МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г. И. Радченко
«___»_____ 2021 г.

Разработка клиент-серверного приложения для мониторинга занятости парковочных мест

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____П. О. Шабуров
«___»_____2021 г.

Автор работы,
студент группы КЭ-405
_____М. Д. Суховей
«___»_____2021 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С. В. Сяськов
«___»_____2021 г.

Челябинск-2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г. И. Радченко
«___»_____2021 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Суховею Максиму Дмитриевичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Разработка клиент-серверного приложения для мониторинга занятости парковочных мест» утверждена приказом по университету от 26 апреля 2021 г. №714-13/12 (приложение №73)

2. **Срок сдачи студентом законченной работы:** 1 июня 2021 г.

3. **Исходные данные к работе:**

– техническое задание;
– документация по ASP.NET;
– документация по API Яндекс.Карт;
– синтаксис C#, JavaScript.

4. **Перечень подлежащих разработке вопросов:**

– рассмотрение существующих аналогов;
– определение инструментария для разработки;

- составление технического задания и определение требований;
- разработка схемы базы данных;
- реализация приложения и базы данных;
- тестирование и отладка разработанного клиент-серверного приложения.

5. **Дата выдачи задания:** 1 декабря 2020 г.

Руководитель работы _____/*П.О. Шабуров*/

Студент _____/*М. Д. Суховей*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2021	
Разработка модели, проектирование	01.04.2021	
Реализация системы	01.05.2021	
Тестирование, отладка, эксперименты	15.05.2021	
Компоновка текста работы и сдача на нормоконтроль	24.05.2021	
Подготовка презентации и доклада	30.05.2021	

Руководитель работы _____/*П.О. Шабуров*/

Студент _____/*М. Д. Суховей*/

АННОТАЦИЯ

М. Д. Суховей. Разработка клиент-серверного приложения для мониторинга занятости парковочных мест. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2021, 54 с., 19 ил., библиогр. список – 28 наим., 2 прил.

В рамках выпускной квалификационной работы осуществляется обзор существующих программных решений мониторинга занятости парковочных мест. Анализируются их преимущества и недостатки. На основе сравнительного анализа выбирается технологическое решение для разработки клиент-серверного приложения. Определяются требования и архитектура предлагаемого решения. Реализуется клиент-серверное приложение, дополняющее функционал аппаратной части системы (конечные устройства сбора информации) для мониторинга занятости парковочных мест. Производится тестирование работоспособности предлагаемого программного решения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Темой выпускной квалификационной работы является разработка клиент-серверного приложения для мониторинга занятости парковочных мест.

По данным аналитического агентства «АВТОСТАТ» по состоянию на январь 2021 года на одну тысячу жителей в Российской Федерации приходится 309 легковых машин - в среднем, каждый третий россиянин является владельцем автомобиля [1]. В условиях высокой автомобилизации городов перед водителем возникает проблема поиска свободного парковочного места вблизи интересующей локации. Особенно остро эта проблема ощутима в городах с численностью постоянного населения миллион и более, в которых и без того непростая транспортная ситуация отягощается участием в трафике автомобилей из соседних городов и регионов. Но даже при наличии достаточного объёма парковочного пространства в условиях повышенной транспортной плотности, без соответствующей визуализации найти «наугад» свободное парковочное место - задача непростая.

В настоящее время существующие решения обозначенной выше проблемы не имеют широкого прикладного применения и достаточной степени научной проработанности, в связи с чем, создание программного продукта, позволяющего упростить водителю процесс поиска свободного места на парковке, имеет важное практическое значение.

Целью выпускной квалификационной работы является создание клиент-серверного приложения, дополняющего функционал аппаратной части системы (конечные устройства сбора информации) для мониторинга занятости парковочных мест.

Для достижения обозначенной цели, необходимо решить следующие задачи:

1. Рассмотреть и изучить существующие аналоги.
2. Проанализировать преимущества и недостатки существующих аналогов.

3. Определить требования к создаваемой системе.
4. Разработать схему базы данных.
5. Реализовать сервер, клиентское приложение.
6. Провести тестирование клиент-серверного приложения с устройством сбора информации.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 ОБЗОР АНАЛОГОВ

Среди основных аналогов разрабатываемой системы можно отметить следующие решения:

- SPOT (Smart Parking Occupation Tracking) [2];
- «Умные парковки» (Интерсвязь) [3];
- Яндекс.Парковки [4].

Наиболее схожим по структуре с разрабатываемой системой является сервис SPOT. Камера, направленная на отслеживаемую парковку, передаёт видеопоток на вычислительное устройство, которое анализирует и обрабатывает поступающую на него информацию для последующей передачи серверу. Сервер, в свою очередь, собирает информацию с таких устройств и преобразует её для отображения на картах в мобильном приложении и веб-версии сервиса. Для отображения используются карты 2ГИС [5]. На картах пиктограммой отмечается парковка. При нажатии на эту пиктограмму во всплывающем окне в виде статичной картинки отображается подробная схема парковочного пространства и текущая занятость парковочных мест (рисунок 1).

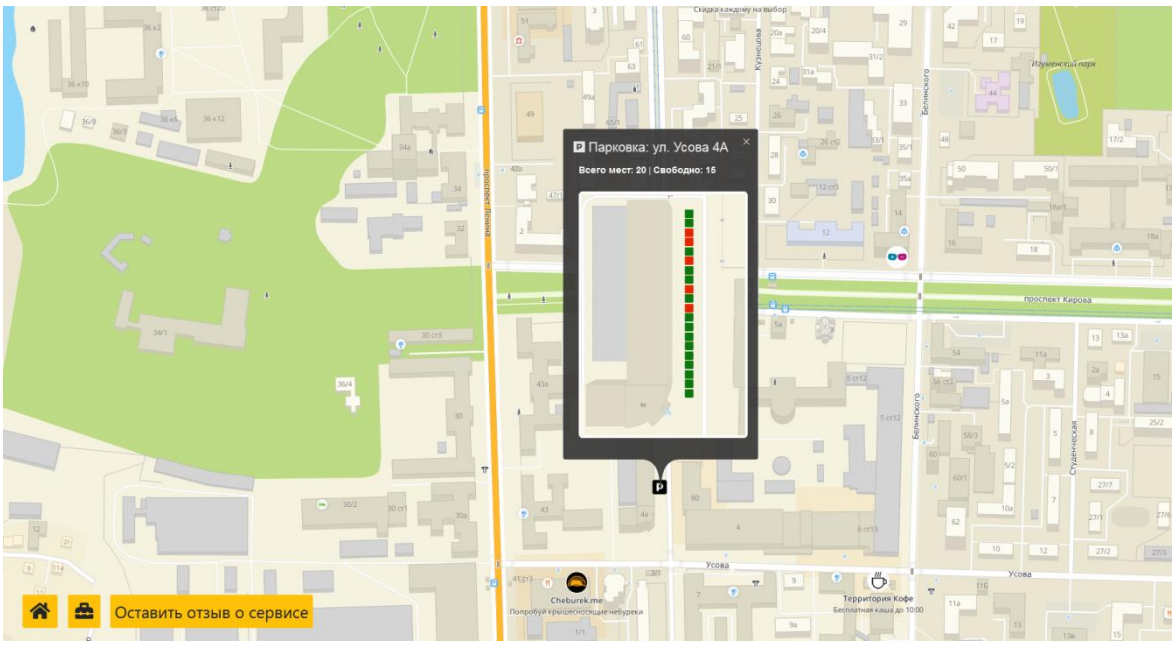


Рисунок 1 - Интерфейс сервиса SPOT

На момент написания выпускной квалификационной работы сервис работает в пилотном режиме: для мониторинга доступны два парковочных пространства в городе Томске, анонсирован релиз мобильного приложения.

Сервис «Умные парковки» был запущен компанией Интерсвязь в 2018 году. Для определения занятости парковочных мест используется изображение с камер видеонаблюдения, направленных на парковочное пространство. Нейросеть анализирует изображение, получаемое с камер, и определяет количество свободных мест. Далее информация передаётся на сервер, который отражает эти сведения на карте [6]. В отличие от рассмотренного ранее сервиса SPOT, «Умные парковки» не отображают подробную схему парковки. Вместо этого пользователю предлагается изображение с камер видеонаблюдения, на котором многоугольниками отмечены свободные места на парковке (рисунок 2).

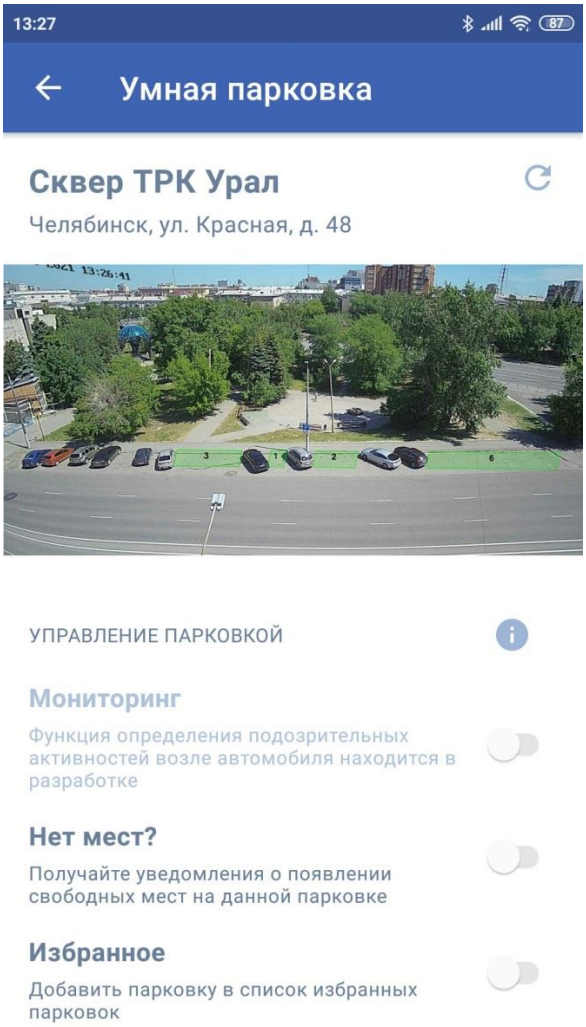


Рисунок 2 - Сервис "Умные парковки"

Среди преимуществ сервиса «Умные парковки» можно отметить наличие функции «Мониторинг», позволяющей отслеживать подозрительные действия около автомобиля. С помощью интерфейса приложения пользователь отмечает автомобиль, за которым необходимо следить, и нейросеть анализирует заданную область. В случае обнаружения подозрительных действий пользователю будет отправлено оповещение [7].

Сервис «Яндекс.Парковки» является косвенным аналогом, поскольку не позволяет отслеживать занятость парковочных мест, однако идейно схож с разрабатываемой системой (рисунок 3).

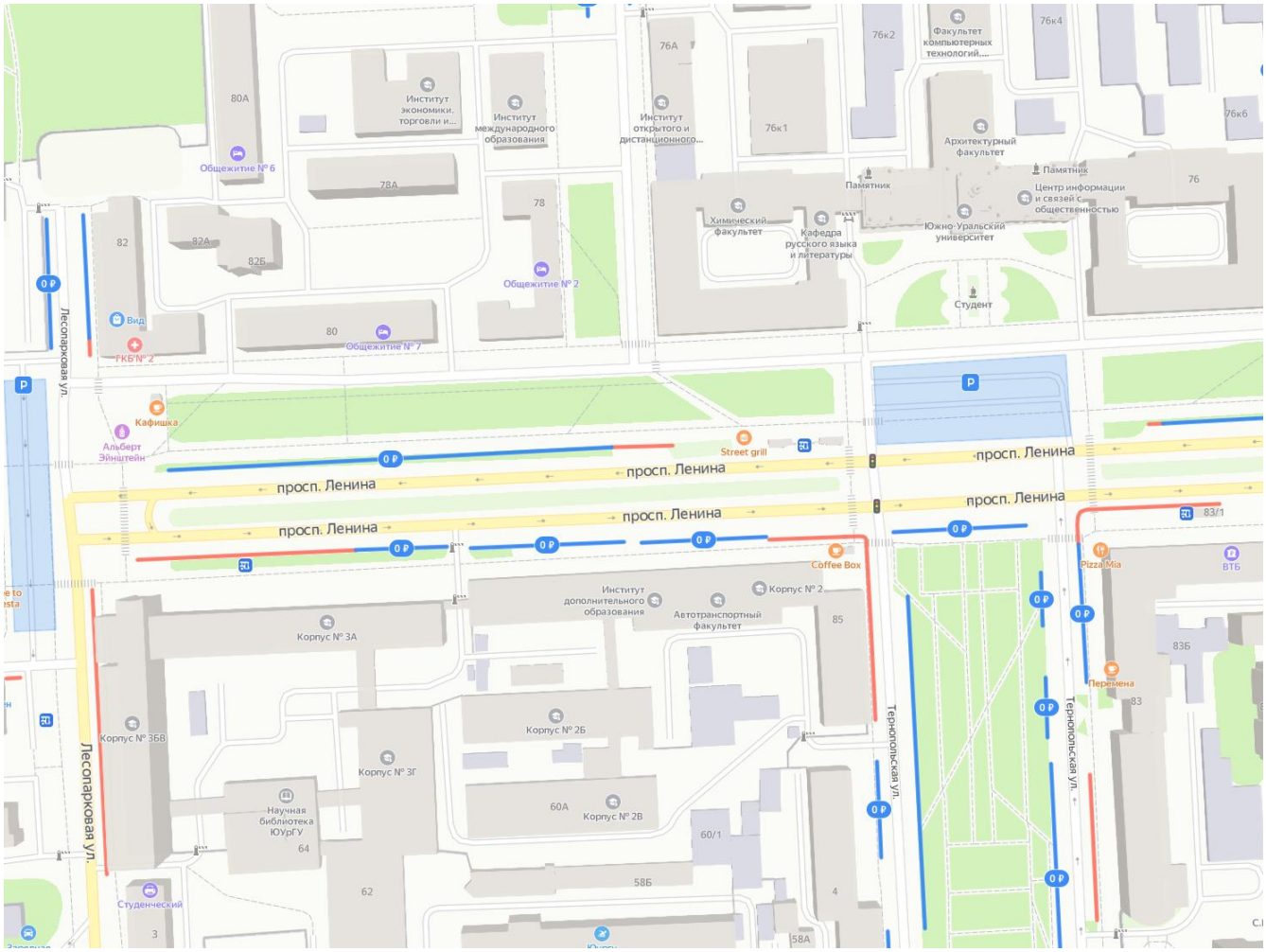


Рисунок 3 - Сервис "Яндекс.Парковки"

1.2 АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

1.2.1 ВЫБОР ПЛАТФОРМЫ РАЗРАБОТКИ

Laravel

Бесплатный фреймворк на основе языка программирования с открытым исходным кодом PHP, который был специально создан для веб-разработки. В основу фреймворка Laravel разработчики заложили шаблон проектирования MVC («Model-View-Controller»). Это облегчает веб-разработчику задачи развёртки и обслуживания создаваемого им приложения [8].

К достоинствам платформы Laravel можно отнести широкую распространённость языка PHP, на которой она основана, бесплатность, относительную простоту освоения и масштабируемость проектов.

Среди недостатков этого фреймворка можно выделить отсутствие строгой типизации и контроля типов данных языка PHP и более медленную работу в сравнении с другими фреймворками [9].

Node.js

Платформа предназначена для разработки масштабируемых веб-приложений. В ней используется событийно-ориентированная архитектура и асинхронное взаимодействие [10].

Среди достоинств платформы необходимо выделить возможность масштабирования, поддержку асинхронной модели работы, относительно высокую производительность и широкий выбор библиотек и инструментов.

К недостаткам можно отнести то, что для безошибочной реализации больших высоконагруженных проектов требуются глубокие познания в этой технологии, в основе Node.js лежит интерпретируемый язык программирования JavaScript, в котором, как и в PHP, нет строгой типизации, и существует риск нарушения работы вызовов, порождённый использованием асинхронной модели [11].

ASP.NET

Платформа разработана компанией Microsoft. Входит в состав фреймворка .NET Framework. Помимо прочего, поддерживает паттерн проектирования MVC (ASP.NET MVC Framework). Поддерживаются языки программирования, входящие в .NET Framework, в числе которых C#, Visual Basic .NET, C++/CLI и другие [12].

К достоинствам ASP.NET можно отнести развитую документацию и большое сообщество разработчиков на этой платформе, разработку и отладку в среде Visual Studio, наличие широкого выбора библиотек и инструментов для разработки, возможность масштабирования разработанного проекта, полный контроль над HTML-разметкой в ASP.NET MVC Framework [13].

Из недостатков можно выделить Windows-ориентированность средств разработки [14], продвинутые версии которых притом не являются бесплатными [15].

1.2.2 ВЫБОР СУБД

Microsoft SQL Server

Система управления реляционными базами данных, разработанная компанией Microsoft. Одно из самых популярных решений для управления базами данных на рынке. Для взаимодействия с базами данных, в частности, управления ими или выполнения запросов к ним, используется язык запросов Transact-SQL, созданный при сотрудничестве Microsoft и Sybase и являющийся расширением языка SQL [16].

Преимущества использования этой СУБД заключаются в развитой документации, наличии бесплатной версии, возможность гибкой настройки параметров управления базой данных и безопасности [17].

Недостатки тесно связаны с преимуществами MS SQL Server: для проектов, где требуются возможности, превышающие функционал бесплатной версии, возникает необходимость использования платной и достаточно дорогостоящей лицензии этой СУБД [17]. При отсутствии достаточного опыта работы гибкость и многочисленность настроек может вызвать трудности у разработчика [16].

PostgreSQL

Объектно-реляционная система управления базами данных. Поддерживает SQL и JSON для реляционных и нереляционных запросов, а также расширенные типы данных и функции для оптимизации производительности [18].

Среди преимуществ этой СУБД можно выделить бесплатность использования (распространяется под лицензией программного обеспечения с открытым кодом), возможность расширения базы данных и гибкость настройки [19].

Из недостатков можно выделить относительно низкую производительность и отсутствие некоторого функционала по сравнению с проприетарными СУБД [19].

Oracle Database

Как и PostgreSQL, Oracle Database является объектно-реляционной системой управления базами данных. Эта СУБД разработана компанией Oracle. Подобно MS SQL Server, имеет собственную надстройку над языком SQL – PL/SQL [20].

Преимущества Oracle Database заключаются в наличии бесплатной версии, портативности, относительно высокой производительности в больших базах данных [21].

Недостатки использования этой СУБД схожи с недостатками MS SQL Server: для больших проектов необходимо приобретать дорогостоящую продвинутую версию, у неопытного разработчика могут возникнуть трудности при настройке обширного количества параметров СУБД [21].

1.2.3 ВЫБОР КАРТОГРАФИЧЕСКОГО СЕРВИСА

2ГИС

Российский картографический и справочный сервис, созданный одноимённой компанией. Есть API (Application Programming Interface), для доступа к которому необходим ключ. Для некоммерческих, исследовательских и образовательных проектов доступ предоставляется бесплатно, но с некоторыми ограничениями. Для обхода этих ограничений необходимо приобретать коммерческий ключ [22].

Преимущества этого картографического сервиса заключаются в наличии возможности отобразить дорожную ситуацию на карту, высокой проработке и детализации самих карт (см. рисунок 1) и наличии справочника по API.

Из недостатков можно отметить отсутствие возможности отображения спутниковых снимков на карте [23].

Google Maps Platform

Картографический и справочный сервис, разработанный компанией Google. Карты можно бесплатно встраивать на сайт при наличии ключа API [24]. Для использования дополнительных функций, таких как построение маршрутов,

геокодинга, геолокации устройства, отображения панорам необходимо подключение платного тарифа. Каждая функция тарифицируется отдельно [25].

Из преимуществ можно выделить наличие спутниковых карт, сервиса отображения дорожных ситуаций и документации по API [24].

Среди недостатков стоит отметить низкую детализацию карт и наличие только англоязычной версии документации [24].

Яндекс.Карты

Картографический и справочный сервис от российской транснациональной компании Яндекс. Как и рассмотренные сервисы, Яндекс.Карты имеют API, доступ к которому осуществляется только с помощью ключа. Для некоммерческого и открытого использования ключ предоставляется бесплатно, но с ограничениями на использование (ограниченное количество запросов к геокодеру и запрет на использование для мониторинга транспорта) [26].

Среди преимуществ наличие подробной документации по API с примерами использования, наличие спутниковых карт и сервиса отображения дорожных ситуаций, а также высокая детализация карт [27].

Относительно 2ГИС и Google Maps, Яндекс.Карты не имеют существенных недостатков.

На рисунке 4 представлено сравнение уровня детализации Яндекс.Карт и Google Maps.

Рисунок 4 – Сравнение детализации Google Maps и Яндекс.Карт

1.3 ВЫВОД

На основе проведённого сравнительного анализа технологических решений для разработки клиент-серверного приложения из каждой группы был выбран наиболее подходящий элемент.

В качестве платформы разработки был выбран ASP.NET MVC Framework. Фреймворки Laravel и Node.js базируются на языках с динамической (JavaScript), либо нестрогой типизацией (PHP), в то время как ASP.NET, являясь расширением платформы .NET, работает с языками со строгой типизацией (C# и F#). И если для клиентской веб-страницы можно применять динамическую типизацию, то для разработки сервера предпочтительнее использовать язык со статической типизацией, чтобы избежать возможных ошибок во время выполнения (Runtime error), связанных, например, с некорректным приведением типов. Возможно использовать Node.js в сочетании со строго типизированным языком TypeScript, однако в этом случае необходимо совершить ряд действий, чтобы подключить поддержку этого языка в проекте Node. Исходя из этого, наиболее рациональным будет создание проекта именно в ASP.NET MVC Framework.

Для создания и управления базой данных была выбрана СУБД Microsoft SQL Server. Для разрабатываемого клиент-серверного приложения не требуется функционал платных версий этой СУБД – весь требуемый функционал можно реализовать в бесплатной версии. В условиях разработки на платформе ASP.NET, в среде ОС Windows, наиболее удобным будет использование именно Microsoft SQL Server.

Для отображения парковок в клиентском приложении на интерактивной карте был выбран сервис Яндекс.Карты. По сравнению с картами 2ГИС, карты Яндекс имеют возможность отображения спутниковых снимков при равном уровне детализации и точности самих карт. По сравнению с сервисом Google Maps, Яндекс.Карты имеют русскоязычную документацию по API с подробными примерами, а также более высокую детализацию карт.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 ТРЕБОВАНИЯ К СИСТЕМЕ В ЦЕЛОМ

1. Наличие серверной части системы, позволяющей работать с устройствами сбора информации.
2. Наличие базы данных для хранения информации о парковках и парковочных местах.
3. Наличие клиентского приложения с простым и понятным интерфейсом.
4. Наличие методов и средств для взаимодействия клиентского приложения с сервером и базой данных.

Реализовать подсистемы:

1. Модуль приёма информации от конечных устройств.
2. Модуль предоставления информации из базы данных клиентскому приложению.

2.2 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Требования к серверной части:

В серверной части должна быть реализована база данных, способная сохранять и выдавать информацию о парковочных местах, получаемую от устройств сбора информации. Должен быть реализован метод для взаимодействия с устройствами сбора информации, принимающий в качестве аргумента JSON-строку с идентификатором парковки и идентификаторами парковочных мест, состояние которых изменилось.

Должно быть реализовано взаимодействие с базой данных. Информация о парковочных местах должна иметь определённую структуру, позволяющую выводить впоследствии эту информацию на интерактивные карты компании Яндекс.

Требования к взаимодействию клиентской и серверной частей:

Взаимодействие между клиентской и серверной частью должно осуществляться посредством HTTP-запросов.

Требования к клиентской части:

Клиентская часть должна быть реализована в виде веб-приложения, запускаемого в браузере, и представлена интерактивной картой с расположенными на ней пиктограммами парковок. Парковки подразделяются на парковочные места. На интерактивной карте должна отображаться схема парковки с выделением отдельных мест. Каждое место должно иметь интуитивно понятное обозначение, в зависимости от того занято оно или свободно; предназначено для инвалидов или нет.

Функциональные требования к подсистеме «Модуль приёма информации от конечных устройств»:

Входящая информация должна проверяться на соответствие заданной структуре JSON-объекта (структура будет уточнена на этапе рабочего проектирования), во избежание записи некорректной информации. Если полученные данные корректны, они записываются в базу данных. В противном случае – сервер возвращает сообщение об ошибке, и запись не происходит.

Функциональные требования к подсистеме «Модуль предоставления информации из базы данных клиентскому приложению»:

На основе параметров запроса от клиентского приложения модуль должен выбрать из базы данных требуемую информацию и вернуть её клиенту.

2.3 НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1. Клиентское веб-приложение должно работать во всех популярных на момент разработки браузерах: Mozilla Firefox (не ниже версии 58.0.2), Opera (не ниже версии 13), Google Chrome (не ниже версии 64.0.3282), Microsoft Edge (не ниже версии 25.10586), а также в их мобильных версиях.

2. Пользователю, работающему с клиентским приложением, должен быть предоставлен непрерывный доступ к приложению.

3. Сервер должен работать непрерывно. Допускаются только запланированные отключения.

Требования к объёму данных:

Объём данных напрямую зависит от количества работающих конечных устройств и не может быть точно определён до ввода в эксплуатацию реальной системы.

Требования к пользователям:

Для использования клиентского приложения допускается любой пользователь, имеющий доступ к сети Интернет.

Требования к временным характеристикам:

Время передачи информации между подсистемами не должно превышать 5 секунд. Худшее время выдачи результата запроса клиентскому приложению не должно превышать 10 секунд. Время обновления информации о занятости парковочного места не должно превышать 1 минуты.

3 ПРОЕКТИРОВАНИЕ

3.1 АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

Разрабатываемая система строится на основании архитектуры «клиент-сервер» (рисунок 5). Для взаимодействия сервера и базы данных используется объектно-ориентированная технология доступа к данным ADO.NET Entity Framework.

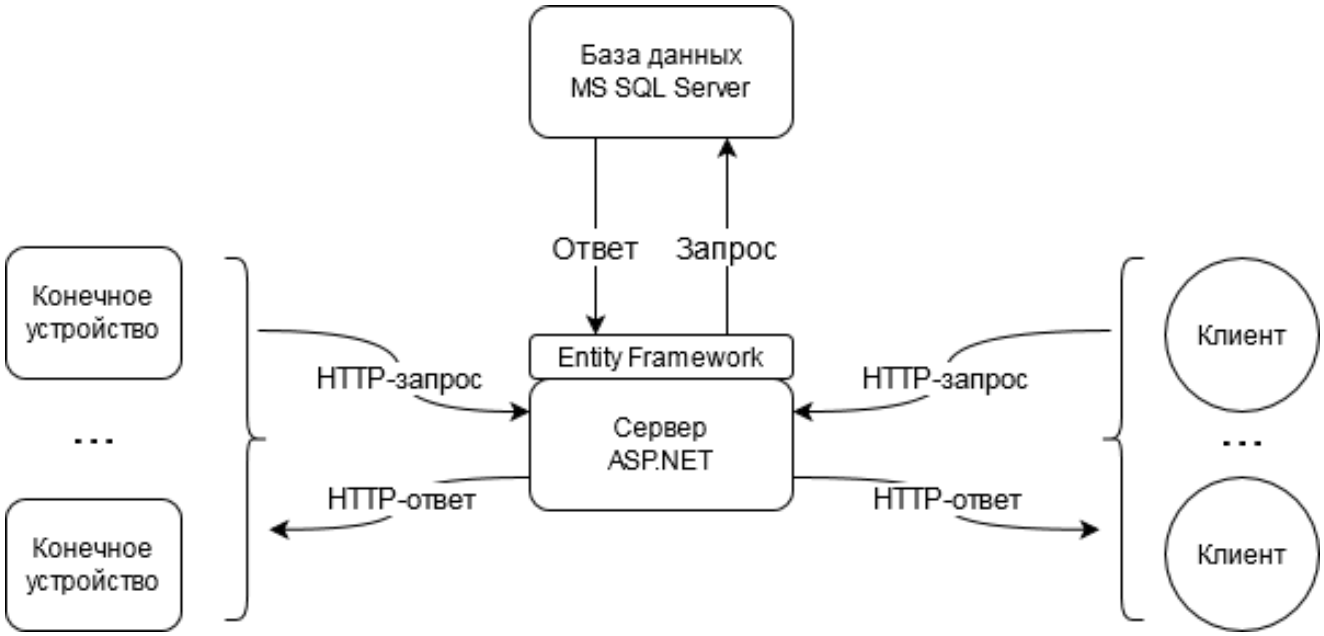


Рисунок 5 – Архитектура разрабатываемой системы

При разработке применяется паттерн проектирования MVC («Model-View-Controller»), который позволяет разделять бизнес-логику, графический интерфейс и данные (рисунок 6).

Пользователь работает с представлением. Во время работы он может инициировать возникновение события, которое будет влиять на контроллер. В соответствии с заложенной логикой, контроллер обрабатывает событие и, либо вносит изменения в модель данных, либо определяет другое представление.

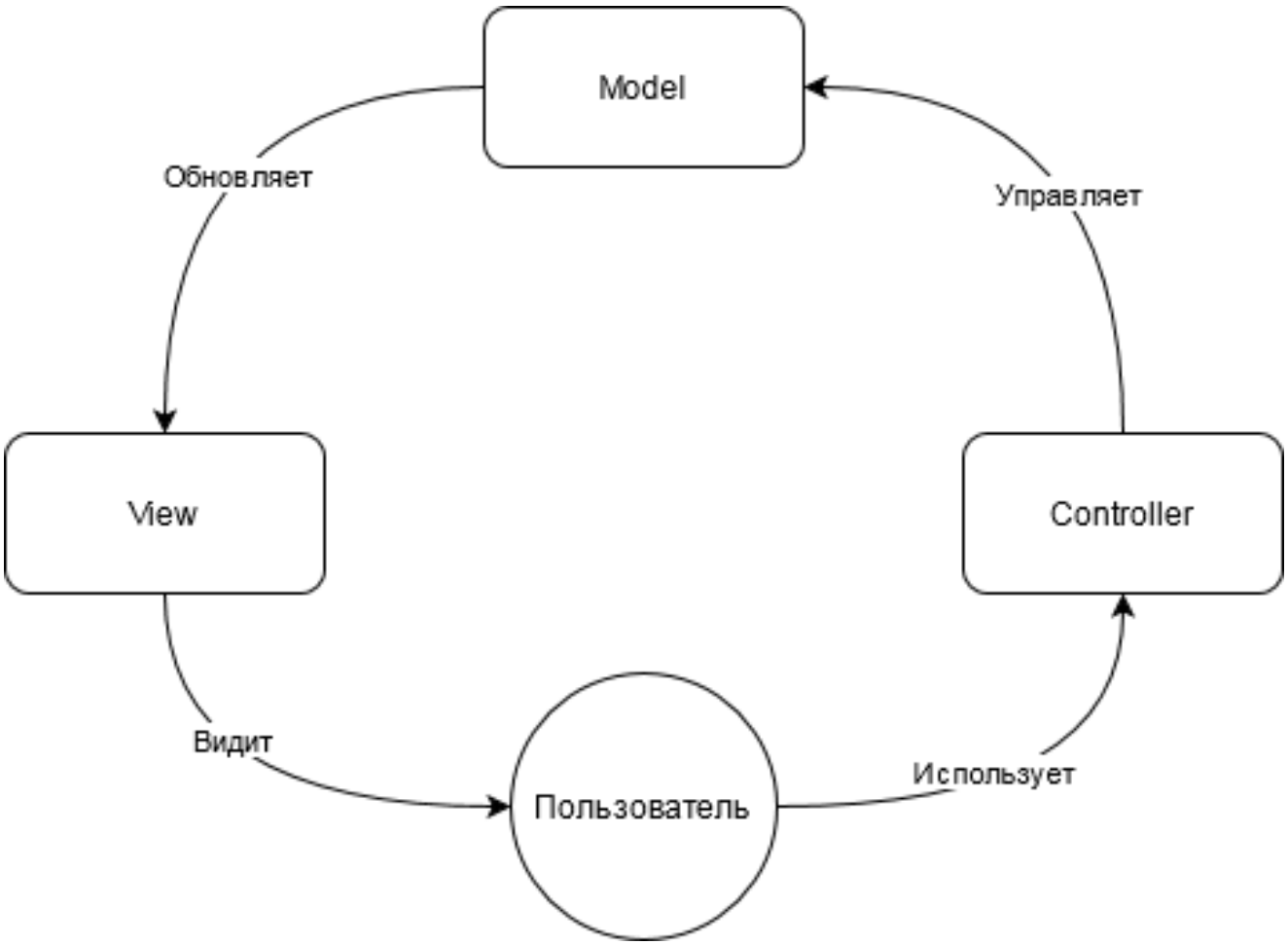


Рисунок 6 – Схема шаблона проектирования MVC

С использованием Entity Framework была получена модель данных для MVC (приложение Б, листинги Б.3–Б.5), основанная на разработанной схеме базы данных, описание которой представлено в пункте 3.2.

Контроллер Home представляет собой реализацию модуля предоставления информации из базы данных клиентскому приложению. В методе по умолчанию Index выполняется выборка всех парковок, информация о которых хранится в соответствующей таблице базы данных. Полученная информация передаётся в представление, которое, в свою очередь, передаётся пользователю и отображается в виде веб-страницы.

Во время работы с представлением пользователь может инициировать событие, выбрав на карте метку с определённой парковкой. Это событие вызовет метод GetPlaces контроллера Home (полученный исходный код в листинге Б.1 приложения Б). В качестве параметра методу передаётся идентификатор

парковки, на основании которого в метод будет сделан запрос к базе данных. В ответ на запрос из базы данных будут возвращены все парковочные места, привязанные к заданной парковке. После чего представление будет изменено – полученные данные отобразятся на интерактивной карте.

В качестве реализации модуля приёма информации от конечных устройств был создан контроллер Device (полученный исходный код в листинге Б.2 приложения Б). В нём был определён метод UpdateLot, в который передаётся JSON-строка с информацией о парковке и коллекцией парковочных мест, состояние которых необходимо обновить. В методе строка обрабатывается и на основании информации, представленной в ней, вносятся изменения в базу данных. В зависимости от результата выполнения операции сервер возвращает соответствующее сообщение на устройство, которое инициировало обновление информации.

3.2 ОПИСАНИЕ ДАННЫХ

Для работы системы необходима база данных, в которой будет храниться информация о парковках и парковочных местах, входящих в них. В результате анализа объектной области была получена следующая схема данных (рисунок 7).

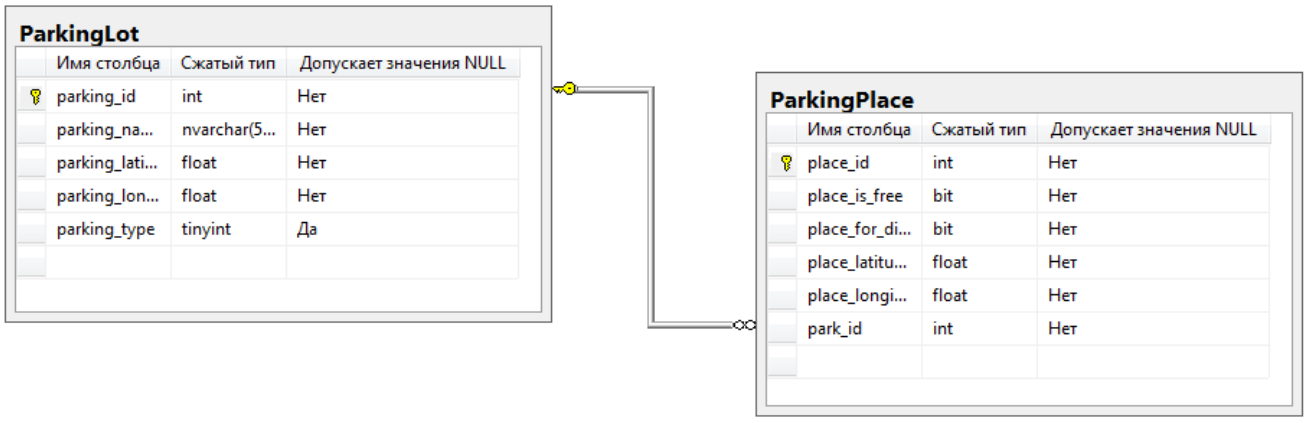


Рисунок 7 – Схема базы данных

ParkingLot – таблица предназначена для хранения информации о парковках. Содержит поля:

- parking_id – int – идентификатор парковки, первичный ключ;

- parking_name – nvarchar(50) – название парковки;

- parking_latitude – float – координата широты парковки;

- parking_longitude – float – координата долготы парковки;

- parking_type – tinyint – тип парковки.

ParkingPlace – таблица предназначена для хранения информации о парковочных местах.

Содержит поля:

- place_id – int – идентификатор парковочного места, первичный ключ;

- place_is_free – bit – занятость места;

- place_for_disabled – bit – место для инвалидов;

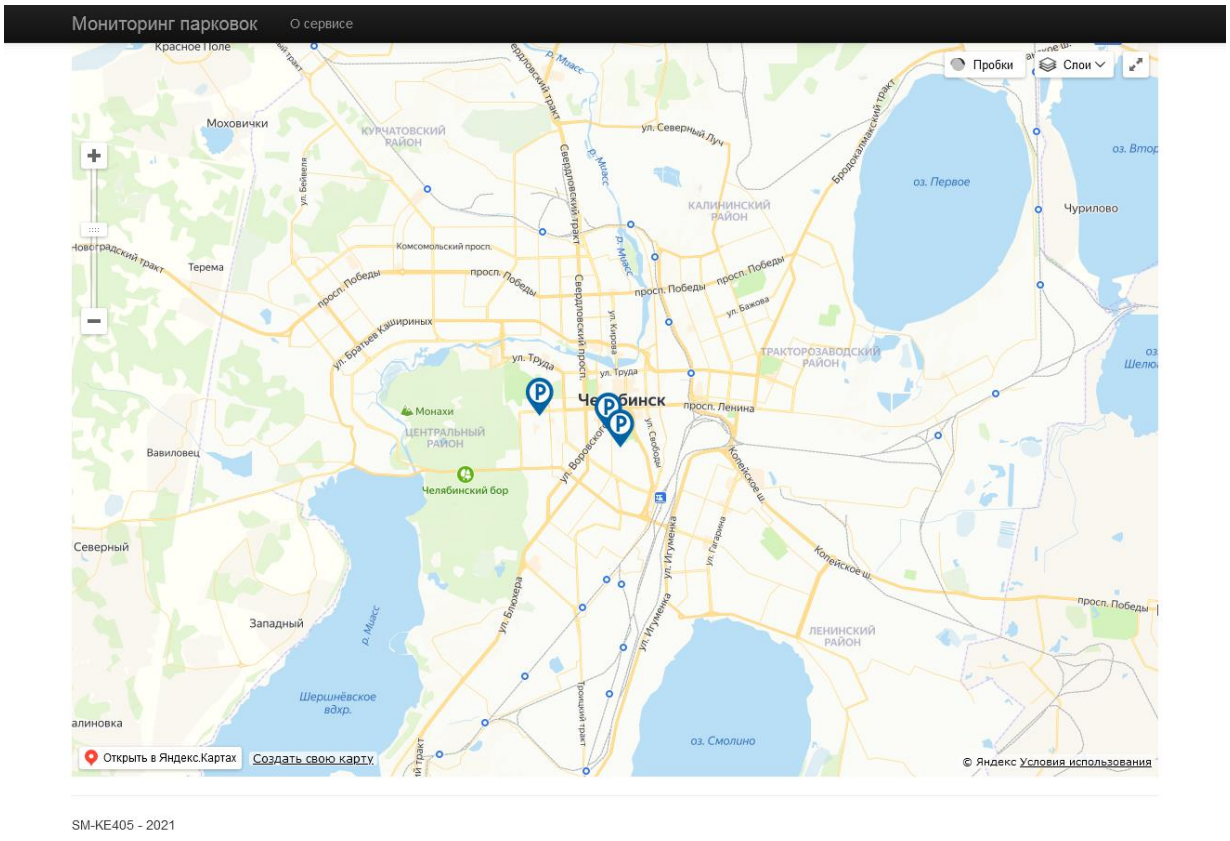
- place_latitude – float – координата широты парковочного места;

- place_longitude – float – координата долготы парковочного места;

- park_id – int – идентификатор парковки к которой принадлежит парковочное место, внешний ключ.

4 РЕАЛИЗАЦИЯ

Клиентское приложение представляет собой веб-сайт с двумя страницами, которые встраиваются в основной макет (исходный код в листинге А.3 приложения А): главной (исходный код в листинге А.1 приложения А) и информационной «О сервисе» (исходный код в листинге А.2 приложения А). На главной странице отображена интерактивная карта. При загрузке страницы, из базы данных запрашивается список парковок, присутствующих в системе. На основе полученной из базы данных информации на интерактивной карте расставляются метки соответствующих парковок (рисунок 8).



SM-KE405 - 2021

Рисунок 8 – Главная страница с интерактивной картой

При активации метки парковки нажатием - на карте будет отображена схема парковки (рисунок 9). Для максимального удобства восприятия информация визуализирована - представлена графически. Каждое парковочное место обозначается кругом. Его цвет зависит от статуса "занято" или "свободно" (красный или зелёный цвет, соответственно), "обычное" или "для инвалидов"

(синий цвет). Во всплывающем окне отобразится адрес парковки, текущее количество свободных мест и её тип (перпендикулярная, параллельная, под углом).

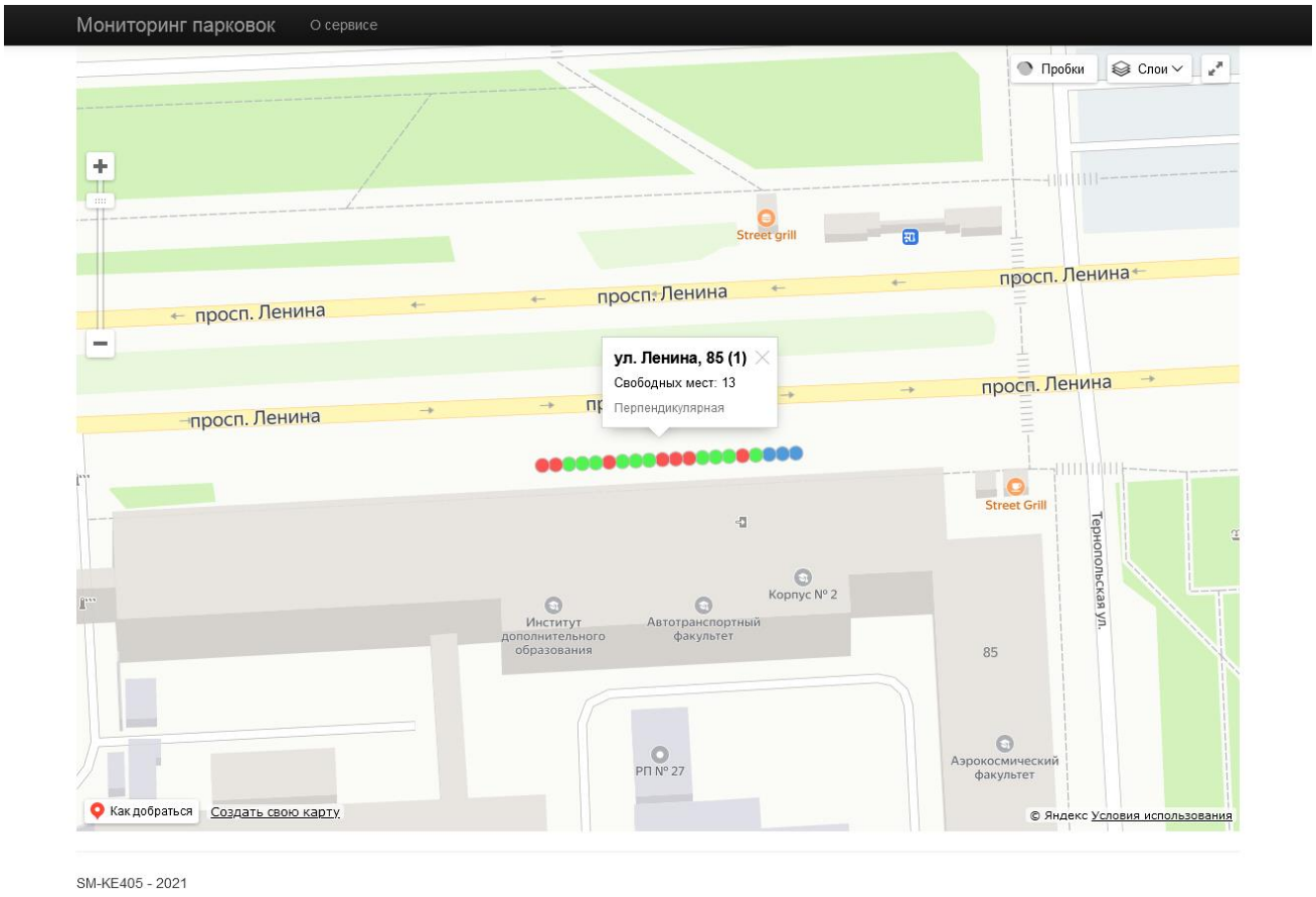


Рисунок 9 – Отображение схемы парковки и текущей занятости мест

При уменьшении масштаба карты всплывающее окно метки закрывается, и парковочные места удаляются с карты (рисунок 10).

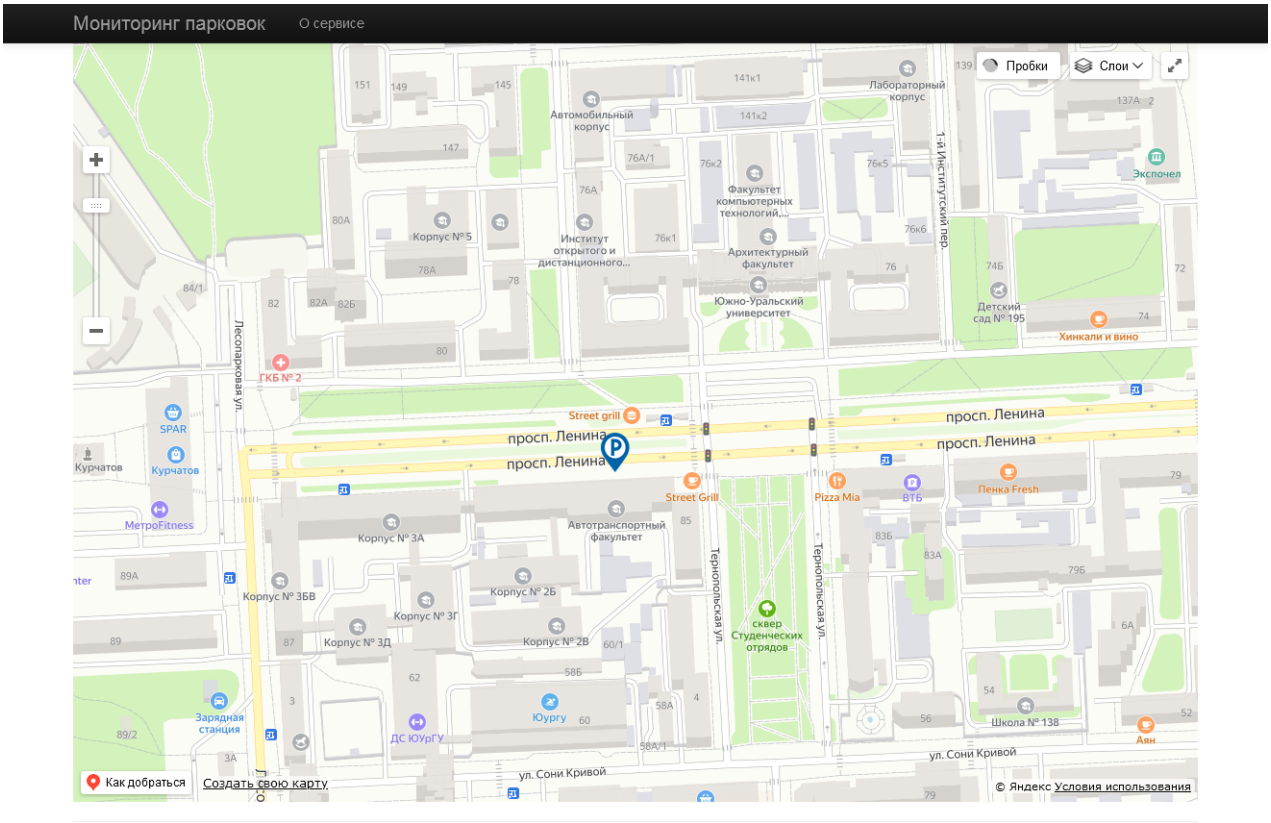


Рисунок 10 – Карта после уменьшения масштаба

Помимо прочего, есть возможность использовать встроенный сервис отображения дорожной ситуации – Яндекс.Пробки, изменить тип карты на «Гибрид» (совмещение схематической карты и спутниковых снимков) и раскрыть интерактивную карту на всю страницу (рисунок 11).

Рисунок 11 – Раскрытая на всю страницу карта с отображением пробок

5 ТЕСТИРОВАНИЕ

5.1 МЕТОДОЛОГИЯ ТЕСТИРОВАНИЯ

Особенность тестирования клиент-серверного приложения заключается в том, что необходимо проверять корректность работы не только клиентской части, но и сервера и базы данных, а также взаимодействия между этими элементами.

Необходимо проверить корректность отображения клиентских страниц. Должны отображаться все элементы, которые были прописаны в коде страницы при разработке. Поведение элементов должно быть прогнозируемым. При активации элементов управления нажатием должны выполняться именно те действия, которые ожидаются от этих элементов. Кроме того, необходимо проверить корректное отображение веб-страницы на мобильных устройствах.

Для проверки корректности взаимодействия клиента и сервера были использованы встроенные инструменты для веб-разработки браузера Mozilla Firefox (версия 88.0.1), в частности, инструмент для анализа сетевых запросов.

Правильная работа сервера заключается в отсутствии непредвиденных исключений и ошибок. Запросы должны обрабатываться в соответствии с логикой работы, прописанной в ходе разработки.

Работоспособность метода для обновления состояния парковки на основе JSON-строки от конечного устройства проверяется путём передачи строки состояния по адресу метода UpdateLot на сервере. При корректной работе возвращается HTML-страница с сообщением об успешном обновлении состояния указанной парковки, изменяются значения в базе данных и, соответственно, изменяется вывод в клиентском приложении.

5.2 ПРОВЕДЕНИЕ ПРОЦЕДУРЫ ТЕСТИРОВАНИЯ

Для проведения тестирования разработанная система была размещена на хостинге SmarterASP.NET [28]. Хостинг предназначен непосредственно для проектов на платформе ASP.NET и поддерживает базы данных MSSQL. Простой

аккаунт позволяет пользоваться хостингом бесплатно в течение 60 дней после регистрации.

После создания аккаунта проект с клиент-серверным приложением был загружен на хостинг. Была подключена созданная ранее база данных. В проекте были изменены строки для подключения к базе данных (приложение Б, листинг Б.8), поскольку ранее использовался локальный хост.

Для приложения был выделен веб-адрес: http://singularity172-001-site1.dtempurl.com/. При переходе по нему открывается главная страница сайта с картой. В версии для персонального компьютера все элементы отображаются корректно. Проверка корректности отображения мобильной версии клиентского приложения (рисунок 12) произведена на мобильном устройстве под управлением операционной системы Android (версия 6.0.1) при помощи браузера Chrome (версия 91.0.4472.88).

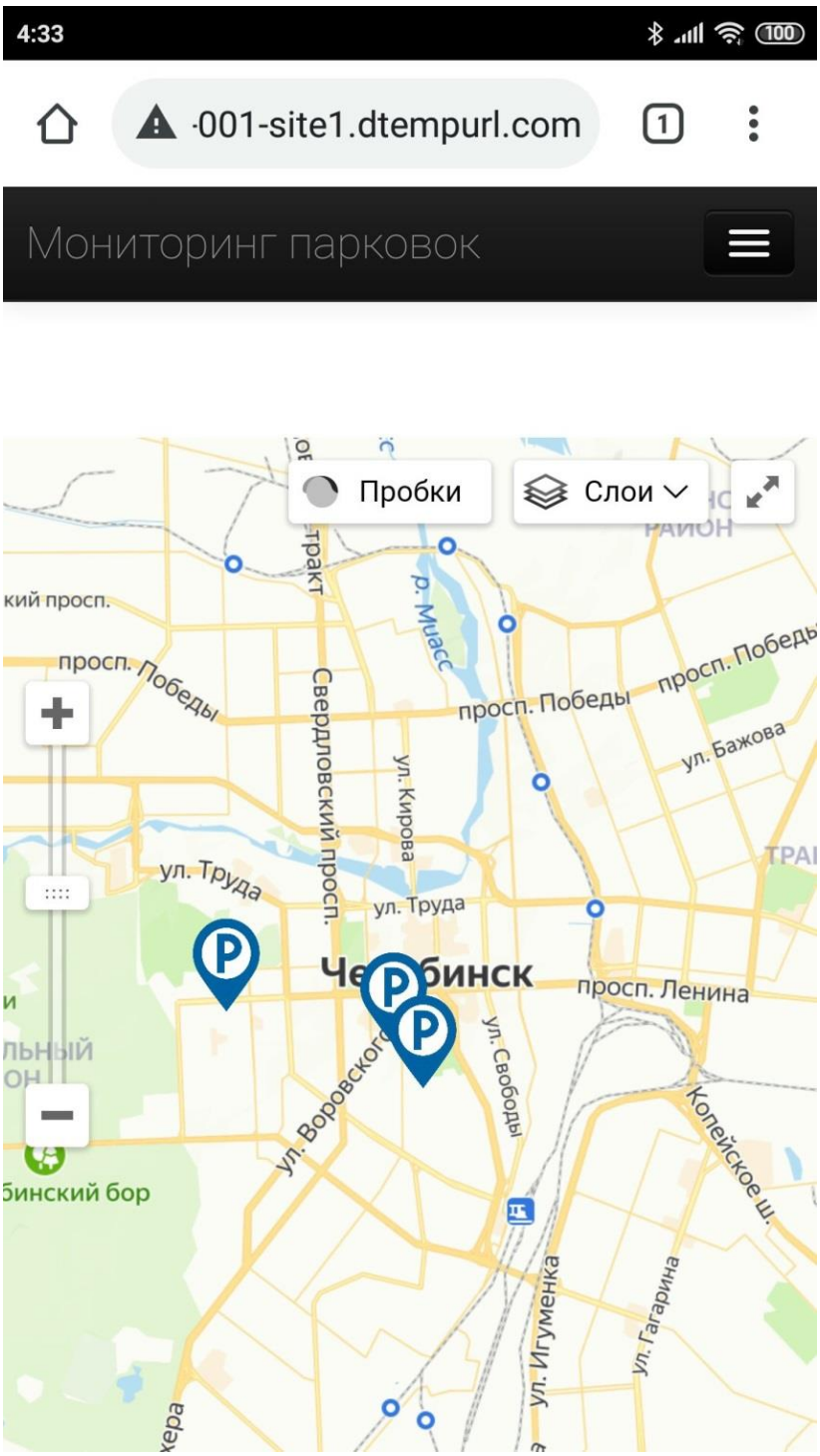


Рисунок 12 – Проверка корректности отображения мобильной версии клиентского приложения

Оценка корректности взаимодействия клиентского приложения и сервера осуществляется с помощью анализатора сетевых запросов. При открытии страницы с клиентским приложением все запросы к серверу выполняются корректно (рисунок 13).

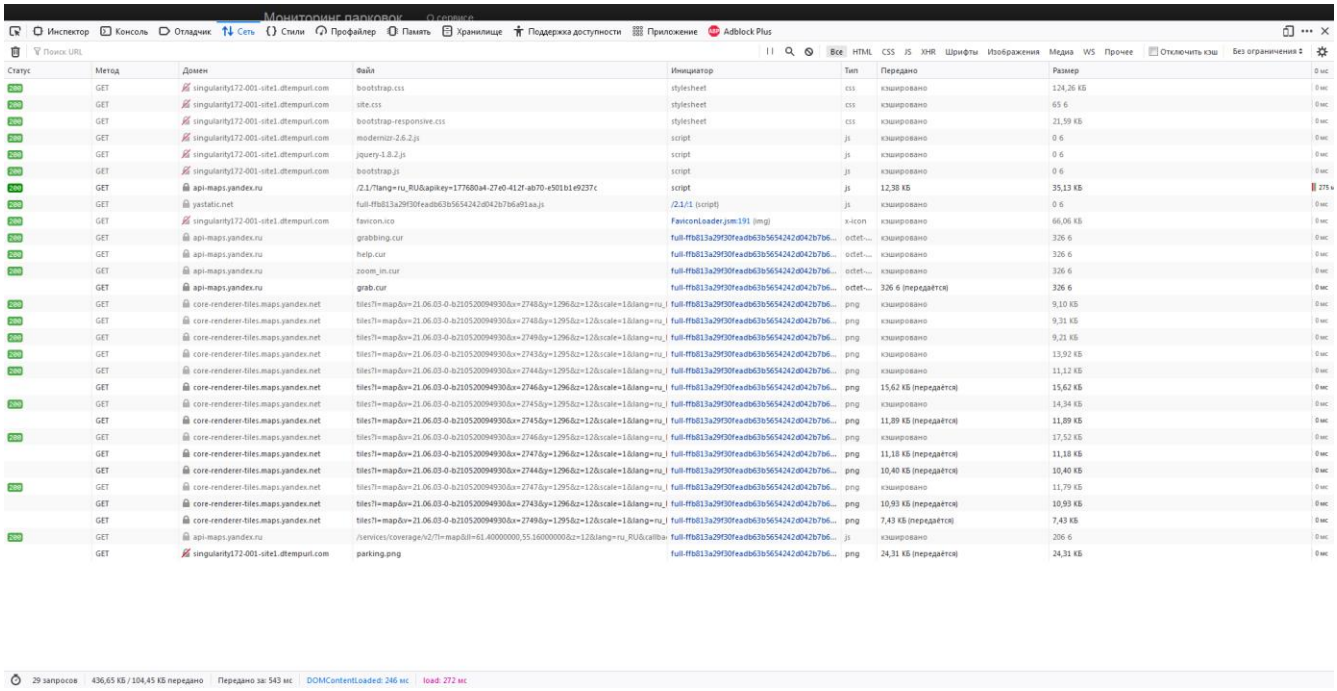


Рисунок 13 – Сетевые запросы при обращении к странице

Далее выполнена проверка корректности работы метода GetPlaces, возвращающего из базы данных парковочные места, в соответствии с выбранной парковкой на карте.

При активации метки парковки нажатием происходит вызов метода GetPlaces. В качестве параметра для запроса передаётся идентификатор выбранной парковки. В ответе должен вернуться JSON-объект со всеми парковочными местами соответствующей парковки. Полученный ответ должен быть обработан клиентом и выведен на карту.

Метод GetPlaces функционирует безошибочно. Из базы данных были выбраны парковочные места для парковки с идентификатором «2» и корректно отображены на интерактивной карте (рисунок 14).

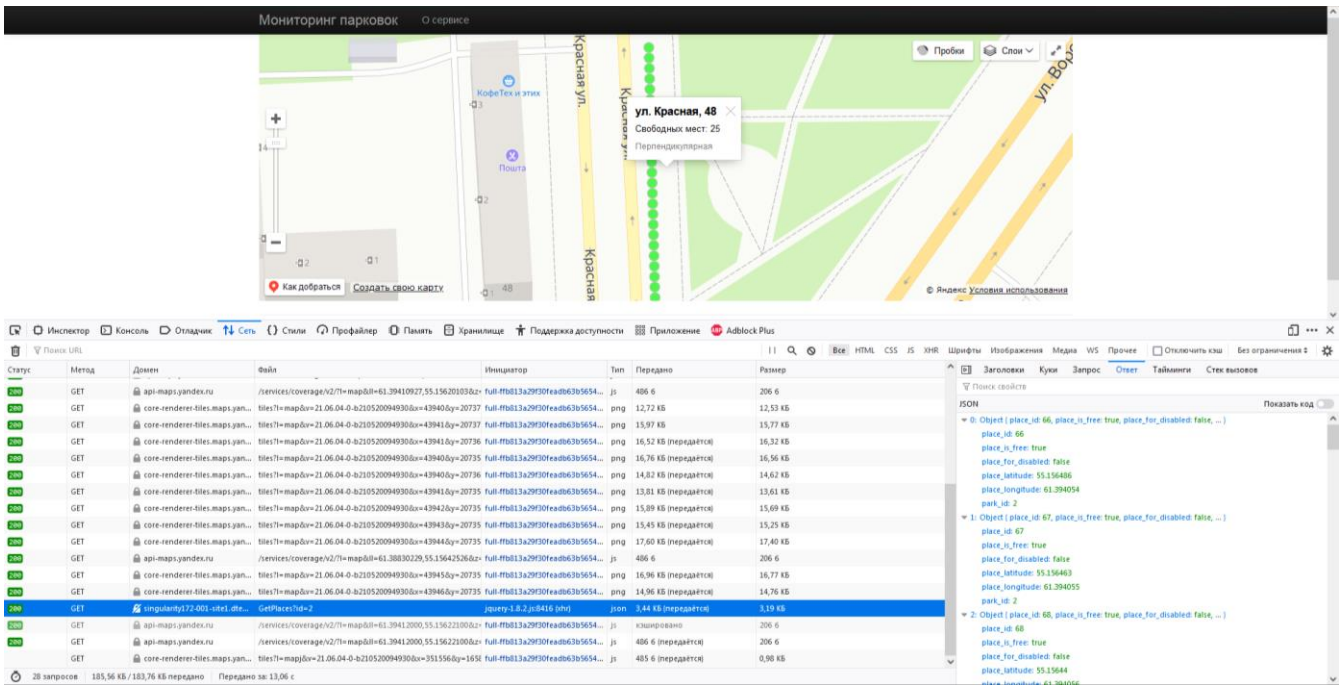


Рисунок 14 – Тестирование метода GetPlaces

Также протестирована работоспособность метода UpdateLot, принимающего от конечных устройств JSON-строку с парковочными местами, состояние которых изменилось, и вносящего изменения в базу данных. JSON-строка основана на минимизированных моделях данных парковок и парковочных мест (исходный код моделей в листингах Б.6, Б.7 приложения Б).

Для этого к серверу был подключен прототип конечного устройства, поддерживающий отслеживание состояния двух парковочных мест. Ему был присвоен идентификатор парковки «3» и два парковочных места с идентификаторами «95» и «96». В исходном состоянии все парковочные места свободны (рисунки 15, 16).

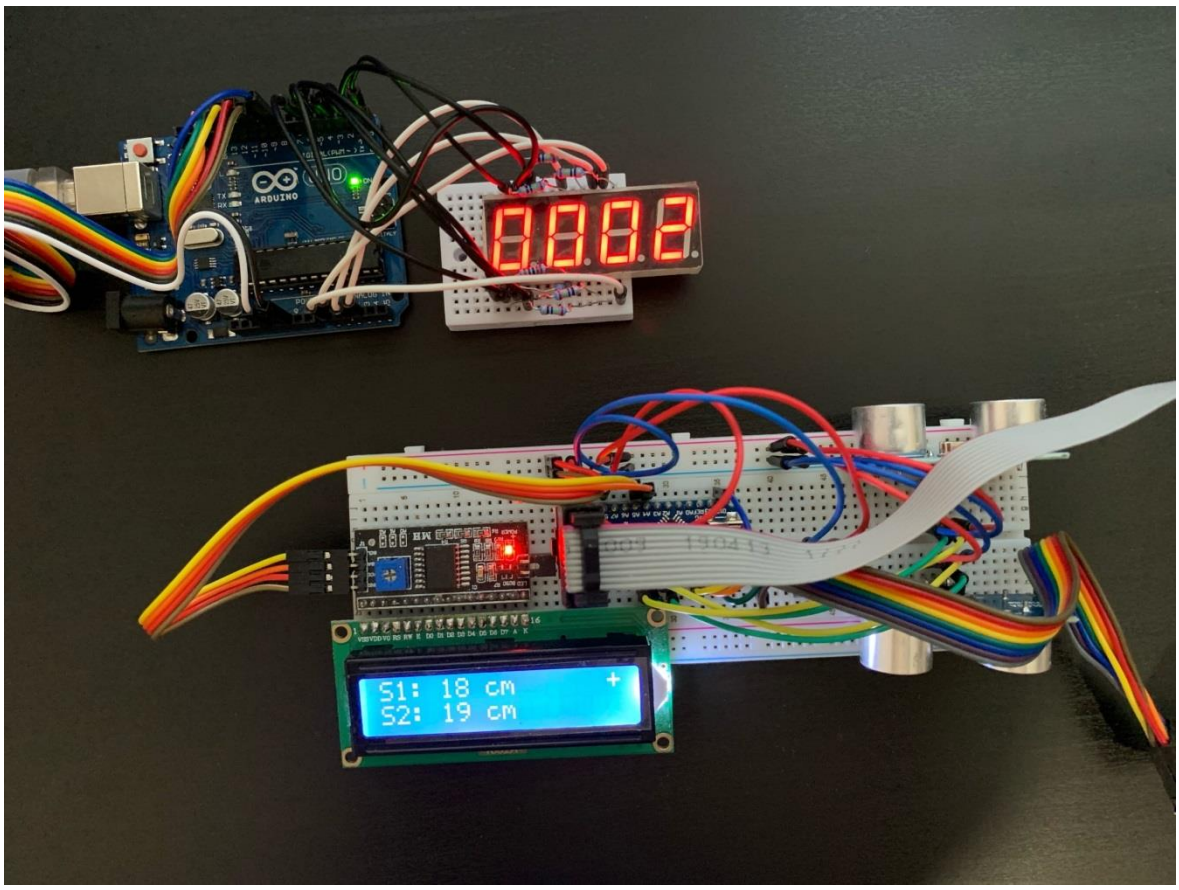


Рисунок 15 – Прототип конечного устройства в исходном состоянии

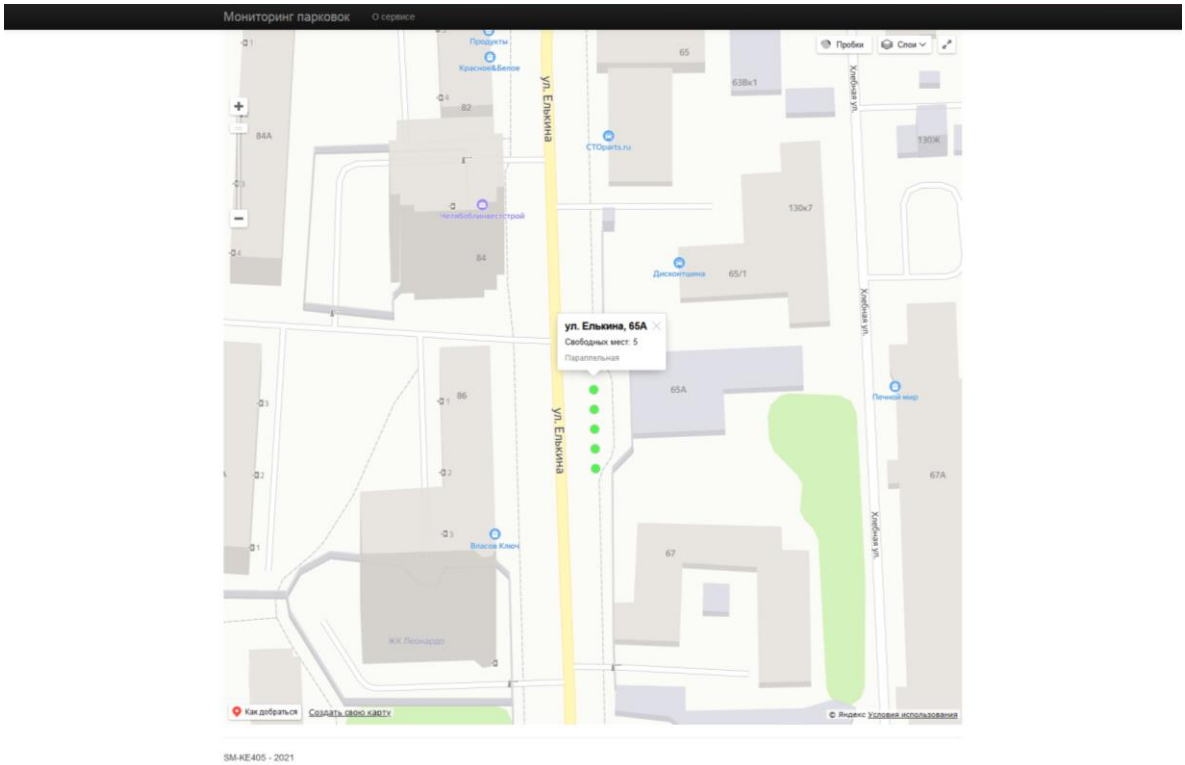


Рисунок 16 – Отображение на карте занятости парковочных мест

Далее была сымитирована парковка автомобиля, вызвавшая изменение состояния парковочного места (рисунок 17).

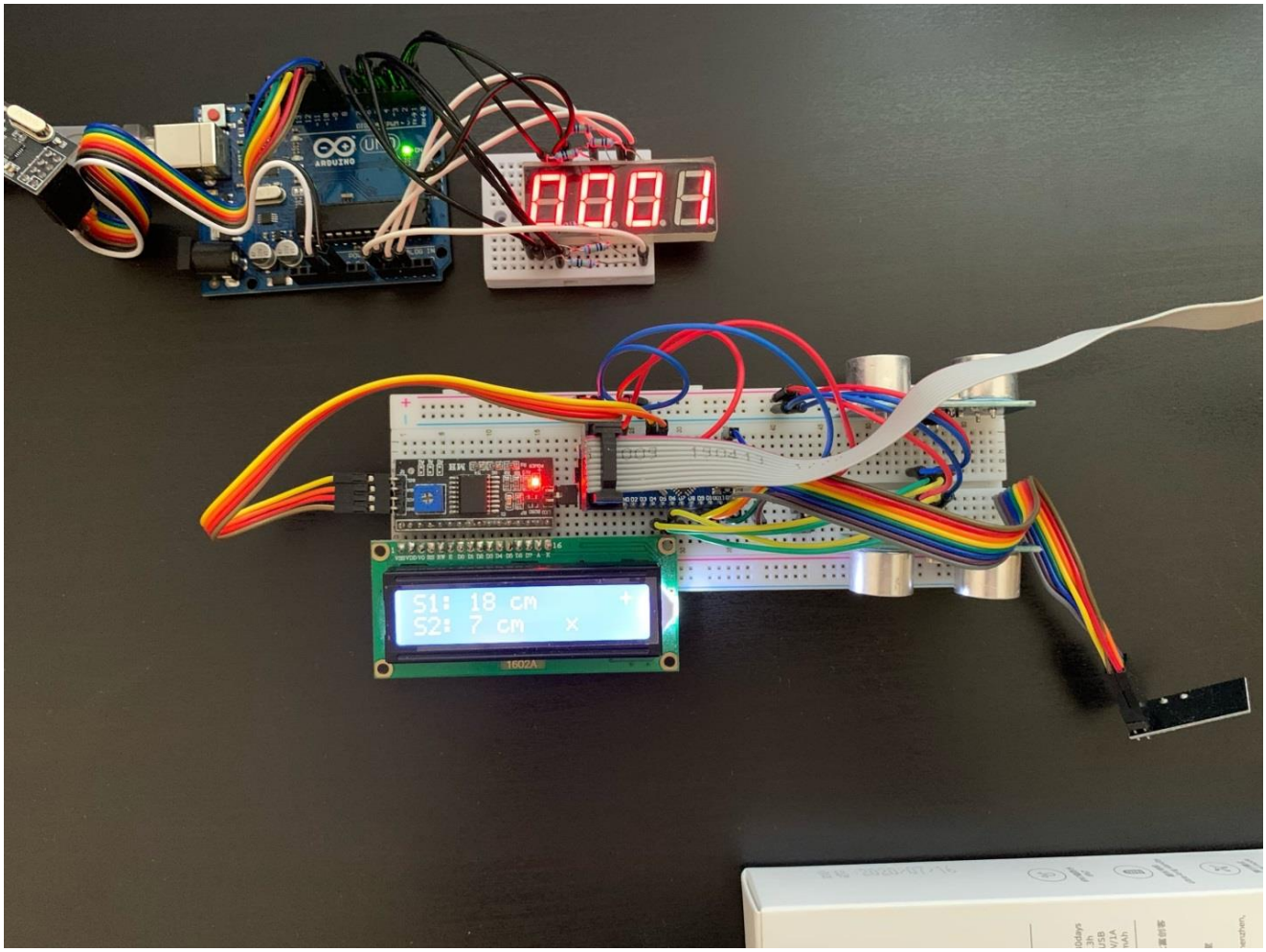


Рисунок 17 – Имитация парковки автомобиля

Конечное устройство передало JSON-строку с информацией о парковочном месте, состояние которого изменилось. Это было записано в лог-файл устройства (рисунок 18).

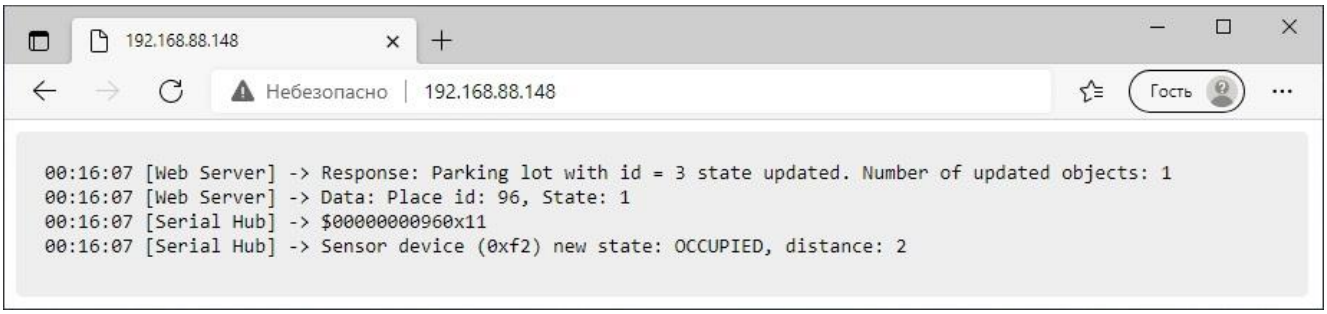


Рисунок 18 – Передача данных о состоянии парковочного места и ответ сервера

После обновления состояния сервер направил устройству ответ с сообщением об успешном завершении операции. На интерактивной карте отобразилось изменения состояния парковочного места с идентификатором «96» (рисунок 19).

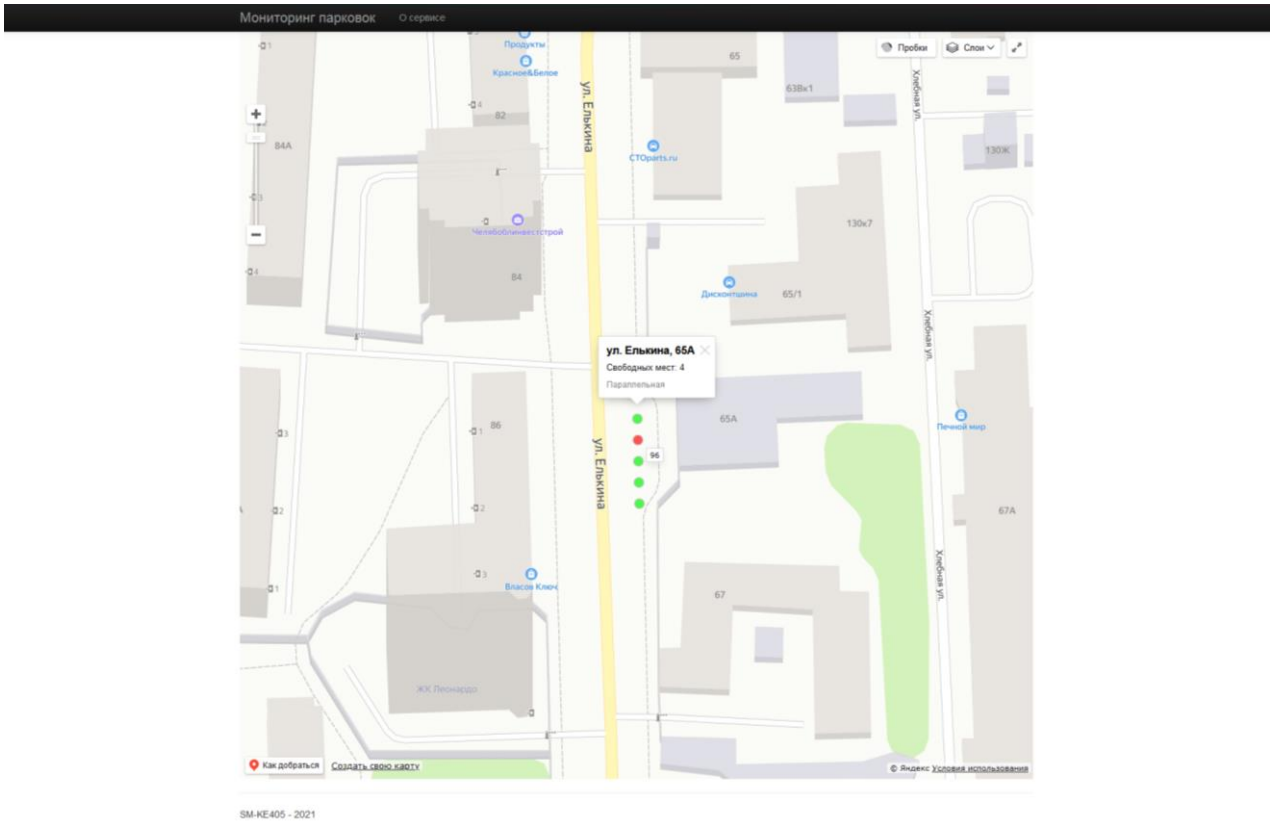


Рисунок 19 – Отображение на карте занятости парковочных мест после имитации парковки автомобиля

При вводе некорректной информации в качестве параметра сервер успешно обрабатывает возникающие исключения и возвращает сообщения об ошибке, не допуская выполнения транзакции с ошибочными данными.

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа освещает все вопросы, необходимые для решения задач проектирования, реализации и тестирования сервиса мониторинга занятости парковочных мест.

На основе необходимых требований создана схема базы данных, реализован сервер и клиентское приложение, проведено тестирование полученного клиент-серверного приложения с предполагаемым устройством сбора информации.

В ходе анализа предметной области выяснилось, что существующие на текущий момент времени аналоги имеют как достоинства, так и недостатки. Собственная система представляет собой новую версию идей программных решений, использованных в аналогах - доработанную и оптимизированную. Помимо собственно мониторинга занятости парковочных мест реализована возможность использования встроенного сервиса отображения дорожной ситуации Яндекс.Пробки и изменение типа карты на «Гибрид» (совмещение схематической карты и спутниковых снимков).

Для проведения тестирования разработанная система была размещена на хостинге. Для приложения был выделен веб-адрес. На стадии тестирования была проведена проверка клиентской части сервиса, сервера и базы данных, а также взаимодействия этих элементов. Процедура тестирования выявила корректную работу всех элементов и работоспособность системы в целом.

При условии обеспечения функционала аппаратной части системы (конечные устройства сбора информации) полученный вариант клиент-серверного приложения для мониторинга занятости парковочных мест, являясь полноценной оптимизированной и дополненной системой, готов для прикладного использования и решения практических задач мониторинга занятости парковочных мест.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Экономика и жизнь. Автомобилизация населения России по сравнению с поздним СССР выросла в пять раз [Электронный ресурс] // Экономика и Жизнь - официальный сайт издания. 2021. URL: https://www.eg-online.ru/news/431843/ (дата обращения: 04.06.2021).
2. SPOTParking | Главная [Электронный ресурс] // SPOTParking: [сайт]. [2020]. URL: https://checkthespot.ru/ (дата обращения: 19.04.2020).
3. Челябинцы начали массово пользоваться парковками с искусственным интеллектом [Электронный ресурс] // Интерсвязь - интернет-провайдер в Челябинске: [сайт]. [2020]. URL: https://www.is74.ru/oldsite/news/all/?ID=102680 (дата обращения: 19.04.2021).
4. Парковки - Яндекс.Карты. Справка [Электронный ресурс] // Яндекс.Справка: [сайт]. [2021]. URL: https://yandex.ru/support/maps/concept/park.html (дата обращения: 19.04.2021).
5. Пономарёв, А. Придуман сервис мониторинга свободной парковки [Электронный ресурс] // Популярная механика. 2020. URL: https://www.popmech.ru/vehicles/news-570384-priduman-servis-monitoringa-svobodnoy-parkovki/ (дата обращения: 19.04.2021).
6. Урал-Пресс. Компания «Интерсвязь» расширяет виртуальный сервис «Умные парковки» [Электронный ресурс] // Новости Челябинска и Челябинской области. 2020. URL: https://uralpress.ru/news/obshchestvo/kompaniya-intersvyaz-rasshiryaet-virtualnyy-servis-umnye-parkovki (дата обращения: 19.04.2021).
7. Смольников, Л. Сервис «Умные парковки» научился охранять автомобили [Электронный ресурс] // Южноуральская панорама. Новости Челябинска и Челябинской области. События, происшествия. 2019. URL: https://up74.ru/articles/news/111854/ (дата обращения: 19.04.2021).
8. Most Used Open-Source Web Frameworks: Laravel vs ASP.NET [Электронный

ресурс] // Best Agile Software Development Company | Bacancy Technology: [сайт]. [2021]. URL: https://www.bacancytechnology.com/blog/laravel-vs-asp-net (дата обращения: 04.06.2021).

9. Pros and Cons of Laravel - Nimap Infotech [Электронный ресурс] // Mobile App Development Company | IT Outsourcing Company: [сайт]. [2019]. URL: https:// nimapinfotech.com/blog/pros-and-cons-of-laravel/ (дата обращения: 04.06.2021).
10. Сухов, К.К. Node.js. Путеводитель по технологии: учебник / К.К. Сухов. - Москва: ДМК Пресс, 2015. - 416 с.
11. Review of Node.JS: Benefits and Shortcomings [Электронный ресурс] // NCube: [сайт]. [2020]. URL: https://ncube.com/blog/review-of-node-js-pros-and-cons (дата обращения: 04.06.2021).
12. Обзор ASP.NET | Microsoft Docs [Электронный ресурс] // Microsoft - официальная страница: [сайт]. [2019]. URL: https://docs.microsoft.com/ru-ru/ aspnet/overview (дата обращения: 04.06.2021).
13. Антонов, И.В. Выбор подхода к разработке веб-приложений бакалаврами направления подготовки "Информационные системы и технологии" / И.В. Антонов, Ю.В. Бруттан // Вестник Псковского государственного университета. - 2015. - Вып. 2. - С. 107-113.
14. ASP.NET Core Pros and Cons [Электронный ресурс] // UKAD: [сайт]. [2019]. URL: https://www.ukad-group.com/blog/aspnet-core-8-pros-and-3-cons/ (дата обращения: 04.06.2021).
15. Advantages and Disadvantages of.NET Framework [Электронный ресурс] // NCube: [сайт]. [2020]. URL: https://ncube.com/blog/pros-and-cons-of-net-framework (дата обращения: 04.06.2021).
16. Microsoft SQL Server Pros and Cons [Электронный ресурс] // LearnSQL.com: [сайт]. [2019]. URL: https://learnsql.com/blog/microsoft-sql-server-pros-and-cons/ (дата обращения: 04.06.2021).

17. Understanding the Pros and Cons of Using Microsoft SQL Server [Электронный ресурс] // Rothmobot: [сайт]. [2018]. URL: https://www.rothmobot.com/the-advantages-and-disadvantages-of-microsoft-sql-server/ (дата обращения: 04.06.2021).
18. What is PostgreSQL? Introduction, Advantages & Disadvantages [Электронный ресурс] URL: https://www.guru99.com/introduction-postgresql.html (дата обращения: 04.06.2021).
19. Pros and Cons of using PostgreSQL for Application Development [Электронный ресурс] // Aalpha: [сайт]. [2019]. URL: https://www.aalpha.net/blog/pros-and-cons-of-using-postgresql-for-application-development/ (дата обращения: 04.06.2021).
20. SQL Server vs. Oracle Database: What You Need to Know [Электронный ресурс] // SqlBot: [сайт]. [2019]. URL: https://www.sqlbot.co/blog/sql-server-vs-oracle (дата обращения: 04.06.2021).
21. Oracle Database Advantages, Disadvantages and Features [Guide 2021] [Электронный ресурс] // NineHertz: [сайт]. [2019]. URL: https:// theninehertz.com/blog/advantages-of-using-oracle-database (дата обращения: 04.06.2021).
22. Подключение API и SDK [Электронный ресурс] // 2ГИС: [сайт]. URL: https:// dev.2gis.ru/order/ (дата обращения: 04.06.2021).
23. API&SDK карт и справочника 2ГИС [Электронный ресурс] // 2ГИС: [сайт]. URL: https://dev.2gis.ru/ (дата обращения: 04.06.2021).
24. Overview | Maps JavaScript API [Электронный ресурс] // Google Developers: [сайт]. [2021]. URL: https://developers.google.com/maps/documentation/javascript/overview?hl=ru (дата обращения: 04.06.2021).
25. Pricing & Plans | Google Maps Platform [Электронный ресурс] // Google Cloud: [сайт]. URL: https://cloud.google.com/maps-platform/pricing/?hl=ru (дата обращения: 04.06.2021).

26. JavaScript API Яндекс.Карт [Электронный ресурс] // Технологии Яндекса: [сайт]. URL: https://yandex.ru/dev/maps/jsapi/?from=mapsapi (дата обращения: 04.06.2021).
27. О справочнике. Статьи [Электронный ресурс] // Технологии Яндекса: [сайт]. URL: https://yandex.ru/dev/maps/jsapi/doc/2.1/ref/concepts/About.html (дата обращения: 04.06.2021).
28. SmarterASP.net - Unlimited ASP.NET Web Hosting [Электронный ресурс] // SmarterASP.NET: [сайт]. URL: https://www.smarterasp.net/ (дата обращения: 04.06.2021).

ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД КЛИЕНТСКОЙ ЧАСТИ

Листинг А.1 - Файл Index.cshtml

```
@{
    ViewBag.Title = "Карта";
}
<input type="hidden" value="@ViewBag.Lots" name="JsonLotsField" />
<div class="container">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <script src="https://api-
maps.yandex.ru/2.1/?lang=ru_RU&amp;apikey=177680a4-27e0-412f-ab70-e501b1e9237c"
type="text/javascript"></script>
        <style>
            body, html {
                padding: 0;
                margin: 0;
                width: 100%;
                height: 100%;
            }

            #map {
                padding-top: 40px;
                max-width: 1170px;
                max-height: 1170px;
                width: 100vw;
                height: 85vh;
            }
        </style>
    </head>
    <body>
        <div align="center">
            <div id="map"></div>
        </div>
    </body>
        @section scripts {
            <script type="text/javascript">
                var mainMap;
                var placesCollection;
                var placesJson;
                var jsonLotsObj =
JSON.parse(document.getElementsByName("JsonLotsField")[0].value);
                var freeCounter = 0;

                ymaps.ready(init);
               // функция определения типа парковки
                function getTypeName(parkingType) {
                    switch (parkingType) {
                    case 1:
                        return "Параллельная";
                    case 2:
                        return "Под углом";
```

```
                    case 3:
                        return "Перпендикулярная";
                    default:
                        return "";
                    }
                }
            // функция отрисовки меток парковочных пространств на карте
                function placeParkingMarks(lotsJson) {
                    for (var i = 0; i < lotsJson.length; i++) {
                        var pExample = lotsJson[i];
                        var placemark = new
ymaps.Placemark([pExample.parking_latitude, pExample.parking_longitude],
                            {
                                balloonContentHeader: `${pExample.parking_name}`,
                                balloonContentFooter:
getTypeName(pExample.parking_type),
                                hintContent: `${pExample.parking_id}`
                            },
                            {
                                iconLayout: 'default#image',
                                iconImageHref: '/img/parking.png',
                                iconImageSize: [30, 42],
                                iconImageOffset: [-5, -38]
                            });
                        placemark.events.add('click',
                            function(e) {
                                mainMap.geoObjects.remove(placesCollection);
                                var pmCoords =
e.get('target').geometry._coordinates;
                                var pmId =
e.get('target').properties._data.hintContent;
                                $.ajax({
                                    url: '/Home/GetPlaces?id=' + pmId,
                                    dataType: 'json',
                                    async: false,
                                    type: "GET",
                                    success: function (data) {
                                        placesJson = data;
                                    },
                                    error: function (request, status, error) {
                                        alert(error + request);
                                    }
                                });
                                mainMap.setCenter([pmCoords[0], pmCoords[1]]);
                                setTimeout(function() { mainMap.setZoom(19); },
10);
                                drawParkingPlaces(placesJson);

e.get('target').properties._data.balloonContentBody = "Свободных мест: " +
freeCounter;
                                freeCounter = 0;
                            });
```

```
                        mainMap.geoObjects.add(placemark);
                    }
                }
            // функция определения цвета парковочного места
                function getPlaceColor(parkingPlace) {
                    if (parkingPlace.place_for_disabled === true &&
parkingPlace.place_is_free === true) {
                        freeCounter++;
                        return "#378BD7DC";
                    } else if (parkingPlace.place_is_free === true) {
                        freeCounter++;
                        return "#37F237DC";
                    } else {
                        return "#FB3939DC";
                    }
                }
            // функция отрисовки парковочных мест
                function drawParkingPlaces(placesJson) {
                    placesCollection = new ymaps.GeoObjectCollection;
                    var plExample = placesJson;
                    for (var j = 0; j < plExample.length; j++) {
                        var circle = new ymaps.Circle([
                                [plExample[j].place_latitude,
plExample[j].place_longitude],
                                1.25
                            ],
                            {
                                hintContent: `${plExample[j].place_id}`
                            },
                            {
                                fillColor: getPlaceColor(plExample[j]),
                                strokeColor: "#C3C3C3",
                                strokeOpacity: 1,
                                strokeWidth: 1
                            });
                        placesCollection.add(circle);
                    }
                    mainMap.geoObjects.add(placesCollection);
                }
            // функция инициализации карты
                function init() {
                    mainMap = new ymaps.Map('map',
                        {
                            center: [55.16, 61.40],
                            zoom: 12,
                            controls: ['zoomControl', 'fullscreenControl',
'trafficControl']
                        },
                        {
                            searchControlProvider: 'yandex#search'
                        });
```

```
                    var mapcontrols = new
ymaps.control.TypeSelector(['yandex#map', 'yandex#hybrid']);
                    mapcontrols.options.set({panoramasItemMode: 'off'});
                    mainMap.controls.add(mapcontrols);
                    placeParkingMarks(jsonLotsObj);
                    mainMap.events.add('boundschange',
                        function(e) {
                            if (e.get('newZoom') < 18) {
                                mainMap.geoObjects.remove(placesCollection);
                                mainMap.balloon.close();
                            }
                        });
                }
            </script>
        }
</div>
```

Листинг А.2 - Файл About.cshtml

```
@{
    ViewBag.Title = "О сервисе";
}
<hgroup>
    <h2>@ViewBag.Title</h2>
</hgroup>
<div class="row-fluid">
    <div class="span12">
        <p>Сервис позволяет получать информацию о занятости парковочных мест. Для
отображения текущего состояния необходимо нажать на маркер, после чего выведется
схема выбранной парковки. Сервис работает в тестовом режиме.</p>
    </div>
</div>
```

Листинг А.3 - Файл _Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Мониторинг занятости парковок</title>
    <link href="~/img/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="navbar-inner">
            <div class="container">
                <button type="button" class="btn btn-navbar" data-
toggle="collapse" data-target=".nav-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
```

```
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Мониторинг парковок", "Index", "Home", null, new
{ @class = "brand" })
                <div class="nav-collapse collapse">
                    <ul class="nav">
                        <li>@Html.ActionLink("О сервисе", "About", "Home")</li>
                    </ul>
                </div>
            </div>
        </div>
    </div>


    <div class="container">
        @RenderBody()
        <hr />
        <footer>
            <p>SM-KE405 - @DateTime.Now.Year</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

ПРИЛОЖЕНИЕ Б ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ

Листинг Б.1 - Файл HomeController.cs

```
using System.Linq;
using System.Web.Mvc;
using Newtonsoft.Json;

namespace parking_monitor.Controllers
{
    public class HomeController : Controller
    {
        ParkingContext db = new ParkingContext();

        public ActionResult Index()
        {
            var query = db.ParkingLots.Select(x => new
            {
                x.parking_id,
                x.parking_latitude,
                x.parking_longitude,
                x.parking_name,
                x.parking_type
            });
            ViewBag.Lots = JsonConvert.SerializeObject(query);
            return View();
        }

        [HttpGet]
        public ActionResult GetPlaces(int id)
        {
            var query = db.ParkingPlaces.Select(x => new
            {
                x.place_id,
                x.place_is_free,
                x.place_for_disabled,
                x.place_latitude,
                x.place_longitude,
                x.park_id
            }).Where(x => x.park_id == id);
            return Json(query, JsonRequestBehavior.AllowGet);
        }

        public ActionResult About()
        {
            return View();
        }
    }
}
```

Листинг Б.2 - Файл DeviceController.cs

```
using System;
using System.Linq;
using System.Web.Mvc;
using Newtonsoft.Json;
using parking_monitor.Models;


namespace parking_monitor.Controllers
{
    public class DeviceController : Controller
    {
        ParkingContext db = new ParkingContext();

        public string Index(int id)
        {
            var query = db.ParkingLots.Select(x => new
            {
                x.parking_id,
                x.parking_name,
                x.parking_type,
                x.parking_latitude,
                x.parking_longitude,
                x.ParkingPlaces
            }).Where(x => x.parking_id == id);
            if (JsonConvert.SerializeObject(query).Length > 2)
            {
                return JsonConvert.SerializeObject(query);
            }
            return "Parking lot with id = " + id + " doesn't exist in database.";
        }

        public string UpdateLot(string lotJson)
        {
            using (var transaction = db.Database.BeginTransaction())
            {
                try
                {
                    var placesUpdateCounter = 0;
                    var lotObject =
JsonConvert.DeserializeObject<ParkingLotMin>(lotJson);
                    var placesObject = lotObject.ParkingPlaces;
                    if (placesObject.Count > 0)
                    {
                        var parkingPlaces = db.ParkingLots.SingleOrDefault(x =>
x.parking_id == lotObject.parking_id).ParkingPlaces;
                        if (parkingPlaces != null)
                            foreach (var place in parkingPlaces)
                            {
                                if (placesObject.SingleOrDefault(o => o.place_id
== place.place_id) != null &&
```

```
                                    place.place_is_free != placesObject
                                        .SingleOrDefault(o => o.place_id ==
place.place_id).place_is_free)
                                {
                                    place.place_is_free =
placesObject.SingleOrDefault(o => o.place_id == place.place_id).place_is_free;
                                    placesUpdateCounter++;
                                }
                            }
                        if (placesUpdateCounter > 0)
                        {
                            db.SaveChanges();
                            transaction.Commit();
                            return "Parking lot with id = " + lotObject.parking_id
+ " state updated. Number of updated objects: " + placesUpdateCounter;
                        }
                        else
                        {
                            throw new Exception("No spots have been updated, but
collection is not empty. Check device settings.");
                        }

                    }
                    else
                    {
                        throw new Exception("Empty spot collection");
                    }
                }
                catch (Exception e)
                {
                    transaction.Rollback();
                    return "Transaction rollback. Error: " + e.Message;
                }
            }
        }
    }
}
```

Листинг Б.3 - Файл ParkingModel.Context.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
//    Этот код был создан из шаблона.
//
//    Изменения, вносимые в этот файл вручную, могут привести к непредвиденной
работе приложения.
//    Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном
создании кода.
// </auto-generated>
//------------------------------------------------------------------------------

namespace parking_monitor
{
```

```
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;

    public partial class ParkingContext : DbContext
    {
        public ParkingContext()
            : base("name=ParkingContext")
        {
        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        public DbSet<ParkingLot> ParkingLots { get; set; }
        public DbSet<ParkingPlace> ParkingPlaces { get; set; }
    }
}
```

Листинг Б.4 - Файл ParkingLot.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
//    Этот код был создан из шаблона.
//
//    Изменения, вносимые в этот файл вручную, могут привести к непредвиденной
работе приложения.
//    Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном
создании кода.
// </auto-generated>
//------------------------------------------------------------------------------

namespace parking_monitor
{
    using System;
    using System.Collections.Generic;

    public partial class ParkingLot
    {
        public ParkingLot()
        {
            this.ParkingPlaces = new HashSet<ParkingPlace>();
        }
        public int parking_id { get; set; }
        public double parking_latitude { get; set; }
        public double parking_longitude { get; set; }
        public string parking_name { get; set; }
        public Nullable<byte> parking_type { get; set; }
        public virtual ICollection<ParkingPlace> ParkingPlaces { get; set; }
    }
}
```

Листинг Б.5 - Файл ParkingPlace.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
//    Этот код был создан из шаблона.
//
//    Изменения, вносимые в этот файл вручную, могут привести к непредвиденной
работе приложения.
//    Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном
создании кода.
// </auto-generated>
//------------------------------------------------------------------------------

using Newtonsoft.Json;

namespace parking_monitor
{
    using System;
    using System.Collections.Generic;

    public partial class ParkingPlace
    {
        public int place_id { get; set; }
        public double place_latitude { get; set; }
        public double place_longitude { get; set; }
        public bool place_is_free { get; set; }
        public bool place_for_disabled { get; set; }
        public int park_id { get; set; }

        [JsonIgnore]
        public virtual ParkingLot ParkingLot { get; set; }
    }
}
```

Листинг Б.6 - Файл ParkingLotMin.cs

```
using System.Collections.Generic;

namespace parking_monitor.Models
{
    public class ParkingLotMin
    {
        public ParkingLotMin()
        {
            this.ParkingPlaces = new HashSet<ParkingPlaceMin>();
        }

        public int parking_id { get; set; }

        public virtual ICollection<ParkingPlaceMin> ParkingPlaces { get; set; }

    }
}
```

Листинг Б.7 - Файл ParkingPlaceMin.cs

```
using Newtonsoft.Json;

namespace parking_monitor.Models
{
    public class ParkingPlaceMin
    {
        public int place_id { get; set; }
        public bool place_is_free { get; set; }
        public int park_id { get; set; }

        [JsonIgnore]
        public virtual ParkingLotMin ParkingLot { get; set; }
    }
}
```

Листинг Б.8 - Файл Web.config

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=301879
  -->
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection,
EntityFramework, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" requirePermission="false" />
  </configSections>
  <connectionStrings>
    <add name="ParkingDB" connectionString="Data
Source=SQL5092.site4now.net;Initial Catalog=db_a748b4_pm;User
Id=db_a748b4_pm_admin;Password=********;" providerName="System.Data.SqlClient" />
    <add name="ParkingContext"
connectionString="metadata=res://*/ModelFromDB.csdl|res://*/ModelFromDB.ssdl|res:/
/*/ModelFromDB.msl;provider=System.Data.SqlClient;provider connection
string=&amp;quot;data source=SQL5092.site4now.net;initial catalog=db_a748b4_pm;user
id=db_a748b4_pm_admin;password=********;MultipleActiveResultSets=True;App=EntityFr
amework&amp;quot;" providerName="System.Data.EntityClient" />
  </connectionStrings>
  <appSettings />
  <!--
    Описание изменений web.config см. по адресу
http://go.microsoft.com/fwlink/?LinkId=235367.
    Следующие атрибуты можно установить с помощью тега <httpRuntime>.
      <system.Web>
        <httpRuntime targetFramework="4.5.2" />
      </system.Web>
  -->
```

```
  <system.web>
    <compilation debug="true" targetFramework="4.7.2" />
    <httpRuntime targetFramework="4.5" maxUrlLength="10999"
maxQueryStringLength="2097151" />
    <globalization culture="en-US" uiCulture="en-US" />
  <httpHandlers>
      <add verb="*" path="routes.axd"
type="AttributeRouting.Web.Logging.LogRoutesHandler, AttributeRouting.Web" />
    </httpHandlers></system.web>
  <system.webServer>
    <security>
      <requestFiltering>
        <requestLimits maxUrl="500" maxQueryString="2097151" />
      </requestFiltering>
    </security>
    <validation validateIntegratedModeConfiguration="false" />
    <handlers>
      <remove name="ExtensionlessUrlHandler-ISAPI-4.0_32bit" />
      <remove name="ExtensionlessUrlHandler-ISAPI-4.0_64bit" />
      <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
      <add name="ExtensionlessUrlHandler-ISAPI-4.0_32bit" path="*."
verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS" modules="IsapiModule"
scriptProcessor="%windir%\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll"
preCondition="classicMode,runtimeVersionv4.0,bitness32" responseBufferLimit="0" />
      <add name="ExtensionlessUrlHandler-ISAPI-4.0_64bit" path="*."
verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS" modules="IsapiModule"
scriptProcessor="%windir%\Microsoft.NET\Framework64\v4.0.30319\aspnet_isapi.dll"
preCondition="classicMode,runtimeVersionv4.0,bitness64" responseBufferLimit="0" />
      <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*."
verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS"
type="System.Web.Handlers.TransferRequestHandler"
preCondition="integratedMode,runtimeVersionv4.0" />
      <remove name="OPTIONSVerbHandler" />
      <remove name="TRACEVerbHandler" />
    <add name="AttributeRouting" path="routes.axd" verb="*"
type="AttributeRouting.Web.Logging.LogRoutesHandler, AttributeRouting.Web"
/></handlers>
  </system.webServer>
  <entityFramework>
    <providers>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer"
/>
    </providers>
  </entityFramework>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-13.0.0.0" newVersion="13.0.0.0" />
      </dependentAssembly>
```

```
      <dependentAssembly>
        <assemblyIdentity name="System.Runtime.CompilerServices.Unsafe"
publicKeyToken="b03f5f7f11d50a3a" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-5.0.0.0" newVersion="5.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-5.0.0.0" newVersion="5.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.WebPages"
publicKeyToken="31bf3856ad364e35" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```