЬ ССЫЛОК

1. Введение в туризм: Учебное пособие [Текст]/ Биржаков М. Б. – СПб: Издательский дом Герда, 2004 г. – стр. 37 – ISBN 5-94125-082-7.
2. Why tourism? [Электронный ресурс]/ World Tourism Organization UNWTO – Режим доступа: http://www2.unwto.org/content/why-tourism – 25.11.2015 – Загол. с экрана.
3. The Global Wellness Tourism Economy 2013 & 2014 [Электронный ресурс]/ Global Wellness Institute – Режим доступа: http://www.globalwellnesssummit.com/images/stories/gwi/GWI\_2014\_Global\_Wellness\_Tourism\_Economy\_Report\_Final.pdf – 02.06.2014 – Загол. с экрана.
4. Number of apps available in leading app stores as of July 2015 [Электронный ресурс]/ Statista – Режим доступа: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores – 17.05.2015 – Загол. с экрана.
5. Трёхуровневая архитектура [Электронный ресурс]/ Википедия – Режим доступа: https://ru.wikipedia.org/wiki/Трёхуровневая\_архитектура – 17.01.2015 – Загол. с экрана.
6. Model-View-Controller [Электронный ресурс]/ Википедия – Режим доступа: https://ru.wikipedia.org/wiki/Model-View-Controller – 07.08.2015 – Загол. с экрана.
7. Паттерны для новичков: MVC vs MVP vs MVVM [Электронный ресурс]/ Хабрахабр – Режим доступа: http://habrahabr.ru/post/215605/ – 14.03.2014 – Загол. с экрана.
8. UML [Электронный ресурс]/ Википедия – Режим доступа: https://ru.wikipedia.org/wiki/UML – Загол. с экрана.
9. Диаграмма прецендентов [Электронный ресурс]/ Википедия – Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма\_прецендентов – Загол. с экрана.
10. Deployment Diagram [Электронный ресурс]/ Diagrama – Режим доступа: http://www.diagrama.xyz/deployment-diagram/ – Загол. с экрана.

ПРИЛОЖЕНИЕ А

Описание предприятия

Компания «ЕПАМ Системз» является первой по рейтингу компанией в Украине по трудоустройству программистов. Компания EPAM была основана в 1993 году в Принстоне двумя одноклассниками Аркадием Добкиным и Леонидом Лознером. Название компании происходило от «Effective Programming for America». Первые офисы были открыты в США и Беларуси. Позже были открыты центральный североамериканский офис в Лоренсвилле, США, штат Нью-Джерси и центральный европейский офис в Будапеште, Венгрия, а также офисы по обслуживанию клиентов в Великобритании, Германии, России, Казахстане, Украине, Польше, Швеции, Швейцарии, Болгарии и Беларуси.

На данный момент занимается следующиими направлениями:

* ИТ-консалтинг;
* разработка программного обеспечения;
* интеграция приложений;
* портирование и миграция приложений;
* тестирование программного обеспечения;
* создание выделенных центров разработки на базе EPAM Systems;
* разработка цифровых стратегий.

Число сотрудников «ЕПАМ Системз» в Харькове достигло 1000 человек в конце 2015 года, что говорит о том, что эта компания является прогрессивной и быстро растущей на современном рынке IT.

Сотрудники делятся на различные отделы, каждый из которых занимается проектами по определенной технологии. Это Java и .Net. Однако, существует большое количество как мануальных, так и автоматизированных тестировщиков, специалистов по фронтенду, системных администраторов и девопсов.

«ЕПАМ Системз» является одной из крупнейших IT-компаний мира, которая разрабатывает качественное программное обеспечение и предоставляет услуги высококвалифицированных сотрудников.