МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка мобильного приложения «Онлайн гид» с использованием технологий дополненной реальности

Студент А.Е. Трухина
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н., доцент, Э.В. Егорова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Аннотация

Тема бакалаврской работы: «Разработка мобильного приложения «Онлайн гид» с использованием технологий дополненной реальности».

Выпускная квалификационная работа освещает проблему недостаточной просвящённости населения в области знания культурной и исторической ценностей собственного города. Предлагаемым решением является разработка мультимедийного приложения с объектами дополненной реальности, позволяющими знакомиться с объектами города, не выходя из дома.

Целью данной работы является разработка мобильного приложения «Онлайн Гид» для изучения городских достопримечательностей населением и гостями города.

Объектом бакалаврской работы является поиск оптимального способа взаимодействия клиентской и серверной частей, ориентированного на передачу сущностей, содержащих различные типы данных.

Предметом исследования в данной работе является разработка сервера, который будет способен не только обрабатывать запросы пользователей, но и будет обладать актуальной архитектурой схожей с большими и популярными в настоящее время проектами. В первом разделе описывается процесс проектирования информационной системы и формулировка выполняемых ею функций. Во втором разделе описываются выбранные для реализации спроектированной системы инструменты, а также математическая модель для модуля оптимизации маршрутов. Третий раздел содержит описание процесса тестирования и его результаты.

Эта бакалаврская работа состоит из пояснительной записки на 60 страниц, введения в две страницы, включая 14 рисунков, список из 25 используемых источников, включая 6 источников на иностранном языке и 1 приложение.

Бакалаврская работа разрабатывалась по заказу компании «EPAM».

Abstract

This graduation work is devoted to the problem related to insufficient enlightenment of the population in the field of knowledge of the cultural and historical phenomena of their own city or town.

The proposed solution to this problem is the development of a multimedia application that includes integration with the popular augmented reality technology that allows people to get acquainted with the objects of the city without leaving their home.

The graduation work consists of an explanatory note, an introduction, including 14 figures, a list of 25 references, including 6 foreign sources and 1 appendix.

The purpose of the research is to design, develop and test the *Online Guide* information system.

The object of the investigation is the search for the effective way of interaction between the client and server parts aimed at transferring entities containing not only various types of data, but also data of large volume, including 3D objects.

The subject of the research is the development of a server that will not only be able to process users' requests, but will also be relevant in terms of executable functions among servers designed for large and popular projects at present.

The main part of the graduation work begins with the analysis of the subject area. The functional requirements for the system are formulated and its operation in various cases is described on the basis of the data obtained. Then, the program development stage begins. At this stage, the available tools and environments are analyzed and selected. After that, the architecture of the application and the ways of interacting the client and the server are determined, as well as the system is developed. At the end of the work on the product, the developed features and capabilities are tested.

Adaptation of the product for other cities can be considered as a further development of the project in question.

Содержание

Введение

Проблема непросвящённости населения в области знаний истории собственного города является как никогда актуальной именно в 21 веке. С появлением смартфонов и их дальнейшим усовершенствованием, всё внимание всех поколений устремилось в сторону этих устройств. Несмотря на общедоступность и лёгкость в поиске полезной информации, а также большой вычислительной мощности мобильных устройств, относительно своих предшественников, большая часть общества предпочитает коротать время в развлекательных приложениях.

Также в последние годы среди туристов набирает популярность самостоятельные путешествия. Для этого им требуется заранее собирать информацию и адреса интересующих культурных объектов и самостоятельно составлять маршрут следования. Вдобавок к этому им может потребоваться информация о билетах: их доступность, время сеанса, цена посещения.

Данная выпускная квалификационная работа предлагает решение вышеупомянутых проблем в проектировании и разработке мультимедийного приложения, совмещающего в себе полезную историческую и фактическую информацию об объеках города, а также объекты набирающей популярность дополненной реальности.

Объектом бакалаврской работы является поиск оптимального способа взаимодействия клиентской и серверной частей, ориентированного на передачу сущностей, содержащих не только различные типы данных, но и данные больших объёмов, включая 3D – объекты.

Предметом исследования в данной работе является разработка сервера, который будет способен не только обрабатывать запросы пользователей, но и будет актуальным по исполняемым функциям среди серверов для больших и популярных проектов в настоящее время.

Целью бакалаврской работы является разработка приложения «Онлайн Гид», которое поспособствует привлечению внимания общественности к истории собственного города, а так же может помочь гостям города изучить его самостоятельно.

Для достижения поставленной цели были установлены следующие задачи для исследования:

- изучить предметную область;
- описать требования к информационной системе;
- проанализировать актуальные архитектурные решения;
- изучить популярные технологии, используемые в больших проектах;
- реализовать серверную часть приложения;
- провести оценку качества разработанного приложения.

Бакалаврская работа содержит в себе 3 раздела. В первой описывается процесс проектирования информационной системы и формулируется описание её конечного вида и выполняемых ею функций. Во второй описываются выбранные для реализации спроектированной системы инструменты, а также математическая модель для модуля оптимизации маршрутов. Третий раздел содержит в себе сформированный план тестирования и их результаты.

Эта выпускная квалификационная работа состоит из пояснительной записки на 60 страниц, введения в две страницы, включая 14 рисунков, список из 25 используемых источников, включая 6 источников на иностранном языке и 1 приложение.

1 Анализ туристической области и рынка её программных продуктов

1.1 Разработка описания информационной системы «Онлайн Гид»

В условиях современного мира привлечь внимание горожан к историческим и культурным ценностям города становится всё сложнее и сложнее.

Обычные экскурсии или бумажные энциклопедии заменяются сухими статьями из интернета, оставляя у горожан неполное или недостоверное мнение о достопримечательностях и истории города.

В век цифровых технологий, когда дети буквально с пелёнок держат в руках смартфон или планшет, существует опасность просто-напросто забыть свои истоки.

Помимо этого, в последние годы стремительно увеличивается процент туристов, которые путешествуют, не пользуясь услугами экскурсоводов или туристических фирм. Это как никогда актуально для нашего города, ведь в настоящее время ведётся строительство и возведение нескольких культурных объектов, которые могут заинтересовать путешественников.

Во избежание банальной безграмотности населения в области знания родного города и его истории, а также для преумножения потока туристов, способствующего привлечению средств в город и дальнейшего его развития в области туризма, решено разработать приложение-гид, которое будет подавать актуальную информацию о городе в медиа формате.

Помимо обычных текстовых статей, пользователю будут предложены видеоролики об искомых объектах, а также 3D модели, реализованные с помощью технологий дополненной реальности, что предоставит пользователю возможность ознакомиться с реальным объектом прямо с экрана телефона.

В настоящее время технологии дополненной реальности находят всё больше и больше областей применения в жизнедеятельности человека: в медицине, образовании, безопасности и пр.

Технологии дополненной реальности позволяют человеку видеть изменённое окружение, созданное путём наложения на реальное изображение различных виртуальных объектов [15].

В основе технологий дополненной реальности лежит взаимодействие между сигналом с камеры и математических алгоритмов, которые в свою очередь определяют в пространстве маркеры, к которым будет осуществляться дальнейшая привязка 3D моделей.

На данный момент все известные технологии дополненной реальности можно разделить на 3 типа:

– безмаркерная технология: распознавание маркеров происходит путём покрытия поступающего на компьютер видеопотока так называемой «сеткой», алгоритмы просчитывают по этой сетки расположение точек, по которым будет производиться привязка модели для конечного вывода. Отсюда и название, т.к. точками привязки модели служат реальные объекты;

– технология на базе маркеров: для работы требуется наличие заданного маркера во входящем видеопотоке, алгоритм распознаёт только этот маркер и производит привязку модели строго только к этой точке. В некоторой степени работает стабильнее, чем «безмаркерная» технология, но при отсутствии маркера на изображении – не будет и 3D модели;

– пространственная технология: для привязки модели использует данные о геоположении (GPS или ГЛОНАСС), а так же направлении относительно сторон света (встроенный гироскоп или компас). Алгоритм привязывает 3D модель к определённым в программе координатам, а также определяет диапазон, в котором пользователю эта модель будет доступна для просмотра.

Вследствие того, что гид должен сопровождать пользователя повсеместно, вариант с использованием маркеров сразу же исключается, т.к. пометить каждый объект в городе не представляется возможным, а также

пропадает возможность просмотра 3D модели без доступа к этому объекту в реальном мире. Помимо этого, пользователь может столкнуться с проблемой недостаточного освещения, которое требуется для распознавания маркеров, определения их местоположения в пространстве и дальнейшей привязке к ним модели.

Подобным образом исключается и использование пространственной технологии – пропадает доступ к просмотру модели вне пешей доступности к реальному объекту.

Поскольку вычислительная способность смартфонов с каждым годом растёт, у большинства носителей мобильных устройств появляется возможность открыть для себя мир дополненной реальности: от детских сказок с интерактивными героями до программ, с помощью которых можно подобрать мебель в комнату путём размещения объектов прямо на экране.

Использование дополненной реальности в приложении "Онлайн Гид" позволит пользователю манипулировать объектом, как ему угодно: приблизить и рассмотреть детали, вращать его по вертикали и горизонтали. Для культурных объектов, представляющих собой какие-либо здания, например, музеи, будет иметься возможность войти в эмулированную комнату, где будут располагаться некоторые модели картин или скульптур.

Технологии дополненной реальности позволят разрабатываемому приложению выделиться среди других туристических продуктов.

Приложение будет помогать пользователю в формировании собственного маршрута посещений объектов города. Технологии искусственного интеллекта, позволят приложению не только рассчитать время, за которое вы можете добраться до пункта назначения, но и предложит такие маршруты, чтобы пользователю предоставлялся выбор из самого быстрого способа просмотра объектов, самого дешёвого просмотра или, например, способа, который учитывает время работы объектов, чтобы успеть просмотреть весь маршрут.

Единственным конкурентом на рынке туристического ПО, который схож по функциональному наполнению с «Онлайн гидом», является чешский продукт «Sygic Travel».

«Sygic Travel» является одним из серии продуктов компании «Sygic». Также компания выпускает ряд приложений для путешествий на машине и грузовом автомобиле, т.к. в Европе большинство трасс не предназначены для грузовых автомобилей, для них существуют отдельные платные дороги. Для самостоятельных путешествий на легковом автомобиле компания предлагает GPS-навигатор, навигатор для камер, фиксирующих скорость, а также приложение, отслеживающее пробег, расход топлива и траты на него.

«Sygic Travel» предоставляет пользователю широкий выбор стран для построения маршрута, а так же просмотра самых популярных мест для посещения в населённом пункте. Есть возможность фильтровать отображение объектов разных типов. Помимо вышеперечисленных возможностей, «Sygic Travel» предлагает пользователю список экскурсий, доступных в текущем населённом пункте. К каждой из них прилагается небольшая статья с перечислением тех мест, которые вы посетите, информация о стоимости посещения в рублях на одного человека, где располагается точка сбора, какие языки знает гид, отзывы и оценки других туристов, а так же прочая информация об экскурсии. Приложение так же помогает пользователю в составлении своих собственных маршрутов с учётом использования общественного транспорта, но при этом оно не содержит никакой информации об объектах кроме нескольких фотографий.

1.2 Разработка диаграммы точек зрения

Разрабатываемое мобильное приложение предполагает эксплуатацию одного вида – для пользователя.

При открытии приложения пользователь попадает на карту города, где отмечены все доступные к просмотру места. Различные иконки позволяют пользователю понять, какой объект он собирается просматривать. Также пользователь может сортировать отображение объектов по типу, например, только памятники или музеи. Далее пользователь выбирает любой интересующий его объект на карте и попадает на страницу с информацией о нём. На данной странице пользователю предоставляются статьи и видеоролики об объекте, а также предлагается просмотреть его виртуальную модель и добавить данный объект в маршрут. В меню маршрутов можно просмотреть различные виды предлагаемых маршрутов.

Рисунок 1.1 – Диаграмма точек зрения для приложения «Онлайн Гид»

Диаграмма идентификации точек зрения позволяет наглядно определить, какие функциональные требования к приложению будут выдвинуты.

1.3 Разработка модели бизнес-процессов в методологии IDEF0

Чтобы представить обозначенные требования в виде последовательности процессов, строится IDEF0 диаграмма. Сперва определяется общая функция, выполняемая приложением [2].

Какую роль выполняют люди, работающие гидами? В обязанности гида входят сопровождение туристической группы по городу, рассказ исторических фактов, а так же каких-либо запоминающихся событий, связанных с этим объектом. Так как разрабатываемое приложение будет служить заменой человеку, то оно должно так же вести пользователя, который выступает в качестве туриста, по городу, рассказывая об интересующих объектах.

Таким образом, основной функцией является навигация пользователя по городу и предоставление сведений о культурных и исторических объектах города.

Рисунок 1.2 – Контекстная IDEF0 диаграмма для приложения «Онлайн Гид»

Для более детального описания последовательности предоставляемых пользователю функций, производится декомпозиция диаграммы.

Пользователь начинает работу с приложением с выбора одного из всех представленных на карте маркированных объектов. После этого он попадёт на информационную страницу об этом объекте. Здесь он может посмотреть, как добраться до него от текущего положения или от заданной на карте точки. Также на данной странице представлены информационные статьи и

видеоролики, а также просмотр 3D-модели выбранного объекта в дополненной реальности, что потребует разрешения для использования камеры мобильного устройства.

Помимо вышеперечисленного, на текущей странице пользователь сможет добавить выбранный объект в свой маршрут или, в случае отсутствия маршрута, создать новый, где нынешний объект будет первым в списке посещения.

По окончанию работы с выбором объектов для маршрута, пользователь возвращается на карту города. Теперь он может войти в меню маршрутов, которое позволит ему оптимизировать его план посещения. Здесь представлены несколько настроек оптимизации:

– обновление маршрута для посещения кратчайшим путём;

– обновление маршрута для посещения с учётом времени работы объектов;

– обновление маршрута для посещения самым дешёвым путём (при необходимости покупки билетов) и пр.

Завершая работу с оптимизацией, если она необходима, пользователь получает свой собственный маршрут, по которому теперь можно следовать.

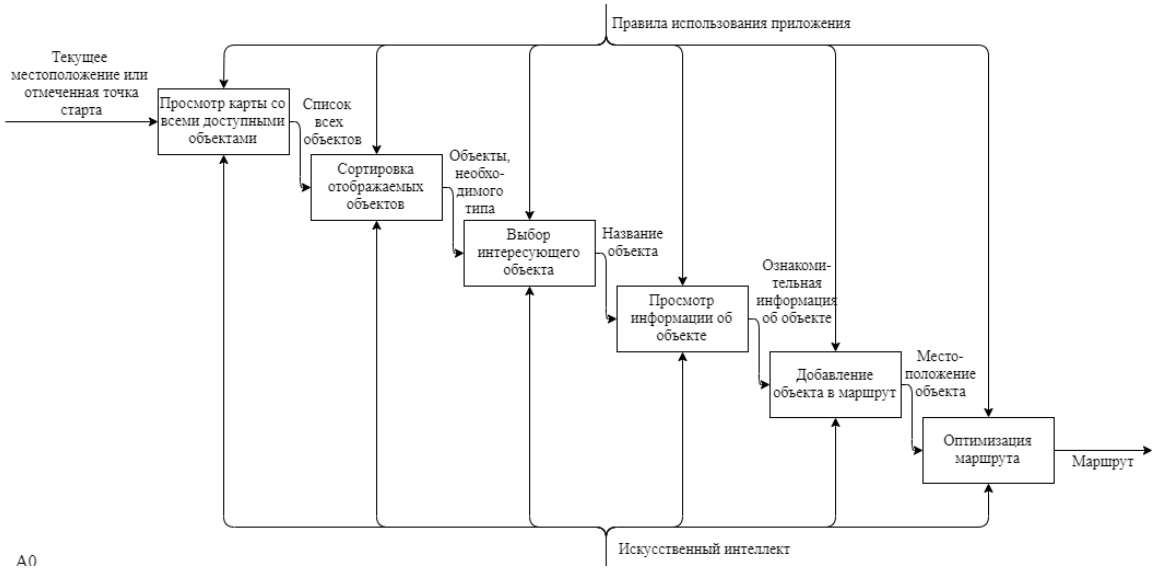


Рисунок 1.3 – Декомпозиция контекстной IDEF0 диаграммы для приложения «Онлайн Гид»

Далее определим, какие данные необходимо хранить для работы приложения.

1.4 Моделирование базы данных приложения «Онлайн гид»

В первую очередь «Онлайн Гид» требует, конечно же, данных, описывающих маркеры на карте [7].

Так же следует выделить в отдельную сущность все типы объектов, представленных на карте. Это поспособствует более удобному, а главное эффективному поиску и сортировке объектов на карте.

Подобным образом описывается и сущность пользователя, которая представляет собой учётную запись каждого человека, необходимую для работы с приложением.

Одним из функциональных требований является построение пользовательских маршрутов. Таким образом, в завершение работы с моделью базы данных, добавим ещё одну сущность, связывающую пользователя и объекты, - маршрут.

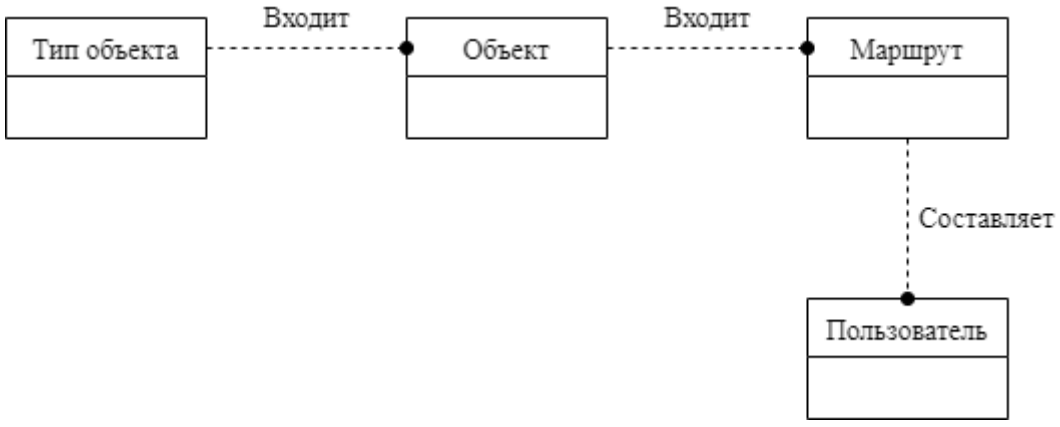


Рисунок 1.4 – Диаграмма сущность-связь IDEF1X нотации логического уровня

Физическая модель раскрывает свойства проектируемой базы данных для выбранной СУБД. Атрибуты сущностей перечисляются, учитывая особенности синтаксиса СУБД.

Определим, какие данные должны помещаться в сущности, изображённые на рисунке 1.5 [8].

Во-первых, тип объекта. Для его описания потребуются лишь поле для первичного ключа и название типа. Так же добавим атрибут для описания данного типа объекта, чтобы пользователь мог ознакомиться с небольшой сноской о каждом типе при фильтрации отображаемых объектов на карте.

Что необходимо для описания сущности любого из культурно значимых мест? Первичный ключ, название, тип, небольшое описание объекта, координатное расположение, физический адрес, а так же вся медиа информация об объекте и его и его 3D модель.

Любой из объектов может быть помещён в маршрут. В описание его сущности так же входит уникальный первичный ключ, название маршрута, описание, которое можно оставить пустым, и, наконец, сама последовательность мест.

Наконец, пользователь описывается личными данными, такими, как фамилия, имя, отчество и пр., а так же массив маршрутов, который может принимать значение NULL.

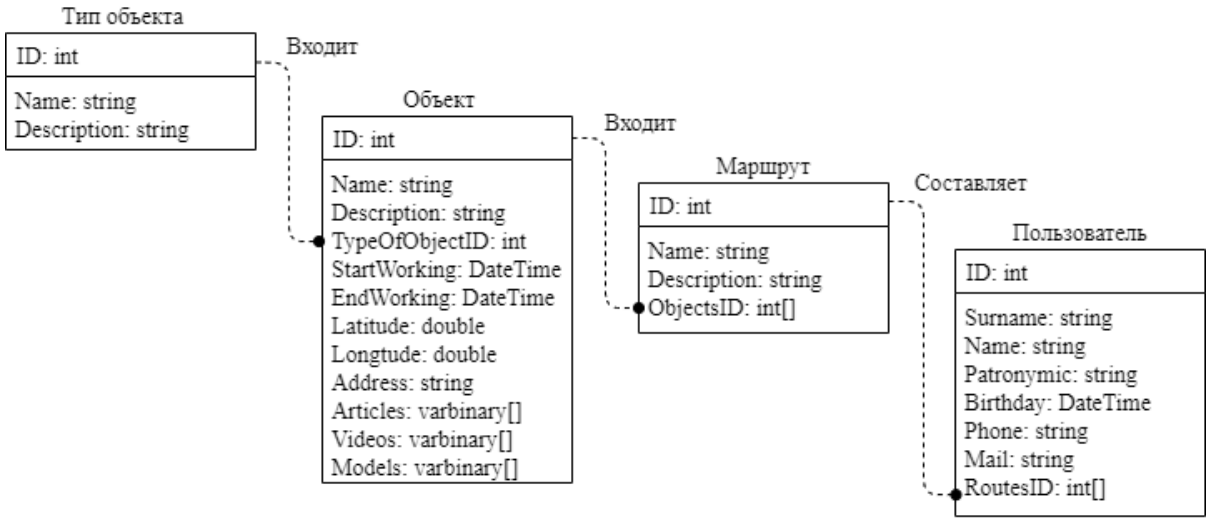


Рисунок 1.5 – Модель IDEF1X нотации физического уровня

Теперь определим, каким образом будут циркулировать потоки данных в приложении с помощью построения диаграммы DFD.

1.5 Моделирование движения потоков данных приложения в стандарте DFD

Сначала формулируется контекстная DFD-диаграмма, которая демонстрирует основную функцию, выполняемую системой, но с учётом хранилищ, которые были описаны выше [10].

Первостепенная задача приложения – сопровождение и ориентирование пользователя по городу и предоставление сведений об объектах города, имеющих туристическую и историческую ценность.

Для начала работы с приложением, пользователю необходимо определить исходную точку для старта прокладывания маршрута. Это можно сделать, отметив произвольную точку на карте, либо выбрать один из предложенных маркеров на карте.

Рисунок 1.6 – Контекстная DFD-диаграмма для приложения «Онлайн Гид»

Далее пользователь самостоятельно выбирает все интересующие его местоположения и добавляет в маршрут. По завершению выбора, можно оптимизировать маршрут по разным параметрам.

По окончанию работы с нужным маршрутом, пользователь получает готовый маршрут, который сопроводит его по городу, а так же содержит всю доступную информацию о выбранных объектах.

Проведём детализацию, раскрывая основную функцию приложения в подсистемы, которые и отвечают за каждый бизнес-процесс. Помимо этого также обозначим внешние сущности, взаимодействующие с системой - пользователя и модуль искусственного интеллекта, и хранилища данных [18].

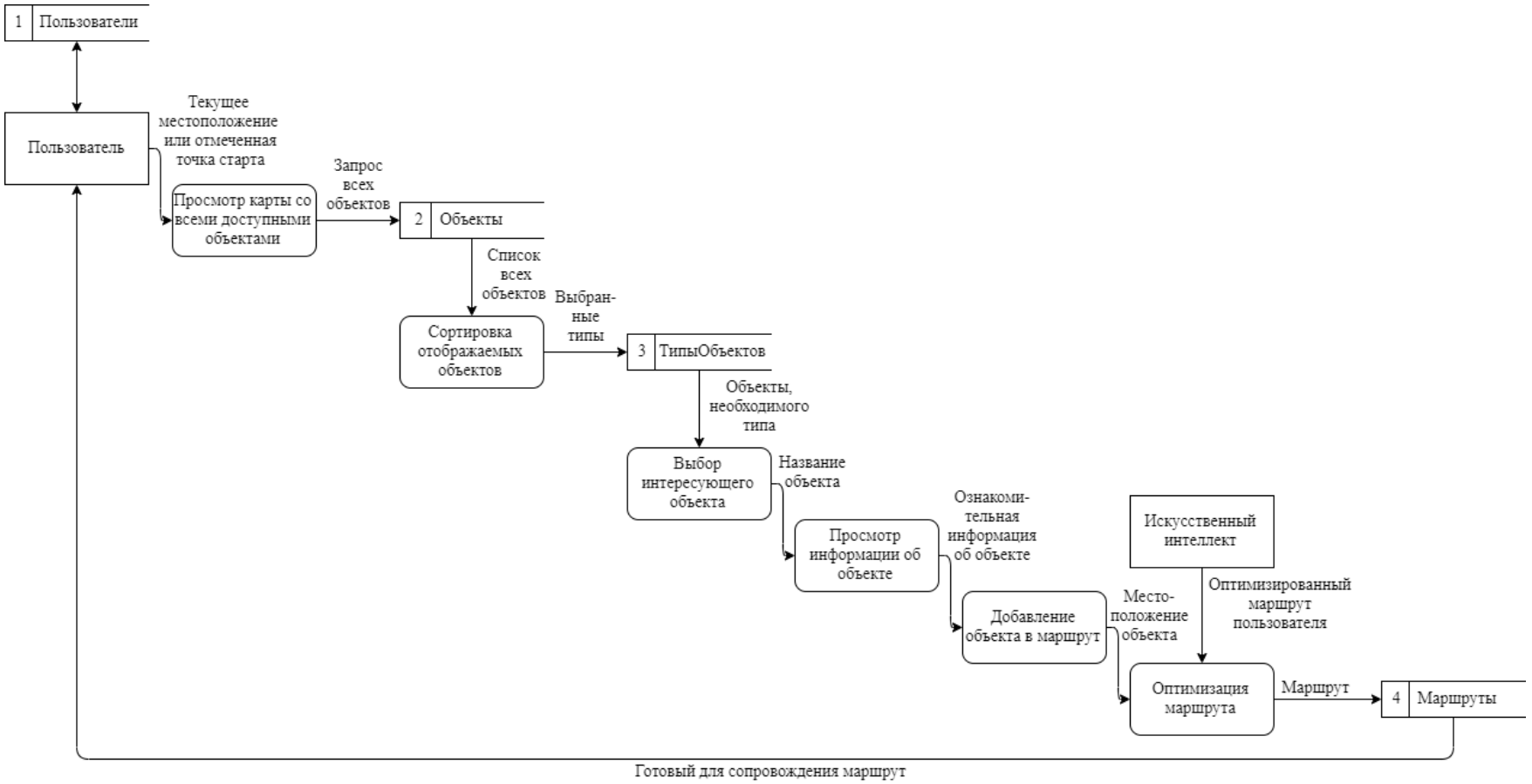


Рисунок 1.7 – Декомпозиция 1-го уровня DFD-диаграммы для приложения «Онлайн Гид»

Декомпозиция контекстной DFD-диаграммы предоставляет нам модель, демонстрирующую входные и выходные потоки данных, которые циркулируют в приложении.

1.6 Описание последовательности выполняемых функций информационной системы «Онлайн Гид»

В завершении описания работы приложения, опишем последовательность действий пользователя и ответов приложения на каждое из них.

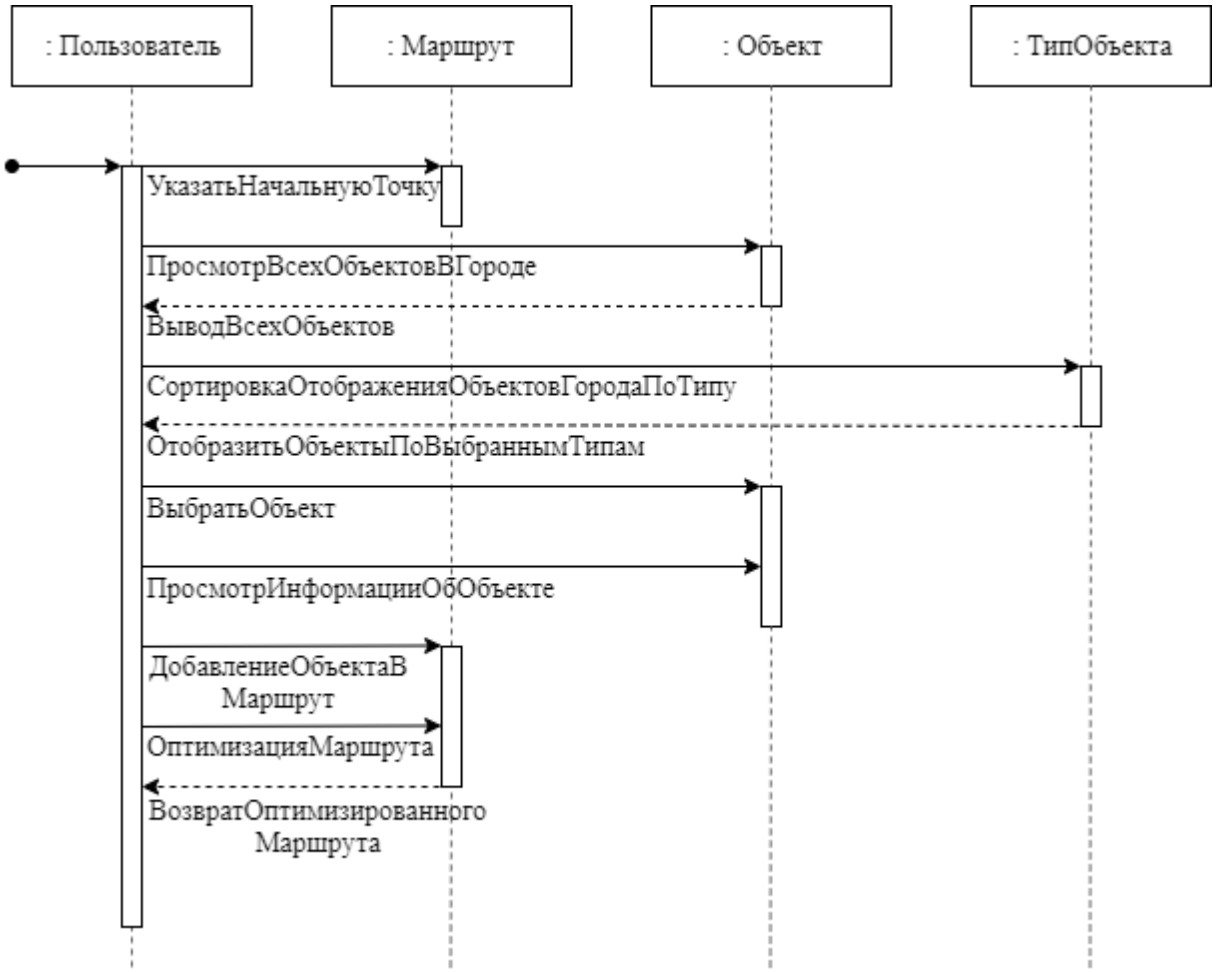


Рисунок 1.8 – Диаграмма последовательности для приложения «Онлайн Гид»

Прежде всего пользователю необходимо указать начальную точку для построения маршрута. Сделать это можно двумя способами: указать её самостоятельно путём манипуляций с картой или же включить передачу геоданных для автоматического определения местоположения устройства. Специфичность второго способа заключается в том, что для него необходимо наличие интернет соединения.

Следующим действием пользователя предусматривается просмотр всех доступных для изучения объектов, также можно произвести сортировку отображаемых маркеров на карте по типу. При выборе интересующего объекта, его можно добавить в маршрут, а также просмотреть информацию о нём прямо на текущей странице. Если объект имеет возможность платного посещения, то выводится так же цена билета. По окончанию подбора объектов для маршрута, его можно оптимизировать. Пользователю будут предложены несколько вариантов для его обновления. Теперь пользователь может следовать по маршруту.

Этап проектирования является определяющим в разработке программы, поскольку именно на данном этапе формируются требования к системе вместе с обозначением поведения системы при совершении каких-либо действий. Отсутствие чётких требований может повлечь за собой такие проблемы как несоответствие системы желаемым требованиям заказчика или, как наиболее опасный для заказчика исход, материальные убытки, которые могут превзойти затраты на техническое задание в несколько раз.

Таким образом, в 1 разделе было проведено проектирование в контексте приложения «Онлайн Гид», результатом которого являются диаграммы, описывающие функциональные требования к системе, которые станут основополагающими в разработке слоя бизнес-логики системы.

2 Описание работы и реализация приложения «Онлайн Гид»

2.1 Выбор инструментов для разработки приложения «Онлайн Гид»

Разработка программного обеспечения начинается с выбора подходящего инструментария, который поможет реализовать программисту весь задуманный функционал.

Для реализации серверной части был выбран объектно-ориентированный язык программирования C#. Клиентская часть реализуется с помощью платформы Xamarin [25].

Изначально C# поддерживался только .NET Framework платформой, которая поддерживается только операционной системой Windows. В 2016 году вышла первая официальная версия .NET Core, поддерживающая возможность кроссплатформенной разработки [13]. Вместе с приходом этого изменения пришли и другие, ориентированные на быстрое развёртывание приложения, включающие «continuous integration» и «continuous delivery» инфраструктуру. Именно эти факторы были наиболее весомыми при выборе языка разработки [14].

Процесс выбора инструментов для разработки стоит начать с определения того, что будет включать в себя конечный продукт.

Во-первых, открывая приложение на телефоне, первое, что видит пользователь, экранную форму входа. На данном этапе пользователю необходимо либо пройти авторизацию, либо, если он впервые вошёл в приложение, зарегистрироваться. Соответственно, на стороне сервера необходимо будет реализовать алгоритм аутентификации.

Также для аутентификации потребуется доступ к базе данных, чтобы проверять логин и пароль, введенные уже зарегистрированным пользователем. В свою очередь подключение базы данных полностью определяет слой доступа к данным приложения. Сюда входит и хранение информации о пользователе при регистрации, и информация, которую хранит сам пользователь, и

информация, которая необходима для работы и наполнения приложения, а также прочая служебная информация.

Во-вторых, после успешного прохождения авторизации, произойдёт переход на экранную форму с картой, на которой будут отображаться все доступные к просмотру и выбору маркеры.

Очевидно, что использование и взаимодействие с картой реализовывается на стороне клиента, но установка настроек для карты и маркеров должна происходить на стороне сервера. В качестве источника данных используем базу данных, а для сортировки и манипуляции ими воспользуемся LINQ.

Выбор мест и первичное составление маршрута реализуются на клиентской стороне. При этом описанные в первом разделе функциональные требования приложения подразумевают предоставление пользователю нескольких оптимизированных вариантов построенного маршрута. Данный функционал реализуется на серверной стороне посредством написания алгоритма для построения дерева принятия решений. Такой выбор расценивался как наиболее рациональный, поскольку при построении дерева принятия решений мы сразу же получаем несколько наилучших вариантов маршрута, которые будут построены относительно разных параметров (самые высокие и самые низкие затраты на посещение мест в маршруте, самый быстрый и самый долгий маршрут и т.д.).

Помимо вышеперечисленного функционала, пользователь будет иметь собственный личный кабинет, в котором он сможет просматривать созданные маршруты, а также ими управлять.

В завершении описания инструментов для разработки приложения стоит определиться со способом работы с объектами дополненной реальности. Этот пункт специально остался последним в перечислении, поскольку телефон, на котором будет вестись дальнейшая разработка и отладка, не отличается высокой

производительностью и большинство библиотек, работающих с дополненной реальностью, требуют высоких мощностей от устройства.

В результате анализа всех общедоступных библиотек, предоставляющих инстументы для работы с 3D моделями, которые не требовали бы больших вычислительных мощностей от устройства, выбор пал на «Unity», поскольку разработка осуществляется на языке C#, а также присутствует поддержка многопоточного кода, что позволит приложению увеличить скорость работы [12].

Разработка приложения начинается с написания серверной части приложения, где и будет описана вся бизнес-логика.

2.2 Разработка серверной части приложения «Онлайн Гид»

Прежде всего определим, что представляет из себя сервер и какова его цель. В программном обеспечении сервером называется программное обеспечение, целью которого является обслуживание запросов, приходящих от клиентов. Сервер и клиент постоянно звучат вместе, поскольку являются составляющими клиент - серверной архитектуры.

Основным преимуществом клиент - серверной архитектуры является возможность перенести вычислительную нагрузку с клиентских устройств на одну серверную машину. Особенно, это актуально в настоящее время, когда даже обычный студенческий ноутбук для учёбы имеет неплохие технические характеристики и вполне может послужить сервером для развёртывания какого-либо приложения для обслуживания нескольких клиентов одновременно.

Вдобавок к плюсам клиент - серверной архитектуры несомненно относится и доступ к базе данных, который обычно реализовывается на стороне сервера. Это обеспечивает клиентам стабильное соединение с базой во время работы со своими данными, отправляя лишь один запрос к серверу, который, в

свою очередь, будет использовать собственную вычислительную мощность для совершения манипуляций с базой, необходимых клиенту.

Помимо доступа к базе, сервер обязательно содержит в себе некую бизнес-логику, которая и нацелена на выполнение пользовательских запросов.

В сложившемся описании взаимодействия между клиентом и сервером не хватает некоторого промежуточного слоя. На данный момент, одним из самых популярных способов обмена сообщениями между клиентом и сервером является написание веб - API.

Веб - API - тот самый промежуточный слой, который позволяет серверу получать и распознавать запросы от клиента. Данное понятие состоит из двух частей:

– веб - способ обмена данными согласно HTTP протоколу;

– API (программный интерфейс приложения) - описание методов, предоставляющих доступ к бизнес-логике приложения.

Веб-API является одной из концепций веб-разработки, которая описывает методы для взаимодействия клиента с сервером, при этом исключая их внутреннюю реализацию.

Веб-API содержит в себе как минимум одну или несколько конечных точек (эндпоинтов). Любой эндпоинт представляет собой путь, описывающий метод предоставления доступа к ресурсу. Этот путь постоянно должен быть актуальным, т.е. при каких-либо изменениях в определениях методов или при смене расположения составляющих компонентов (файлов), API становится неработоспособным [16].

Для того, чтобы сохранять работоспособность приложения при обновлении функционала, веб-API поддерживает политику версионирования. Как клиент может заметить, что используемое API её поддерживает? При обращении напрямую к API из браузера в адресной строке должно содержаться выражение “v*”, где символ звездочки определяет версию, в которой выпущен

этот метод. Таким образом, версионирование позволяет обеспечивать и обратную совместимость приложения - когда клиент не обновился до актуальной версии приложения, а серверная часть обновлена, то клиент продолжает получать стабильную работу приложения [20].

Каждый метод веб - API представляет собой обработчик одного из REST - запросов. В результате обращения к одному из эндпоинтов, клиент получит объект в формате JSON или XML.

REST - одна из разновидностей архитектур веб - API, которая позволяет обрабатывать запросы (как правило HTTP) и возвращать результат из другого адресного пространства.

Системы с REST - архитектурой имеют ряд отличительных особенностей, которые и выделяют их среди других. Сюда входят:

– производительность;

– масштабируемость.

Во-первых, высокая производительность обеспечивается за счет компонентов, легко поддающихся изменениям, которые вносит разработчик для удовлетворения клиентских потребностей. При этом система всё ещё остаётся надежной в плане отказоустойчивости, поскольку в случае некорректной работы одного из компонентов, вся система продолжит работу и ошибка будет воспроизводиться только в конкретном эндпоинте.

Во-вторых, такая архитектура гарантирует беспроблемное масштабирование. REST позволяет свободно увеличивать или уменьшать количество компонентов (в том числе и эндпоинтов) за счёт своего единообразного интерфейса. Это обусловлено открытостью связей между компонентами и предоставляет возможность легкого расширения списка взаимодействующего функционала.

Масштабирование также обеспечивает простоту миграции (переносимости) программного кода вместе с данными использующимися

системой, что является немаловажным фактором при разработке и поддержке системы.

Теперь, когда мы знаем преимущества REST - архитектур, необходимо изучить и требования, которым должна соответствовать система, чтобы на выходе мы получили продукт, реализующий все вышеперечисленные преимущества архитектуры:

– приложение должно иметь клиент - серверную архитектуру, т.е. сервер и клиент должны быть представлены разными, как минимум, проектами. Это позволит добавлять, заменять, удалять и т.д. оба компонента, что делает приложение гибким не только для разработки, но и для использования непосредственно конечным пользователем;

– сервер не должен хранить в себе данные пользователя. Его контакт с пользовательскими данными ограничиваются только получением и распознаванием запроса для идентификации и записью обработанной информации в базу;

– унифицированный интерфейс серверной и клиентской частей для легкости в замене и поддержке кода;

– каждый ответ сервера должен иметь отметку о необходимости кэширования. Это поможет клиентскому приложнию определять является ли информация, пришедшая с сервера актуальной;

– серверная часть приложения должна иметь многоуровневую архитектуру. Под данным понятием подразумевается, что каждый слой не только отвечает за одну свою функцию – принятие, обработка, хранение данных и т.д., но и доступ имеет только в собственным функциям.

Выше упоминались эндпоинты REST – архитектуры. Эти конечные точки представляют собой ресурсы системы и физически они определяются в виде url – адресов (от англ. «Uniform Resource Locator» - унифицированный

определитель ресурса). Именно эндпоинты являются указателями методов, которые поддерживаются сервером, и в некоторых случаях в них могут передаваться параметры запросов от клиента.

REST – архитектура не завязана на какую-либо технологию. Использование любой выбранной вами технологии, которая будет обеспечивать взаимодействие REST, будет иметь ряд своих преимуществ [22].

Для разработки приложения, которое будет поддерживать обслуживание запросов от нескольких пользователей одновременно, было выбрано взаимодействие клиента и сервера по протоколу HTTP. Это решение обуславливается актуальностью данного решения среди реальных проектов, поскольку оно не является зависимым от выборанного для реализации языка программирования.

HTTP (от англ. HyperText Transfer Protocol – «протокол передачи гипертекста») – протокол передачи данных в виде HTML-кода, а в настоящее время поддерживает и другие форматы [21].

HTTP поддерживает большое количество методов. В это множество входят:

- get;
- post;
- put;
- delete;
- head;
- connect;
- options;
- trace;
- patch.

Все вышеперечисленные методы являются кэшируемыми и безопасными. При этом, из всех представленных методов, чаще всего в информационных системах используются первые 4.

Метод «get» (в переводе с англ. «получать») – запрашивает с сервера определённый ресурс. Результатом служит только тот набор данных, который возвращает этот метод на сервере.

Стоит обратить внимание, что метод «get» может использоваться вместе с параметрами. Чаще всего им служит идентификатор сущности (уникальный номер или последовательность симловов для каждого объекта), но так же параметром или группой параметров могут служить другие составляющие поля объекта. В случае использования параметра, который уникален для каждой сущности, результатом будет одна сущность, если же значение поля будет не уникально, то результат вернётся в виде массива сущностей. Помимо вышеперечисленных возможных параметров, ещё допустимо применение параметра для просмотра значения только запрашиваемого поля внутри сущности. Физически такие запросы можно распознать по увеличению запрашиваемого URL-адреса метода необходимыми полями и их значениями.

Метод «post» (в переводе с англ. «отправить») – отправляет на сервер определённую сущность, которую необходимо, например, добавить в базу или обработать на сервере. Результатом может быть, как состояние об успешности операции, либо же сущность, соответствующая успешной обработке объекта.

Метод «put» (в переводе с англ. «положить») – отправляет на сервер сущность или некоторые её поля, как и метод «post», при этом внутри этого запроса, также должен быть и параметр, по которому должна будет определиться сущность, требующая обновления (при условии, что она уже существовует). Это метод также поддерживает параметризацию, как и в «get» методе.

Метод «delete» (в переводе с англ. «удалить») – отправляет на сервер сущность или набор параметров для её опреления, которую требуется удалить (например, в контексте баз данных).

Для определения состояния запроса после выполнения, в HTTP предусмотрены числовые коды состояний. Они содержатся в ответе сервера и сопровождаются словесным пояснением. Все коды делятся на несколько разделов, сгруппированных по причине возникновения. Наиболее часто встречающиеся:

а) информационные:
- 100 Continue;
- 102 Processing;

б) успешные:
- 200 Ok;
- 201 Created;
- 202 Accepted;
- 204 No content;

в) перенаправление:
- 300 Multiple choices;
- 301 Moved permanently;
- 302 Moved temporarily;
- 304 Not modified;
- 305 Use proxy;
- 307 Temporary redirect;
- 308 Permanent redirect;

г) ошибка клиента:
- 400 Bad request;
- 401 Unauthorized;

- 403 Forbidden;
- 404 Not found;
- 405 Method not allowed;

д) ошибка сервера:

- 500 Internal server error;
- 502 Bad gateway;
- 503 Service unavailable;
- 504 Gateway timeout.

Ещё одним преимуществом HTTP является передача фрагментов кода на языке «HTML» и их воспроизведение на стороне клиента.

Чем ещё полезен выбор HTTP для взаимодействия между клиентом и сервером? Выбранный протокол позволяет управлять аутентификацией, которую также необходимо реализовать в приложении «Онлайн гид» [6].

Аутентификация – процесс определения существования пользователя в системе. Чаще всего аутентификация пользователя происходит по логину и паролю: сущность со значениями логина и пароля отправляется на сервер, далее проверяется база данных, в которой хранятся все пользователи. Если такой пользователь найден – аутентификация пройдена и можно пользоваться приложением, в противном случае – нет, а значит, человек, вводивший исходные данные, не имеет доступа к системе.

Описанный выше алгоритм является самым простым для понимая процесса. В настоящее же время основным критерием хорошей системы считается его безопасность. Это относится и к аутентификации. Помимо того, что у злоумышленников не должно быть возможности перехватить данные пользователя, так же стоит учитывать и случаи, когда им всё-таки удаётся украсть данные пользователя. В таких ситуациях, система должна предусматривать обмен максимально непонятной для просмотра информацией.

Одним из актуальных, а вдобавок и безопасных, видов аутентификации на данный момент считается аутентификация на основе JWT.

Что такое JWT? Для начала, это аббревиатура от английского «JSON Web Token». В основе JWT аутентификации лежит шифровка уникальных полей пользователя вместе с некоторыми служебными полями, формирование из этого токена и дальнейшая его передача клиенту, который в свою очередь будет каждый раз передавать этот токен серверу для того, чтобы получить доступ к ресурсу. Если сервер декодирует полученный токен, проверяет значения полей, назначенных для определения пользователя, составит по ним подпись и она совпадёт с той, что будет сгенерирована по имеющимся данным по пользователю, то считается, что пользователь прошёл аутентификацию.

Сам токен состоит из 2 частей: заголовок («header») и подложка («payload»).

Заголовок содержит в себе полностью служебные данные. Это тип формируемого токена и тип алгоритма, по которому он будет шифроваться. Определяются на этапе разработки, поскольку для всех пользователей будут иметь одно и то же значение.

Среди популярных алгоритмов шифровки можно выделить такие как «HS256» и «RS256». Первая аббревиатура расшифровывается как «HMAC-SHA256».

Алгоритм «HMAC-SHA256» основывается на функциях хэширования. Для его работы необходим секретный ключ, который должен быть статичным и задаваться на стороне сервера единожды во время разработки. Если секретный ключ будет изменён во время использования приложения пользователями, то они больше не смогут войти под своими старыми учётными данными, поскольку алгоритм не сможет корректно расшифровать поступающие данные и пользователи будут незнакомы системе.

«HS256» является ассинхронным методом шифрования, а так же дважды применяет алгоритм хэширования. Такое поведение не позволяет восстановить зашифрованную информацию даже при наличии секретного ключа, что гарантирует отсутствие доступа к системным данным у сторонних пользователей.

В основе «RS256» или же «RSA» лежит криптографический алгоритм. Для шифрования данным способом, так же требуется секретный ключ, но уже не один. На стороне клиента генерируются открытый и закрытый ключи. Открытый ключ может сообщаться кому угодно, а закрытый должен оставаться неизвестным. Эта пара ключей позволяет создать зависимость, которая позволят шифровать и дешифровать данные. Надёжность кодирования данных зависит от сложности этих ключей, по которым в процессе будут вычисляться функции шифрования и обратная ей.

В подложке же шифруются уже данные по каждому конкретному пользователю. Сюда входит издатель и получатель токена, назначение токена, дата его создания, срок начала и конца его действия, а также его идентификатор. Эти поля по умолчанию доступны у каждого токена. Сюда же попадают пользовательские поля и их значения, определяемые на стороне сервера для идентификации каждого пользователя. Ещё такие пары полей и их значений называются «claims» (от англ. «заявки»).

В завершении формируется подпись из полученных зашифрованных заголовка и подложки. Это позволяет сохранить подпись неизменной, если токен будет перехвачен и данные в нём претерпят изменения, поскольку секретный ключ не будет храниться в самом токене.

После того, как пользователь прошёл аутентификацию, он должен будет попасть на новую страницу с картой. Выбор карты для использования в приложении не уступает по важности вышеперечисленным аспектам

разработки, поскольку стоимость поддержки конечного продукта напрямую зависит от него.

Ввиду того, что ведётся разработка учебного проекта, ограничения в выборе библиотеки для работы с картой минимальны. Практически любой крупный и известный производитель ПО, предоставляющий свои карты во всеобщее пользование, разрешает бесплатную интеграцию в любой продукт, который в дальнейшем не будет монетизироваться. Среди таких крупных производителей сразу же на ум приходят компании «Яндекс» и «Google».

Для разработки были выбраны карты «Google». Этот выбор не обуславливается колоссальной разницей в функционале продуктов. Здесь элементарно сыграл фактор внешнего вида карт, которые были сочтены как наиболее подходящие для конечного продукта.

Начинать интеграцию с «Google» картами следует с получения API ключа, который предоставит доступ к API карт. Сделать это можно в консоли для разработчиков от «Google» (рис. 2.1). Для этого на соответствующем сайте необходимо будет создать проект для своего приложения, заполнить все информационные поля и придумать уникальное имя для него. Далее нужно будет открыть доступ к API и службам, мы окажемся на странице с выбором SDK (от англ. «Software Development Kit» - набор для разработки ПО), выбираем пакет для Android системы и включаем её – мы подключили API.

Для получения нужного нам API ключа, необходимо войти в раздел с набором для разработки, который мы включили ранее. Перейдя во вкладку «Учётные данные», выбираем тип генерируемых данных. В нашем случае, это API ключ, выбираем его из выпадающего списка. Вновь заполняем поля, требующиеся для создания ключа – это уникальное имя для него и ограничения, накладываемые на приложение. Поскольку мы разрабатываем мобильное приложение для операционной системы «Android», то выбираем соответствующий пункт в меню. Сохраняем изменения.

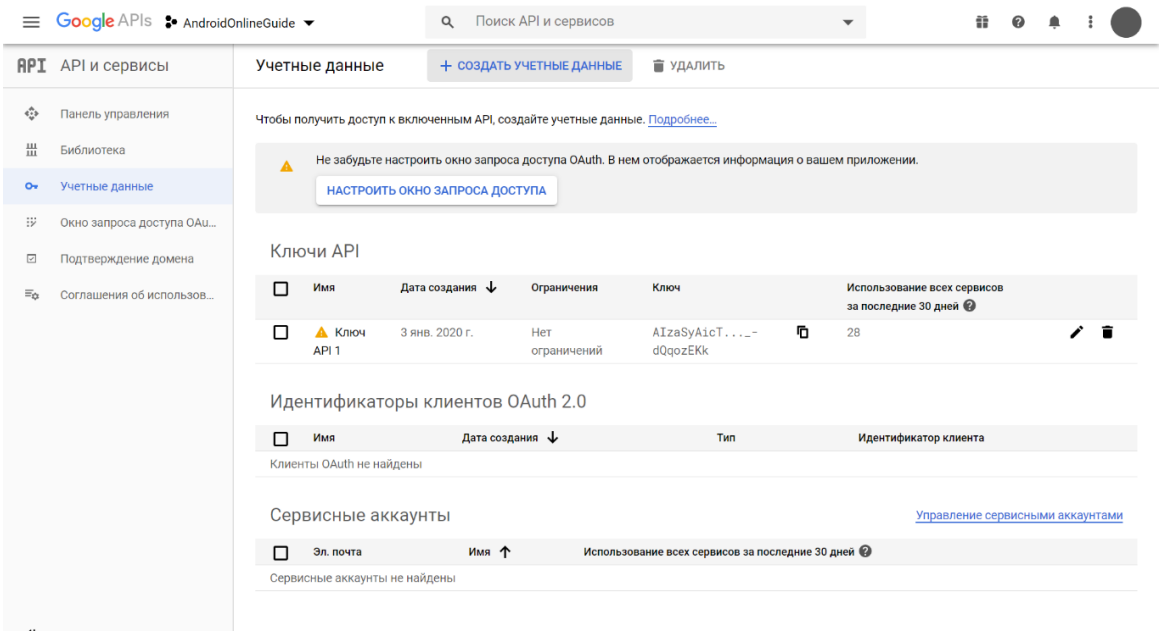


Рисунок 2.1 – Консоль разработчика «Google» для приложения «Онлайн Гид»

После успешного завершения создания API ключа, мы окажемся на странице со списком всех наших ключей. Теперь мы можем скопировать его и разместить в своём приложении.

Теперь на серверной стороне можно определить настройки для отображения карт на мобильном клиенте. Это и координаты для отображения по умолчанию, и приближение карты, и данные для отображения маркеров.

Следующим вопросом для рассмотрения является способ конфигурирования карты с сервера. Если настройки, не требующие передачи информации о значениях нескольких полей можно с лёгкостью разместить в файле настроек приложения, то сущности объектов, размещаемых на карте, мало того, что содержат в себе много полей, так ещё и могут быть редактируемыми. Поэтому целесообразно хранить их в базе данных.

Доступ к базам данных непосредственно из приложения реализовывается посредством ORM (от англ. «Object-Relational Mapping» – объектно-

ориентированное отображение). Эта технология обеспечивает транспортировку записей из базы данных и их последующее преобразование в объекты, определённые в приложении, а также обратный процесс – помещение экземпляров классов в соответствующие таблицы базы данных.

Основным преимуществом использования ORM – технологий, по сравнению с ручным взаимодествием с базой, является поддержка реляционных баз данных. Несмотря на то, что существуют и другие виды баз данных (такие как объектно-ориентированные или иерархические), реляционные базы данных в настоящее время пользуются наибольшей популярностью среди реальных проектов.

Особенностью реляционных баз данных является тот факт, что таблицы внутри таких баз могут содержать только простые данные. При этом один объект может быть описан более, чем одной таблицей. В таких случаях, нам потребовалось бы искать альтернативу оператору «JOIN» в языке запросов SQL, чтобы конечный результат выводился в полном составе, а не в виде ссылок или внешних ключей на него.

Ещё одним немаловажным фактором является эффективность работы ORM. Очевидный факт – для запросов, результатом которых должна быть большая выборка, способ обращения к базе напрямую будет более эффективным. В нашем же случае, когда речь идёт о небольшом проекте, ORM-технология предоставит большую производительность, нежели прямые запросы к базе. Во – первых, она избавляет разработчика от написания лишнего кода, который зачастую однообразен. Соответственно, меньше непредвиденных ошибок. Во – вторых, как говорилось выше, ORM сразу производит конвертацию таблиц в описанные на языке разработки объекты.

Поскольку разработка приложения ведётся на языке программирования C# платформы .NET Core, то ORM – технология предоставляется компанией – издателем «Microsoft» в виде библиотеки Entity Framework [17].

Работа с Entity Framework начинается с выбора принципа работы с базой. Определить его можно по наличию базы для проекта. Если база отсутствует, то подойдёт один из двух способов взаимодействия «Code First» и «Model First», в противном же случае – «DB First».

«Code First» подход основан на программном описании классов, которые будут представлять сущности базы банных. Также при таком способе легко определить сущности необходимые в базе.

«Model First» подход заключается в построении «EDMX» модели базы данных. Сделать это можно с помощью инструментов, выпускаемых вместе со средой разработки «Visual Studio». После определения этой модели изменять поля сущностей уже будет нельзя и необходимо будет вновь создавать модель.

«DB First» подход требует наличия готовой и сконфигурированной базы данных. Для успешной работы при таком подходе, требуется основываться на той структуре базы, которая уже есть, а не описывать сущности, считающиеся верными в текущем контексте.

Для разработки приложения «Онлайн Гид» был выбран «Code First» подход взаимодействия, поскольку во время проектирования системы была разработана структура базы данных, то готовая база данных отсутствовала, вследствие чего для разработки приложения необходимо только повторить её в виде объектно-ориентированных классов.

Следующий шаг работы с базой – её конфигурирация. Entity Framework позволяет устанавливать атрибуты на членах класса, которые определяют следующие аннотации:

- первичный ключ;
- внешний ключ;
- требования к длине содержимого поля;
- обязательно ли поле для заполнения;

– сопоставление имени таблицы и столбцов.

К сожалению, Entity Framework не предоставляет возможностей для настройки отношений между сущностями. Для реализации связей существует библиотека «Fluent API».

«Fluent API» предоставляет комплект методов для конфигурации как отношений, так и столбцов сущностей. Для этого необходимо определить специальный класс настроек, наследующийся от конфигурационного интерфейса предоставлямого библиотекой. Затем в каждый класс, который нуждается в установке связи, помещается ссылка на класс – наследник.

На данном этапе могут появиться трудности с использованием конфигурационных методов. Это обусловлено тем, что «Fluent API» напрямую реализует только отношения «один-к-одному» и «один-ко-многим».

Для реализации «один-к-одному» требуется в связанных классах определить свойства, которые будут ссылаться друг на друга, а затем в конфигурационном классе сущности – родителя последовательно использовать методы «HasOne» затем «WithOne».

Для реализации «один-ко-многим» требуется в классе – родителе определить свойство с ссылкой на наследника, а в дочерней сущности отпределить массив ссылок на родителя, что позволит в дочерней сущности иметь ссылки на несколько экземпляров класса. Затем в конфигурационном классе-родителе использовать методы «HasOne» затем «WithMany».

В большинстве случаев, в разработке требуется настройка отношения «многие-ко-многим». Эта связь не поддерживается прямыми методами «Fluent API», поэтому требуется создание промежуточной сущности, которая будет содержать, как минимум, два столбца. Внутри них будут уникальные поля сущности – родителя и сущности – наследника. Этот класс задаётся вручную и должна содержать в себе 4 свойства:

– ссылка на сущность – родителя;

– свойство для хранения значения ключа родителя;

– ссылка на сущность – наследника;

– свойство для хранения значения ключа наследника.

В завершении работы с настройкой связи «многие-ко-многим» в классе – конфигураторе этой промежуточной сущности, во – первых, необходимо создать составной первичный ключ, состоящий из уникальных записей таблицы. Во – вторых, нужно последовательно использовать методы «HasOne» и «WithMany» сначала для одной сущности, затем для второй. В данной ситуации, порядок применения методов не играет роли.

Теперь, когда структура базы определена, требуется задание строки подключения, чтобы настроить место инициализации базы. Такая строка имеет следующий вид: «Server = .\\SQLEXPRESS; Database = OnlineGuideDB; Trusted_Connection = True; User ID = OnlineGuideServer; Password = Qwerty312», где:

– «Server» – имя сервера, который будет хостить базу данных;

– «Database» – имя базы данных, по которому будет происходить подключение на стороне приложения;

– «Trusted_Connection» – включение проверки подлинности;

– «User ID» – логин для доступа к базе данных;

– «Password» – пароль для пользователя, имеющего доступ к базе данных.

Entity Framework принято использовать вместе с библиотекой LINQ, которая является аналогом языка запросов SQL для коллекций объектов классов и входит в состав BCL (от англ. «Base Class Library» – библиотека базовых классов) языка C#.

LINQ (от англ. «Language Integrated Query» – язык интегрированных запросов) предоставляет разработчику унифицированный интерфейс для

запросов к различным источникам данных и возвращающий соответствующие условиям коллекции объеков [1].

Основными преимуществами LINQ являются:

– возможность анонимной типизации;

– допускается использование методов расширения;

– разрешается задание условий путём написания лямбда - выражений и пр.

Следует уточнить, что LINQ предназначен не только для классов программы, а также и для других источников данных. Данное поведение достигается за счёт реализации интерфейса «IQueryable» к внешним данным [3].

Интерфейс «IQueryable» не перегружен какой-либо логикой, его основные цели – это связь между базой и приложением, а также оптимизация запросов к базе данных. Запрос напрямую к базе был бы более затратным в использовании памяти, особенно при применении нескольких операторов. Если же использовать возможности «IQueryable», то сначала будет формироваться синтаксическая структура запроса, а при использовании дополнительных операторов, она будет наращиваться на вызываемый метод.

В то же время этот интерфейс оказывается и проблемой для разработчика, поскольку он требует явного преобразования в «IEnumerable», который предоставит результат запроса в виде массива исходных объектов. Для этого используется соответствующий метод «ToIEnumerable» или «ToList», который является наследником «IEnumerable». Но даже с учётом этой особенности, такой подход к запросам к базе является более эффективным, чем прямые запросы [5].

Существует несколько разновидностей технологии LINQ:

– LINQ to objects;

– LINQ to entities;

– LINQ to SQL;
– LINQ to XML;
– LINQ to DataSet;
– Parallel LINQ.

Наиболее популярными являются первые 3 типа, они так же используются и в разрабатываемом приложении [23].

«LINQ to objects» используется для всех коллекций внутри приложения. Применение методов данного типа значительно увеличивают скорость работы обработки коллекции, по сравнению с аналогичной обработкой, но с помощью цикла «foreach». Кроме того «LINQ to objects» предоставляет мощный функционал для фильтрации, группировки и упорядочивания, имеющий лаконичный вид, избавляющий разработчика от написания громоздкого кода.

«LINQ to entities», соответственно своему названию, предназначены для работы с объектами Entity Framework. Эти методы ориентированы на работу с SQL, поэтому имеют схожий синтаксис с оригинальным языком запросов.

«LINQ to SQL» позволяет разработчику выполнять команды на оригинальном языке запросов, а также предоставляет инструменты для разработки хранимых процедур и пользовательских функций прямо на стороне приложения.

Помимо вышеперечисленных преимуществ использование технологий LINQ считается более лёгким в контексте восприятия кода для разработчика, чем последовательные условные операторы. Вдобавок к повышенной скорости разработки несомненно добавляется эффективная отладка, за счёт которой сокращается время обнаружения ошибок уже на этапе написания и компиляции приложения.

В завершении повествования работы пользователя с приложением и окончания описания всех интеграций, согласно функциональным требованиям,

требуется наличие алгоритма для оптимизации маршрутов. В качестве алгоритма для оптимизации построенных маршрутов был выбран алгоритм построения деревьев принятия решений «CART».

2.3 Описание математической модели модуля оптимизации маршрутов, построенных в приложении «Онлайн Гид»

Алгоритм построения деревьев принятия решений «CART» был выбран для решения задач оптимизации маршрутов в приложении «Онлайн гид». Выбор на данный алгоритм пал неспроста. Наиболее весомым преимуществом для разрабатываемого продукта является построение бинарных деревьев, которые в качестве результата работы предоставляют нам наилучшие объекты для каждого выходного признака.

Алгоритм «CART» (от англ. «Classification And Regression tree» - дерево классификации и регрессии) разрабатывался с 1974 по 1984 годы 4 учёными: Лео Брейманом, Джеромом Фридманом, Чарлзом Стоуном и Ричардом Олшеном.

Данный алгоритм основан на построении бинарных деревьев, поскольку для классификации необходимо строгое определение: соответствует ли объект текущему признаку или нет. Эту потребность как раз и реализует бинарное дерево, поскольку оно может иметь только 2 потомков (поэтому его второе название «двоичное дерево») [19].

Результатом построения деревьев принятия решений по алгоритму «CART» являются наборы объектов, которые считаются наиболее «подходящими» по выходному признаку. Каждый узел бинарного дерева подразумевает разбиение на две группы: справа располагаются объекты удовлетворяющие правилу в узле, слева, соответственно, не подходящие под него.

Как упоминалось выше, структура бинарного дерева обеспечивает мининальную неопределённость. В алгоритме «CART» неопределённость формализуется на основе индекса «Gini». Индекс «Gini» вычисляется по формуле 3.1:

$$Gini(T) = 1 - \sum_{i=1}^{n} p_i^2, \tag{3.1}$$

где n – общее количество объектов, p_i – относительная частота класса i в T.

В случае разделения T на две части T1 и T2, где в T1 наборе N1 объектов и в T2 наборе N2 объектов, Gini вычисляется по формуле 3.2:

$$Gini_{split}(T) = \frac{N_1}{N} \cdot Gini(T_1) + \frac{N_2}{N} \cdot Gini(T_2). \tag{3.2}$$

Для определения наилучшего разбиения, выбирается минимальный показатель $Gini_{split}(T)$. Качество полученного разбиения вычисляется по формуле 3.3:

$$Gini_{split} = \frac{L}{N} \cdot \left(1 - \sum_{i=1}^{n} \left(\frac{l_i}{L}\right)^2\right) + \frac{R}{N} \cdot \left(1 - \sum_{i=1}^{n} \left(\frac{r_i}{L}\right)^2\right) \rightarrow min, \tag{3.3}$$

где N – количество примеров в узле – родителе, L – количество примеров в левом потомке, R – количество примеров в правом потомке, l_i – количество примеров i - го класса в левом потомке, r_i – количество примеров i - го класса в правом потомке.

Для уменьшения вычислений формула может претерпевать следующие изменения и становится вида формулы 3.4:

$$Gini_{split} = \frac{1}{N} \cdot \left(L \cdot \left(1 - \frac{1}{L^2} \cdot \sum_{i=1}^{n} l_i^2\right) + R \cdot \left(1 - \frac{1}{R^2} \cdot \sum_{i=1}^{n} r_i^2\right)\right) \rightarrow min. \tag{3.4}$$

Поскольку константы не влияют на результат минимизации, то можно преобразовать имеющуюся формулу последовательно в формулы 3.5, 3.6, 3.7:

$$Gini_{split} = L - \frac{1}{L} \cdot \sum_{i=1}^{n} l_i^2 + R - \frac{1}{R} \cdot \sum_{i=1}^{n} r_i^2 \rightarrow min, \quad (3.5)$$

$$Gini_{split} = N - \left(\frac{1}{L} \cdot \sum_{i=1}^{n} l_i^2 + \frac{1}{R} \cdot \sum_{i=1}^{n} r_i^2 \right) \rightarrow min, \quad (3.6)$$

$$\hat{G}_{split} = \frac{1}{L} \cdot \sum_{i=1}^{n} l_i^2 + \frac{1}{R} \cdot \sum_{i=1}^{n} r_i^2 \rightarrow max. \quad (3.7)$$

Так как выше мы избавились от левых и правых потомков, которые в сумме являются общим количеством примеров в родительском узле, то теперь для выбора наилучшего варианта нам необходимо ориентироваться на максимальные показатели разбиений.

Основными преимуществами «CART» перед другими алгоритмами являются:

– высокая скорость работы алгоритма;

– алгоритм не требует вычисления дополнительных переменных или ввода параметров, всё необходимое вычисляется во время обработки набора объектов, а выбор основывается на значениях индекса «Gini»;

– алгоритм «самоочищается» от шумов, посредством использования механизма разбиения, который перемещает такие объекты в отдельный узел;

– построенные деревья решений не являются статичными, при обновлении исходного набора объектов, будет построено новое дерево с новыми результатами;

– дерево принятия решений, построенное по «CART», имеет более простую структуру, чем деревья построенные другими алгоритмами.

Даже при наличии всех этих достоинств, алгоритм имеет и свои недостатки, ограничивающие возможности приложения.

Поскольку алгоритм «CART» входит в число алгоритмов, которые требуют обучающих данных, то результат работы может не удовлетворять ожидающимся прогнозам из-за качества подобранных для выборок.

Также завершение работы алгоритма может сопровождаться построением слишком сложных структур бинарных деревьев, которые трудно поддаются обобщению. В таких случаях, обязательно использование специальных механизмов сокращения или отсечения ветвей. Например, подход условного вывода, который не производит уменьшение размеров дерева за счёт отсечения ветвей с малым ветвлением, а совершает дополнительные вычисления, позволяющие обобщать большее число объектов и производить более точную классификацию [11].

Для внедрения алгоритма в разрабатываемое приложение необходимо определить, каким именно образом следует подходить к оптимизации маршрутов: результатом работы будет выборка объектов на каждый выходной признак или же им будет один объект на каждый выходной признак, который потребует зацикливания для всех исходных объектов [4].

Опытным путём было определено, что цикличный подход к оптимизации работает более эффективно, чем вывод сразу готовых выборок. Для этого был подготовлен массив входных данных, который был обработан вышеупомянутыми алгоритмами. Результаты продемонстрировали, что подход с готовыми выборками работает быстрее, но они не являются наиболее оптимальными по выходному признаку. Подход с цикличным вычислением оказался менее производительным, но выходным выборки были рассчитаны, как наиболее подходящие, поэтому было принято решение использовать именно его.

В результате разработки информационной системы была получена серверная часть, способная обрабатывать запросы пользователей на аутентификацию, регистрацию, получение всех доступных к посещению мест, построению маршрутов и пр.

Процесс разработки характерен анализом актуальных решений и последующим выбором подходящих инструментов. Разработка информационной системы «Онлайн Гид» производилась на платформе .NET Core 3.1 в связке с объектно-ориентированным языком программирования C#. Такой выбор позволил увеличить скорость разработки и не концентрировать внимание на конфигурации приложения для развертывания [24].

Таким образом, в результате 2 раздела был реализован слой бизнес-логики, поскольку он не требует никаких внешних интеграций. Затем настраивался доступ к базе данных, чтобы логика могла манипулировать набором каких-либо данных. После этого велась разработка взаимодействия серверной части с клиентской. Завершилась реализация серверной части добавлением к слою бизнес-логики модуля оптимизации маршрутов, построенных в приложении «Онлайн Гид».

3 Оценка качества разработанной системы «Онлайн Гид»

3.1 Функциональное тестирование серверной части приложения «Онлайн Гид»

Тестирование представляет собой воспроизведение реализванных функций и на основе полученных результатов формируется оценка качества продукта.

Существует несколько видов тестирования [9]:

- функциональное;
- нефункциональное;
- связанное с изменениями.

Функциональное тестирование проводится с целью проверки качества отдельных функций на разных уровнях: в полной изоляции от других методов, в интеграции с другими функциями или системами и пр. Другими словами, данный вид тестирования позволяет рассмотреть поведение системы «снаружи».

Первым шагом при таком тестировании является покрытие кода модульными тестами, которые гарантируют работоспособность тестируемых функций. Такое тестирование не обеспечивает корректность работы, но позволяет удостовериться, что функция не будет возвращать пользователю ошибку или приводить к падению всей системы.

При разработке приложения «Онлайн гид» было создано 6 проектов. Из них 4 относятся к серверной и 2 к клиентской части. В процессе разработки все тестируемые функции из 4 проектов были покрыты модульными тестами.

Показатель качества написанного кода на уровне модульных тестов выражается в процентном соотношении покрытия тестами. В контексте разработки «Онлайн гида» этот показатель должен быть выше 60%, поскольку проект учебный и не содержит в себе сложных вычислений.

Проверить показатель покрытия модульными тестами можно с помощью соответствующей функции, приходящей вместе с Visual Studio Enterprise, но поскольку получение данной версии несколько проблематично, то можно воспользоваться сторонними утилитами. В качестве такой программы было выбрано расширение для Visual Studio «dotCover» от компании «JetBrains».

Данное расширение выполняет необходимый анализ кода и показывает процентное покрытие тестами. Единственный его минус – цена лицензионного продукта. При этом компания-производитель предоставляет 30 дней бесплатного доступа для апробации функционала. Этого времени должно хватить на проверку покрытия уже имеющимися юнит-тестами.

Поскольку данное тестирование нацелено на проверку работоспособности компонентов серверной части, а также всвязи с отсутствием достаточного опыта по тестированию логики мобильных приложений, то проекты с клиентской частью были обозначены специальным атрибутом «ExcludeFromCodeCoverage», который исключает их при анализе покрытия кода.

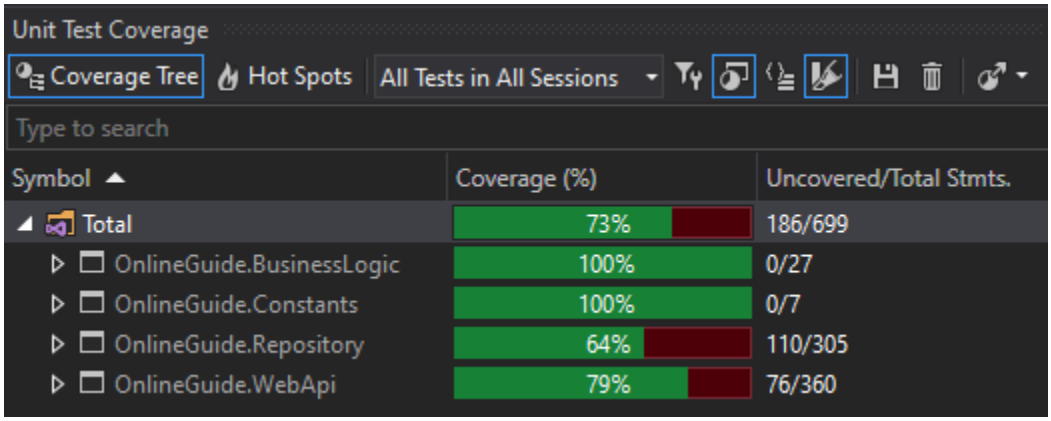


Рисунок 3.1 – Показатели покрытия юнит-тестами серверной части приложения «Онлайн гид»

Далее следует углубиться в работу конкретных функций. Поскольку способ взаимодействия с сервером реализован посредством HTTP, то большую часть внимания следует уделить именно веб – API. В контексте разработанного

продукта сделать это можно, как минимум, двумя способами: напрямую использовать интерфейс веб – API, предоставляемый библиотекой «Swagger» или отправлять запросы из сторонней программы, например, «Postman» или «SoapUI».

Воспользуемся обоими способами, чтобы наглядно продемонстрировать работоспособность методов в условиях смены источника запроса. В случае прямых запросов на страницах веб – API, адрес запросов – внутренний и «CORS» политика не действует на запросы. В случае запросов извне, например, из «Postman» запрос будет идти с другого домена, «CORS» политика может быть нарушена и клиент получит ответ с ошибкой.

Проведём тестирование метода «POST» в контроллере «User» (приложение А). Данный метод был создан для прохождения пользователем регистрации. В качестве аргумента принимает сущность пользователя, а возвращает сущность ответа со статусом запроса и описанием результата запроса.

Метод создания пользователя был выбран для тестирования неспроста. Этот метод общедоступен и не требует прохождения аутентификации.

Большинство методов покрыты атрибутом «Authorize» и требуют токена для доступа к ним. В обоих случаях сначала необходимо отправить запрос к эндпоинту токена с аргументом, содержащим сущность с логином и паролем зарегистрированного пользователя. Далее на веб – API потребуется ввести полученный токен в окно авторизации с приставкой «Bearer». В случае с «Postman», необходимо воспользоваться вкладкой «Authorization» и выбрать из выпадающего списка типов аутентификации «Bearer token». В поле рядом с этим списком ввести полученный токен. Таким образом, проверяются все остальные методы, связанные с местами города и маршрутами. На данном этапе фукнциональное тестирование заканчивается.

Успешным результатом любого из методов будет двухсотый ответ от сервера, а так же объект с полями «success» со значением true, а также «resultDescription» со строковым значением описания результата, не содержащего слова «error».

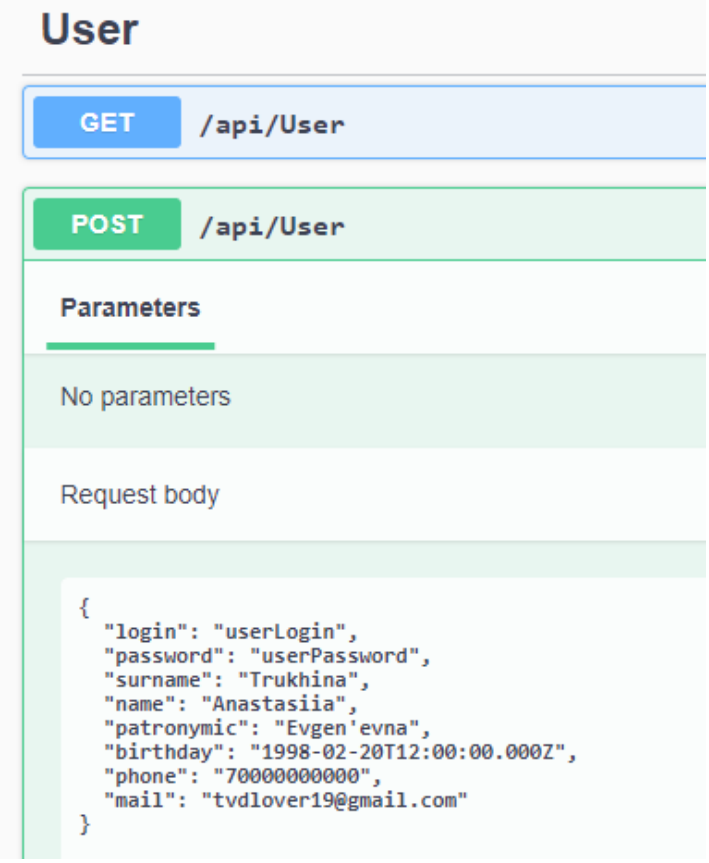


Рисунок 3.2 – Запрос на создание пользователя из веб-API

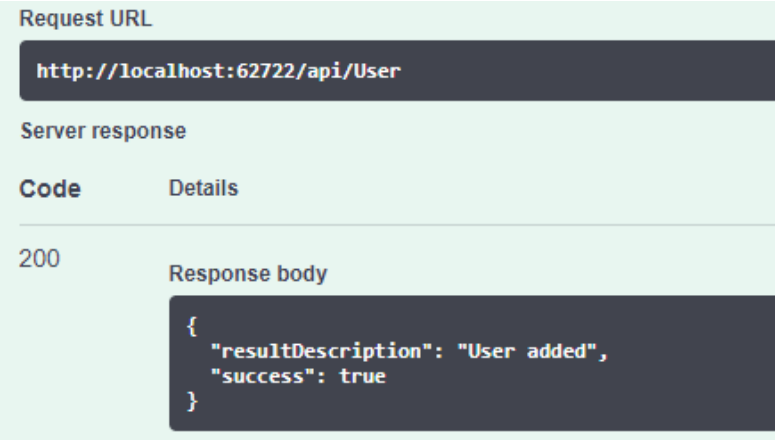


Рисунок 3.3 – Успешный ответ на создвание пользователя из веб-API

Отправим тот же самый запрос из «Postman». Для этого нужно только сформировать JSON сущность, повторяющую структуру объекта на сервере и написать в адресную строку соответствующий адрес.

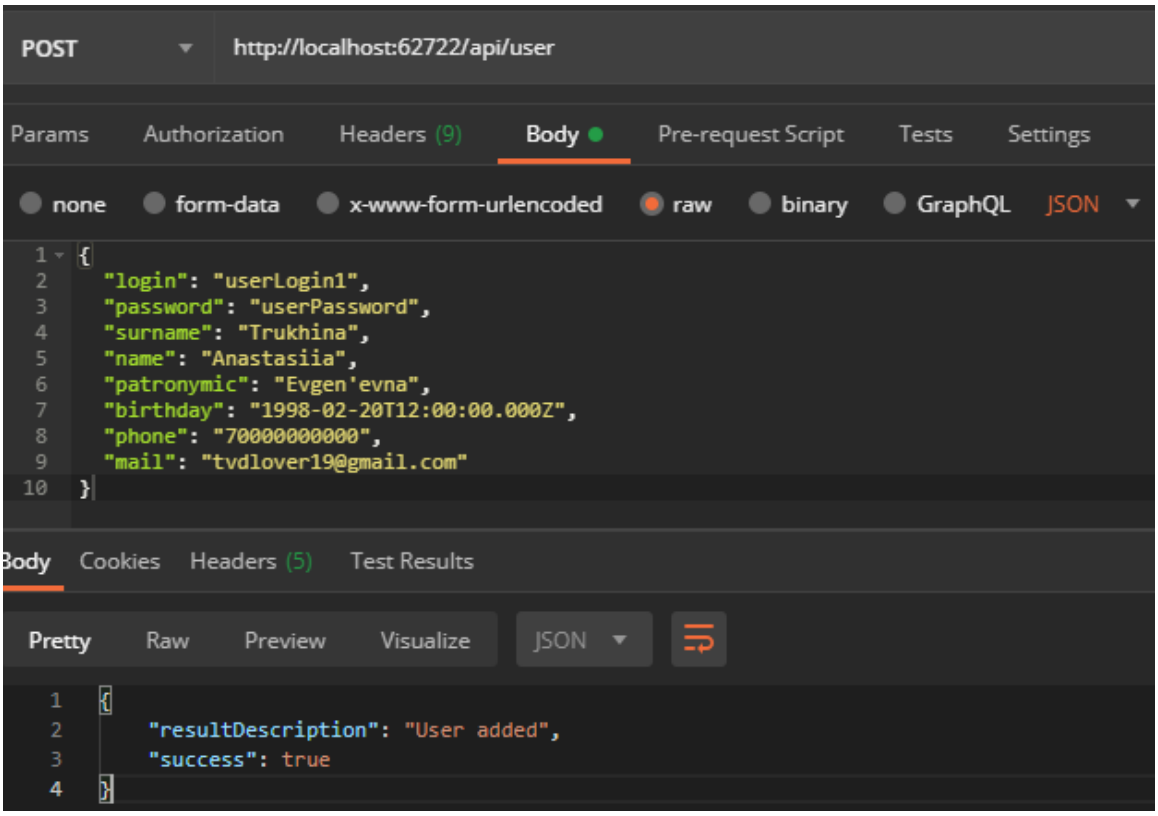


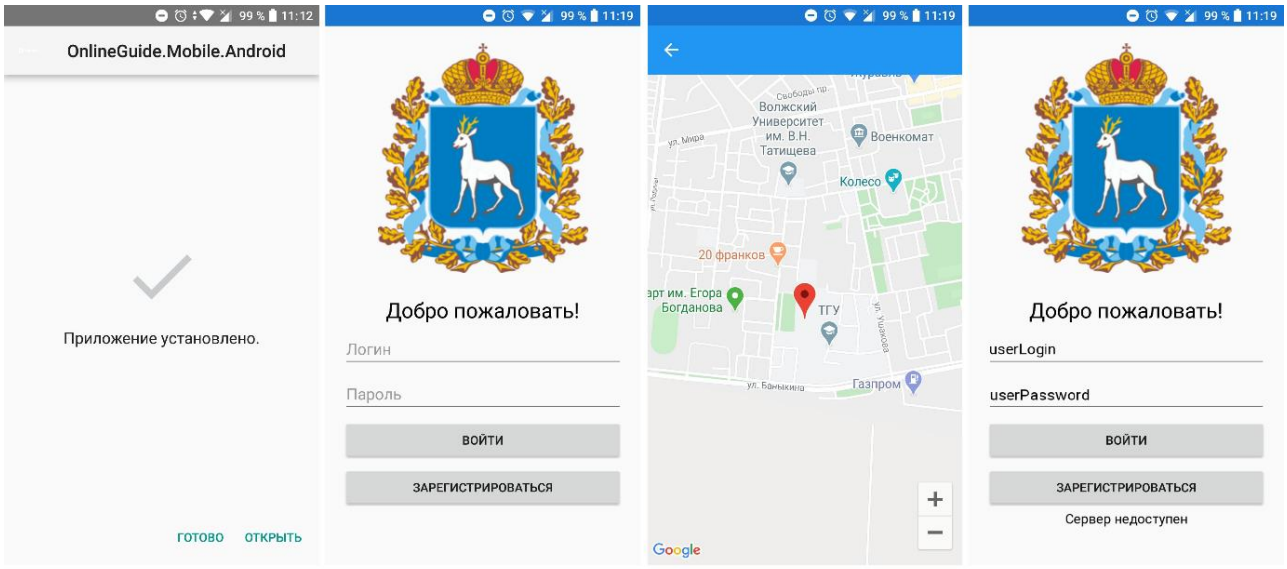
Рисунок 3.4 – Запрос на создание пользователя и ответ с помощью «Postman»

Таким образом, было произведено тестирование всех разработанных методов. Результирующие ответы сервера на все запросы содержали в себе поле «success» со значением true, что демонстрирует успешную обработку и корректное выполнение запроса.

3.2 Нефункциональное тестирование приложения «Онлайн Гид»

Нефункциональное тестирование своим названием подразумевает проверку не отдельных функций системы, а её поведение в целом.

Одним из видов такого тестирования является тестирование установки. Для его проведения в контексте приложения «Онлайн Гид» была создана первая релизная версия серверной и клиентской частей, затем сервер был развёрнут на той же машине, а полученный «apk» – файл сборки для Android был скачан на мобильное устройство.



А Б В Г

Рисунок 3.5 – Экранные формы при установке и использовании мобильного приложения «Онлайн гид»: А – после успешной установки, Б – после открытия приложения, В – после успешной попытки входа при доступном сервере, Г – после попытки входа при недоступном сервере

Тестирование позволяет определить в приложении баги, связанные с неправильной реализацией логики в нём, ведь они в дальнейшем могут повлечь за собой серьёзные проблемы с приложением и спровоцировать отток пользователей, а также отследить тонкие места в инкрементации продукта.

Таким образом, было проведено тестирование приложения «Онлайн Гид», которое сопровождается успешными результатами функционального и нефункционального тестирования установки. Полученные результаты после

тестирования показывают, что разработанный функционал работает исправно, а также следует корректной логике. Следовательно, мы имеем высокое качество разработанного фукнционала приложения «Онлайн Гид», а также соответствие системы всем объявленным требованиям.

Заключение

Завершающим разделом в бакалаврской работе является формулировка выводов, сложившихся в процессе её написания.

Прежде всего следует определить прогресс в достижении цели, поставленной для выполнение выпускной квалификационной работы. Для этого просмотрим все задачи, реализованные в процессе её написания.

Во-первых, в результате проектирования информационной системы, были получены необходимые знания в области онлайн туризма, которые позволили сформулировать требования к программному продукту, способному в конечном счёте конкурировать с наиболее популярными аналогами на рынке.

Во-вторых, был проведён анализ технологий, используемых в актуальных проектах, а также их изучение. Отталкиваясь от вычислительных мощностей, имеющихся в доступе, были подобраны соответствующие инструменты для дальнейшей реализации. Так же были проанализированы известные алгоритмы для построения деревьев принятия решений. Среди них был выбран алгоритм «CART», который отличался незамысловатым алгоритмом и был реализован в виде модуля оптимизации маршрутов.

В-третьих, произошло ознакомление с видами тестирований и на основе доступных для тестов ресурсов был составлен план тестирования. Тестирование проводилось для разработанного функционала серверной части, с целью определения его качества.

Таким образом, результатом проделанной работы является разработанное приложение «Онлайн гид», которое нацелено на привлечение внимания граждан к культурной стороне города, а также привлечение и ознакомление туристов с историей города и помощь в ориентации по нему.

Серверная часть имеет многослойную архитектуру, что в дальнейшем позволит заменять её составляющие. Слой бизнес-логики реализован на языке C# и имеет интеграцию со слоем доступа к базам данных посредством

использования библиотеки Entity Framework. Доступ к бизнес-логике инкрементирован на уровне веб – API с RESTful архитектурой, что предоставляет разработчику унифицированный интерфейс, за счёт которого появляется возможность быстрого масштабирования. Веб – API взаимодействует с клиентской частью по протоколу HTTP, что гарантирует быструю обработку запросов сервером.

Клиентская часть реализована на платформе Xamarin, использующей язык C# для разработки, что позволило заметно уменьшить умственные и временные затраты на изучение стороннего языка для реализации.

Также на стороне клиентской части реализована интеграция с платформой Unity, которая реализует главную особенность разрабатываемой системы – объекты дополненной реальности. Для этого был использован плагин Vuforia, предоставляющий алгоритм для определения ровных поверхностей, пригодных для размещения объектов.

Поскольку разработанное приложение ориентировано только на город Тольятти, то его дальнейшее развитие может включать в себя адаптацию для других городов и добавление интеграций с платёжными системами, чтобы пользователи могли покупать билеты прямо на странице с описанием объекта.

Список используемой литературы

1. Албахари Д., Албахари Б. C# 7.0 Карманный справочник / Д. Албахари, Б. Албахари. – СпБ.: ООО «Альфа-книга», 2017. – 224 с.
2. Гвоздева, В.А. Основы построения автоматизированных информационных систем / В.А. Гвоздева, И.Ю. Лаврентьева. - М.: Форум, Инфра-М, 2016. – 320 с.
3. Джепикс, Ф. Язык программирования C# 7 и платформы .NET и .NET Core / Ф. Джепикс, Э. Троелсен. – СПб.: ООО «Диалектика», 2018. – 1328 с.
4. Джонс, М. Программирование искусственного интеллекта в приложениях / М. Джонс. – ДМК Пресс, 2006. – 312 с.
5. Евдокимов, П.В. C# на примерах / П.В. Евдокимов. – СПб.: Наука и Техника, 2019. – 320с.
6. Зегжда, Д.П. Основы безопасности информационных систем / Д.П. Зегжда, А.М. Ивашко. - М.: Горячая линия - Телеком, 2017. - 452 с.
7. Ипатов, Ю.В. Методологии и технологии системного проектирования информационных систем / Ю.В. Ипатов, Э.Р. Ипатова. - М.: Флинта, 2016. - 256 с.
8. Коцюба, И.Ю. Основы проектирования информационных систем. Учебное пособие / И.Ю. Коцюба, А.В. Чунаев, А.Н. Шиков. – СПб: Университет ИТМО, 2015. – 206 с.
9. Куликов, С.С. Тестирование программного обеспечения. Базовый курс / С.С. Куликов. – Минск: Четыре Четверти, 2017. – 312 с.
10. Неруш, Ю.М. Проектирование логистических систем: Учебник и практикум для бакалавриата и магистратуры / Ю.М. Неруш, С.А. Панов, А.Ю. Неруш. - Люберцы: Юрайт, 2016. – 422 с.
11. Норвиг, П. Искусственный интеллект. Современный подход / П. Норвиг, С. Рассел. – Вильямс, 2006 – 1408 с.

12. Палто, В.С. Менеджер событий на языке C# в Unity3D: разработка, оценка удобства использования и производительности / В.С. Палто, П.Ф. Фролов, С.А. Фролов. – Московский финансово-промышленный университет «Синергия», 2016 – 98 с.

13. Прайс, М. Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов / М.Дж. Прайс. – 3-е изд. – СПб.: Питер, 2018. – 640 с.

14. Пэкетт, Д. ASP.NET Core. Разработка приложений / Д. Пэкетт, С. Тиммс, Д. Чамберс. – СПб.: Питер, 2018. – 464 с.

15. Торн, А. Искусство создания сценариев в Unity / А. Торн. – ДМК пресс, 2015 – 362 с.

16. Фримен, А. ASP.NET Core MVC 2 с примерами на C# для профессионалов / А. Фримен. – 7-е изд. – Диалектика, 2019 – 1008 с.

17. Фримен, А. Entity Framework Core 2 для ASP.NET Core MVC для профессионалов / А. Фримен. – Диалектика, 2019 – 624 с.

18. Чистов, Д.В. Проектирование информационных систем : учебник и практикум для академического бакалавриата / Д. В. Чистов. — М. : Юрайт, 2016. — 258 с.

19. A Step by Step CART Decision Tree Example: сайт. — URL: https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/ (дата обращения: 17.02.2020). – Текст: электронный.

20. Albahari, B. C# 7.0 in a Nutshell / B. Albahari, J. Albahari. – O'Reilly Media, Inc. – 2018. – 1088 p.

21. Jin ,B. Designing Web APIs / B. Jin, Sahni S., Shevat A. – O'Reilly Media. – 2018. – 232 p.

22. Lauret, A. The Design of Web APIs / A. Lauret. – Manning Publications. – 2019. – 392 p.

23. Miles, R. C# Programming Yellow Book / R. Miles. – Department of Computer Science University of Hull. – 2015. – 408 p.

24. Nagel, C. Professional C# 7 and .NET Core 2.0 / C. Nagel. – John Wiles & Sons, Inc. – 2018. – 1440 p.

25. Petzold, C. Creating Mobile Apps with Xamarin.Forms. Cross-platform C# programming for iOS, Android, and Windows / C. Petzold – Microsoft Press. – 2016. – 438 p.

Приложение А

Реализация метода для регистрации нового пользователя

```
// POST: api/User
[HttpPost]
public async Task<PostResponse> Post([FromBody] Web.User user)
{
	try
	{
		var data = _mapper.Map<DB.User>(user);
		_manager.Set(data);

		return new PostResponse
		{
			Success = true,
			ResultDescription = "User added"
		};
	}
	catch (Exception ex)
	{
		_logger.LogError(ex, "Post user error");
		var result = new PostResponse
		{
			Success = false,
			ResultDescription = ex.Message
		};

		return result;
	}
}
```

Полная версия кода доступна по ссылке: https://bitbucket.org/xvenart/gqw_onlineguide/src/dev/