Министерство образования Республики Беларусь

Учреждения образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра автоматизированные системы обработки информации

Расчетно-пояснительная записка к курсовой работе
по курсу «Проектирование автоматизированных систем»
на тему:

# «МОДУЛЬ СТАТИСТИЧЕСКОГО УЧЕТА ЗАГРУЖЕННОСТИ АВТОСТОЯНОК»

Руководитель работы:
Доцент кафедры ИТАС
Ломако А. В.

Выполнил:
Студент группы 720601
Тихонович А. М.

Минск 2020

# РЕФЕРАТ

МОДУЛЬ СТАТИСТИЧЕСКОГО УЧЕТА ЗАГРУЖЕННОСТИ АВТОСТОЯНОК: курсовой проект / А.М. Тихонович. – Минск: БГУИР, 2020, – п.з. –53 с., чертежей (плакатов) – 1 л. формата А1.

Курсовой проект на тему «Модуль статистического учета загруженности автостоянок» разработан с целью повышения степени удовлетворенности клиентов за счет за накопленной информации о клиентском поведении, правильной настройке инструментов маркетинга, оптимизации работы сотрудников компании.

Пояснительная записка к курсовому проекту состоит из введения, 4 разделов, включающих литературный обзор по теме курсового проекта, сравнительный анализ существующих аналогов системы, описание процессов проектирования и программной реализации, а также технико-экономического обоснования разработки и внедрения системы, заключение, список использованных источников и приложение, содержащее листинг кода отдельных классов.

Для разработки автоматизированной информационной системы был выбран объектно-ориентированный язык программирования *ABAP 7.40*. Система представляет собой веб-приложение доступ, к которому, пользователи (работники автостоянок) смогут получать посредствам веб-браузера. Она предназначена для использования в локальной сети либо сети Интернет.

В результате работы над курсовым проектом была спроектирована и разработана система управления бронирования места на автостоянках, дополнение базы данных автостоянками, подготовлено руководство пользователя и выполнено экономическое обоснование разработки и внедрения системы.

Результаты, полученные в ходе курсового проектирования, могут использоваться участниками компаний массового обслуживания. Данная модель имеет все сведения о структуре такой системы, как компоненты системы взаимосвязаны друг с другом. Созданная модель является также хорошим примером создания *UML* диаграмм и может служить примером написания моделей других проектов. Автоматизация позволяет повысить производительность и качество системы, оптимизировать процессы управления, снизить затраты.

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет информационных технологий и управления

УТВЕРЖДАЮ
Заведующий кафедрой
_____

(подпись)
_____2020г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Тихонович Александре Максимовне*

1. Тема проекта Модуль статистического учета загруженности автостоянок.
2. Срок сдачи студентом законченного проекта *20 декабря 2020 г.*
3. Исходные данные к проекту, *web*-сайт предлагает введение операции бронирования, контролируемой системой бронирования. Система обеспечивает возможность забронировать место, отменить бронь. Также система предоставляет пользователю возможность изменять настройки автостоянки, а так же просматривать статистику ее загруженности. В системе фиксируются все необходимые данные о машинах и автостоянках. Среда разработки – *SAP Logon*, *BPwin*, *Enterprise Architect* .
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке):

*Введение*

1. *Общесистемная часть*
2. *Проектно-расчетная часть*
3. *Реализационная часть*

*Заключение*

*Список использованных источников*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Диаграмма вариантов использования (ПД, формат А1)*

6. Консультант по проекту (с обозначением разделов проекта) *А. В. Ломако*

7. Дата выдачи задания *10 сентября 2020 г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*раздел 1 к 15.10 – 15 %;*

*раздел 2 к 15.11 – 50 %;*

*раздел 3 к 15.12 – 15 %;*

*оформление пояснительной записки и графического материала к 20. 12 – 20 %*

*Защита курсового проекта с **20.12** по **22.12.2020**г.*

Руководитель *А. В. Ломако*

(подпись)

Задание принял к исполнению *А. М. Тихонович*

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

В настоящее время *Web* приложения пользуются особой популярностью, поэтому было решено создать приложение, одним из вариантов использования которого является упрощение управления автостоянками машин, добавления машин на автостоянку и наглядное представление машин на автостоянке.

Реализация модуля статистического учета загруженности автостоянок значительно облегчит работу работникам автостоянок и их менеджерам, а также ускорить работу их взаимодействия.

Курсовой проект выполнен с учётом требований к содержанию и оформлению курсового проекта, а также в соответствии с методическими указаниями.

Пояснительная записка является документацией к разработанному приложению и содержит следующую информацию:

В первом разделе происходит обзор состояния вопроса. Рассматриваются основные технологии, которые необходимо применить при написании приложения.

Во втором разделе дана постановка задачи.

В третьем разделе описаны этапы проектирования приложения. Приведены основные графические схемы и диаграммы.

В четвертом разделе описана реализация приложения, приведены наиболее интересные фрагменты кода с описанием их реализации.

В пятом разделе приведено функциональное тестирование приложения.

На сегодняшний день развитие интернет-технологий вносит новые преимущества в разработку таких систем. Современные средства построения интернет-приложений делают сравнимыми по быстроте и удобству с обычными настольными приложениями, являясь при этом доступными в любое время и в любом месте и будучи платформ независимыми. Такие преимущества склоняют заказчиков, а, следовательно, и разработчиков, к созданию именно приложений в веб-интерфейсом как альтернативы (а теперь все чаще – как замены) настольным решениям.

# 1 ОБЩЕСИСТЕМНАЯ ЧАСТЬ

## 1.1 Анализ предметной области

Курсовой проект по дисциплине «Проектирование автоматизированных систем» предусматривает проектирование и разработку систем обработки данных для заданных объектов управления, автоматизированных систем (АИС) разного назначения, программных модулей.

В данном курсовой работе рассматривается программный модуль статистического учета загруженности автостоянок.

Функционирование данного сервиса строится на предоставлении возможности быстрого и простого способа взаимодействия между работниками автостоянок и системой регистрации машин. Актуальность данного сервиса обусловлена тем, что работникам будет проще вводить все данные о машинах и автостоянках, так же работникам будет проще увидеть всю статистику по автостоянкам и их загруженность.

## 1.2 Аналоги

Существует немалое количество сервисов с подробной информацией о автостоянках. Во время рассмотрения аналогов учитывается в первую очередь функциональность сервиса – то, насколько широкий спектр возможностей он предоставляет. Так же учитывается удобство то, насколько этим сервисом удобно пользоваться.

### 1.2.1 Сервис *parkouka.by*

Является широкой сетью с информацией о автостоянках, их местонахождении, стоимости за час парковки и т.д. Предоставляет личный кабинет, правила пользования, контроль и обратную связь. Данный сервис позволяет найти автостоянку по карте и посмотреть насколько она нагружена. Главная страница сайта представлена на рисунке 1.1.

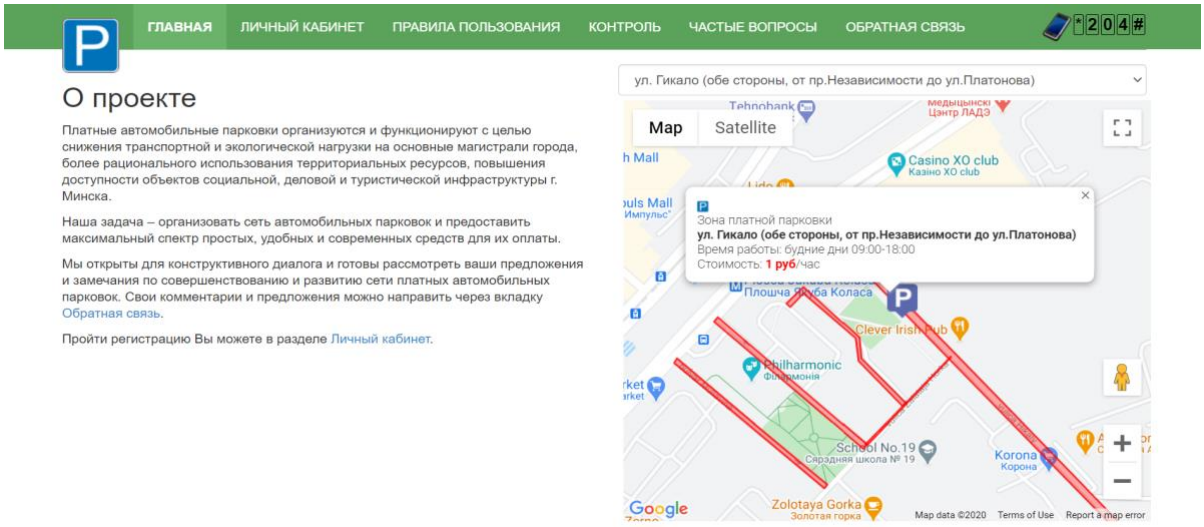



Рисунок 1.1 – Главная страница сайта *www.parkouka.by*

Дизайн сайта выглядит приятно.

### 1.2.2 Сервис *upgaip.by*

Также является сервисом с информацией о автостоянках. Не предоставляет возможность посмотреть из загруженность и забронировать место на автостоянке. Больше других возможностей о. Главная страница сайта представлена на рисунке 1.2.

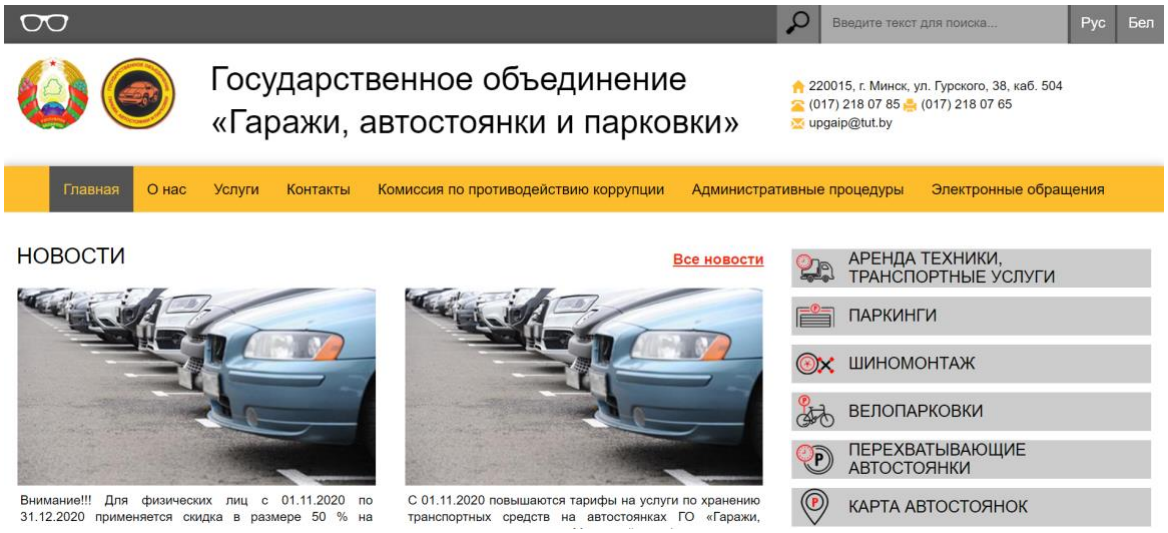



Рисунок 1.2 – Главная страница сайта *www.pgaip.by*

Дизайн сайта выполнен в нейтральных тонах, выглядит минималистично и приятно.

### 1.2.3 Сервис *autoban.by*

Является самым удачным на данный момент сервисом просмотра загруженности автостоянок в Беларуси. Главная страница сайта представлена на рисунке 1.3.

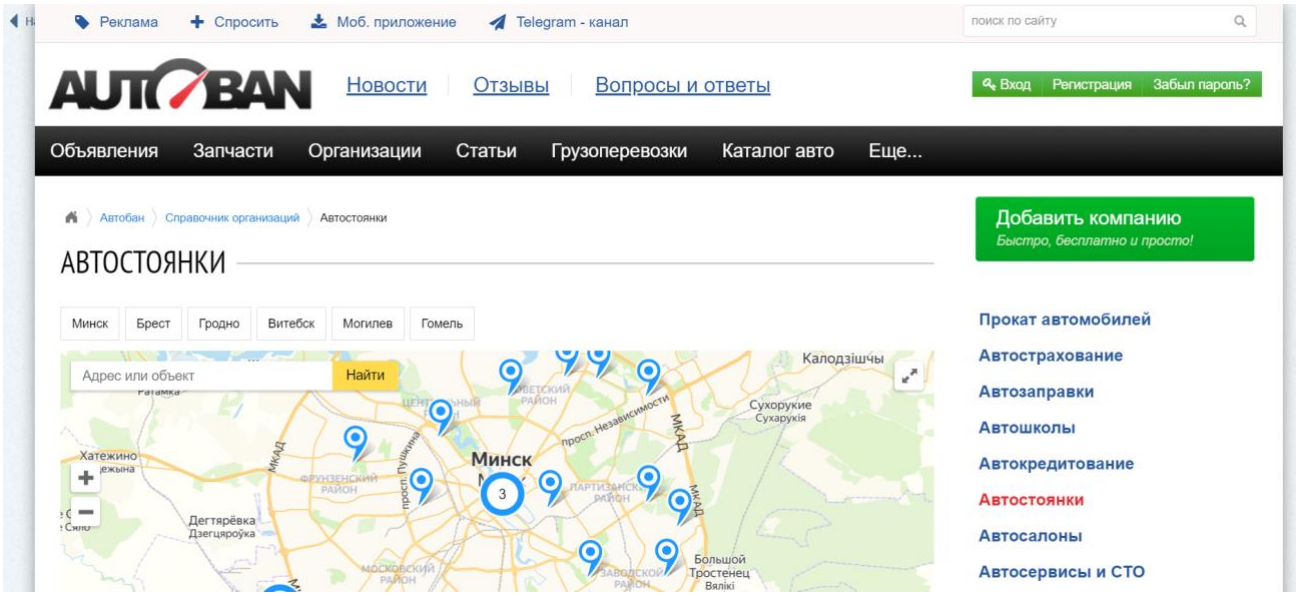



Рисунок 1.3 – Главная страница сайта *www.autoban.by*

Предоставляет всю информацию о автостоянках и все услуги связанные с автотранспортом, прокатом, мойкой и т.д. Позволят работать с автостоянками напрямую без посредников.

## 1.3 Постановка задачи

Перед разработчиком стоит задача спроектировать и разработать веб-сервис статистического учета загруженности автостоянок. Она включает в себя подробное изучение предметной области данного курсовой работы: сбор и группировка информации о данных автостоянок и машин. В результате должен получиться проект веб-сервис, который бы позволял хранить, обрабатывать,

автоматизировать и изменять информацию. Веб-сервис должен иметь удобный, лёгкий и доступный для восприятия пользовательский интерфейс. Должны быть продуманы специальные запросы по систематизации и обработке хранимой информации. В проекте должны быть изучены и хорошо продуманы вопросы защиты и обновления информации. Данный проект должен быть предназначен для круга работников автостоянок для облегчения регистрации машин.

В данном курсовом проекте проектируется БД, которую может использовать любой зарегистрированный пользователь (работник автостоянки). БД облегчает работу работникам, потому что можно свободно и легко найти информацию об интересующем автостоянке и машин на ней, и для этого затратить мало сил и времени.

– содержать необходимую информацию о автостоянках, машинах, арендах мест на автостоянках;

– обеспечивать возможность выполнять запрос, поиск, изменение и систематизацию данных;

– иметь удобный пользовательский интерфейс для работы с ней любого пользователя;

– иметь необходимые запросы и формы для обработки хранимой информации.

## 1.4 Анализ возможностей методологии и инструментальных средств проектирования

При разработке информационной системы был использован системный структурный подход. Модели предоставляются в виде иерархично упорядоченных диаграмм.

Для разработки *IDEF0, IDEF3* и *DFD* диаграмм можно использовать *AllFusion Process Modeler*. *AllFusion Process Modeler*, совмещает в одном инструменте средства моделирования функций (*IDEF0*), потоков данных (*DFD*) и потоков работ (*IDEF3*), координируя эти три основных аспекта бизнеса для соответствия потребностям аналитиков и системных аналитиков. *AllFusion Process Modeler*, позволяет повторно использовать ключевую информацию моделирования с точки зрения базовых аспектов, чтобы определить точки конфликтов и, в конечном счете, достичь их согласования.

*Adobe Architecture* представляет собой *CASE* средство проектирования и разработки информационных систем и программного обеспечения для управления предприятиями. Как и другие *CASE* средства его можно применять для анализа и моделирования бизнес процессов.

Принципиальное отличие *Adobe Architecture* от других средств заключается в объектно-ориентированном подходе. Графические модели, создаваемые с помощью этого средства, основаны на объектно-ориентированных принципах и языке *UML (Unified Modeling Language)*. Инструменты моделирования *Adobe Architecture* позволяют разработчикам создавать целостную архитектуру процессов предприятия, сохраняя все взаимосвязи и управляющие воздействия между различными уровнями иерархии.

*Microsoft Visio* – векторный графический редактор, редактор диаграмм и блок-схем для *Windows*. Аналогично с *Adobe Reader*, в стандартный набор программ *MS Office* входит только средство для просмотра и печати диаграмм *Microsoft Visio Viewer*. Первоначально *Visio* разрабатывался и выпускался компанией *Visio Corporation*. *Microsoft* приобрела компанию в 2000 году, тогда продукт назывался *Visio* 2000, выполнен ребрендинг, и продукт включен в состав *Microsoft Office*. Последняя версия продукта выпущена в 2016 году.

# 2 ПРОЕКТНО-РАСЧЕТНАЯ ЧАСТЬ

## 2.1 Функциональная методология *IDEF0*-диаграммы

Описание системы с помощью *IDEF0* называется функциональной моделью. Функциональная модель предназначена для описания существующих бизнес-процессов, в котором используются как естественный, так и графический языки. Для передачи информации о конкретной системе источником графического языка является сама методология *IDEF0*.

Методология *IDEF0* предписывает построение иерархической системы диаграмм - единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция - система разбивается на подсистемы, и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

Вход – материальные объекты (например, сырье) или информация, обрабатываемые в процессе выполнения работы для получения результата (Выхода).

Управления – правила выполнения работы (методы, стандарты и т.д.).

Механизмы – ресурсы для выполнения работы (персонал, станки, оборудование и т.д.).

Выход – результат выполнения работы (готовая продукция, результаты анализа информации и т.д.).

Блок диаграмма с соответствующими элементами приведена на рисунке 2.1.


Рисунок 2.1 – Блок диаграмма

## 2.2 Создание модели в стандарте *IDEF0*-диаграммы

Сначала приводится описание функционирования системы в целом в виде контекстной диаграммы. Контекстная диаграмма представлена на рисунке 2.2.

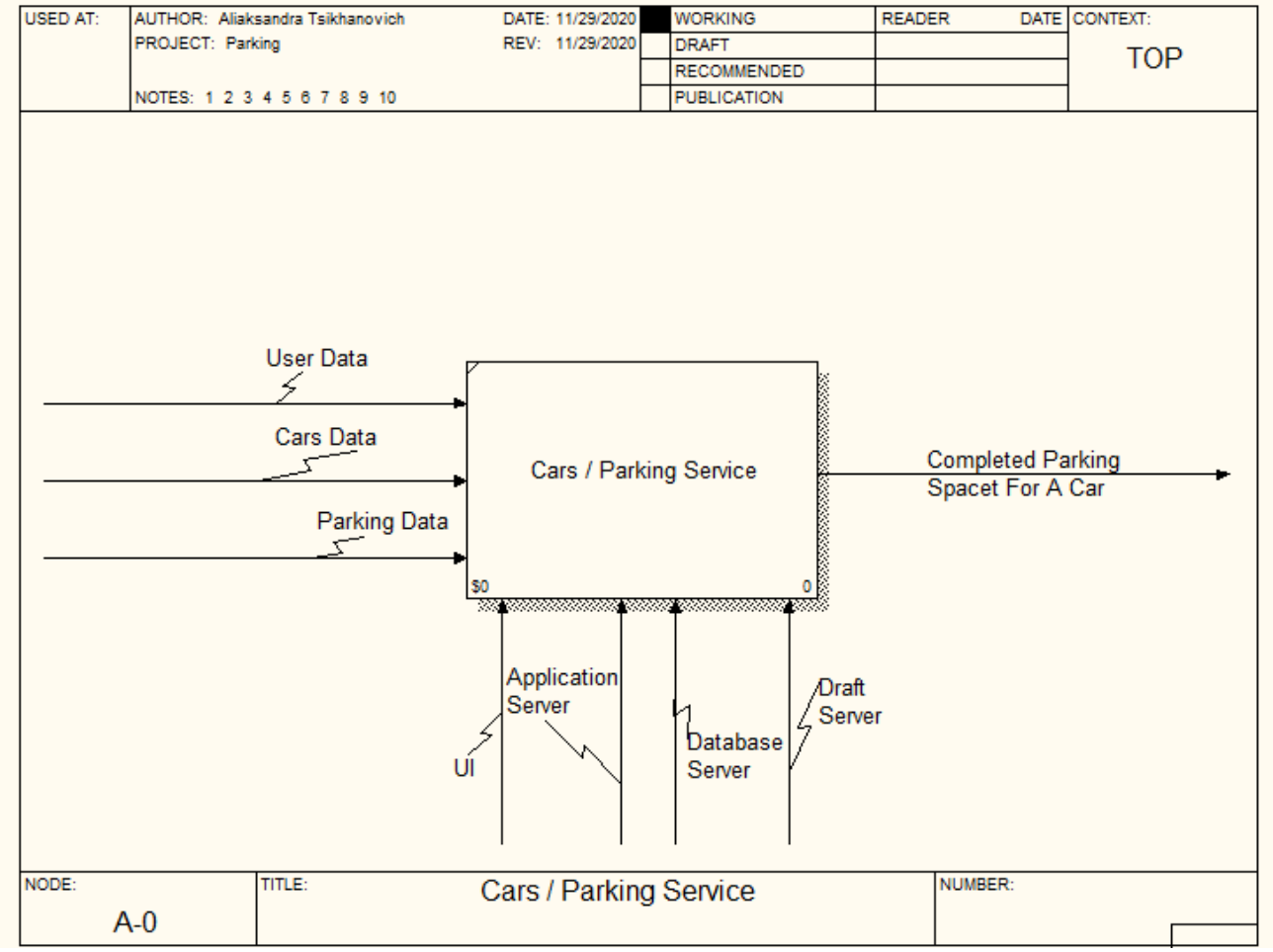



Рисунок 2.2 – Контекстная диаграмма

На вход программного модуля подаются сведения о пользователе, машине и автостоянок.

На выходе пользователь будет видеть подробную информацию о окончательной аренде парковочного места на автостоянке.

Механизмы, обеспечивающие работу данного программного модуля: интерфейс системы – передаёт программному модулю информацию о действиях пользователя; сервер приложения – обрабатывает запросы программного модуля;

сервер базы данных – обеспечивает программный модуль необходимой информацией.

На рисунке 2.3 представлена схема функционирования модуля.

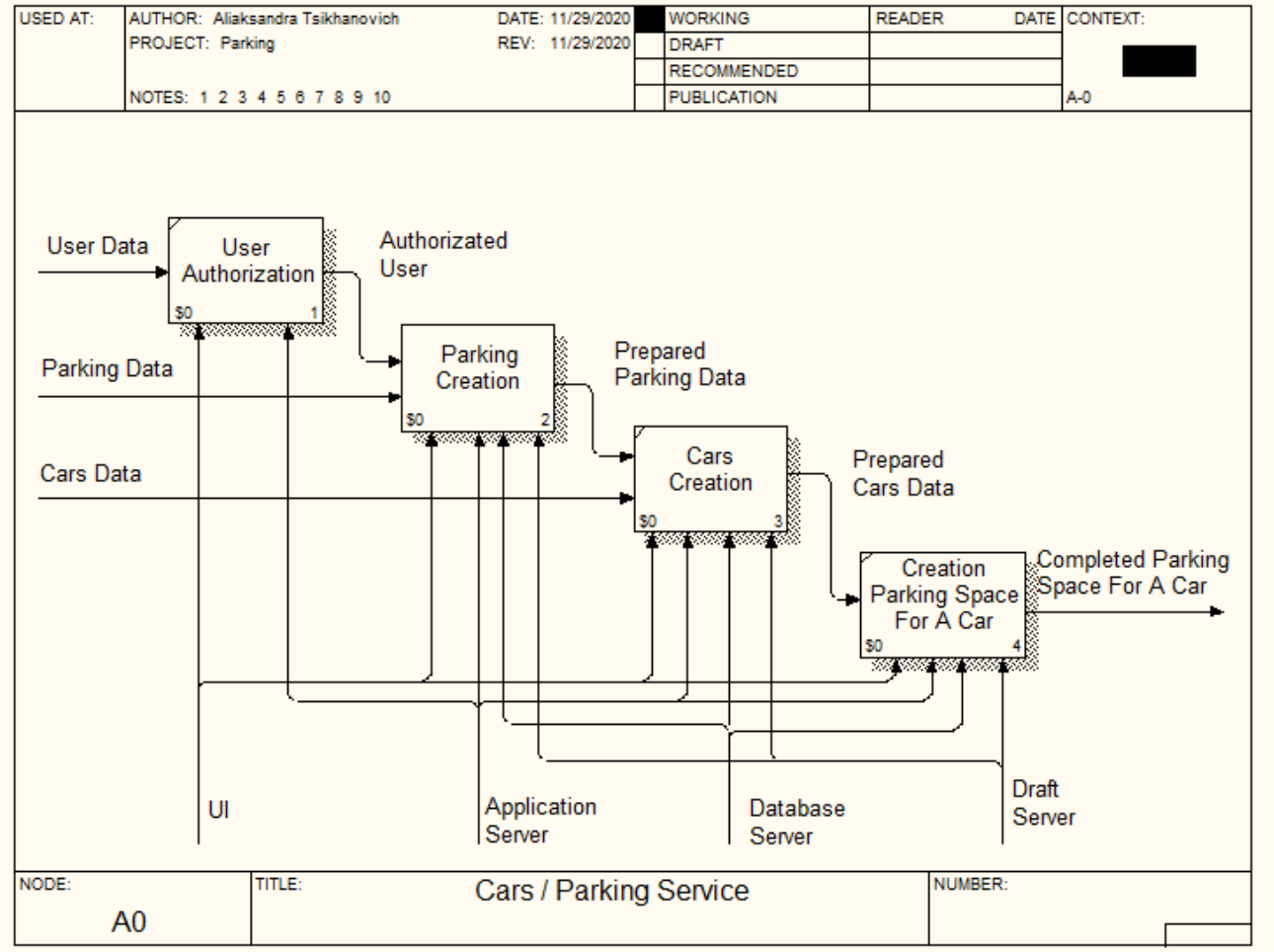



Рисунок 2.3 – Диаграмма декомпозиции

На диаграмме изображены действия, которые возможны в процессе работы с программным модулем.

Для получения прав на создание записи о машине или автостоянке пользователь должен авторизоваться в системе. В случае отсутствия аккаунта – зарегистрироваться. Для этого нужно ввести запрашиваемые данные и подтвердить регистрацию переходом по ссылке, которая будет выслана на указанную пользователем почту.

Далее пользователь видит список доступных автостоянок для машин. Пользователь создать запись о машине с нужными параметрами для его поиска.

Каждый функциональный блок можно рассмотреть в виде вложенной контекстной диаграммы. Блок разбивается на дочернюю декомпозицию и представляет более углубленную структуру компонента системы или отдельной её составляющей. Для примера приведена диаграмма декомпозиции для работы «Создание записи автостоянки», показанная на рисунке 2.4.

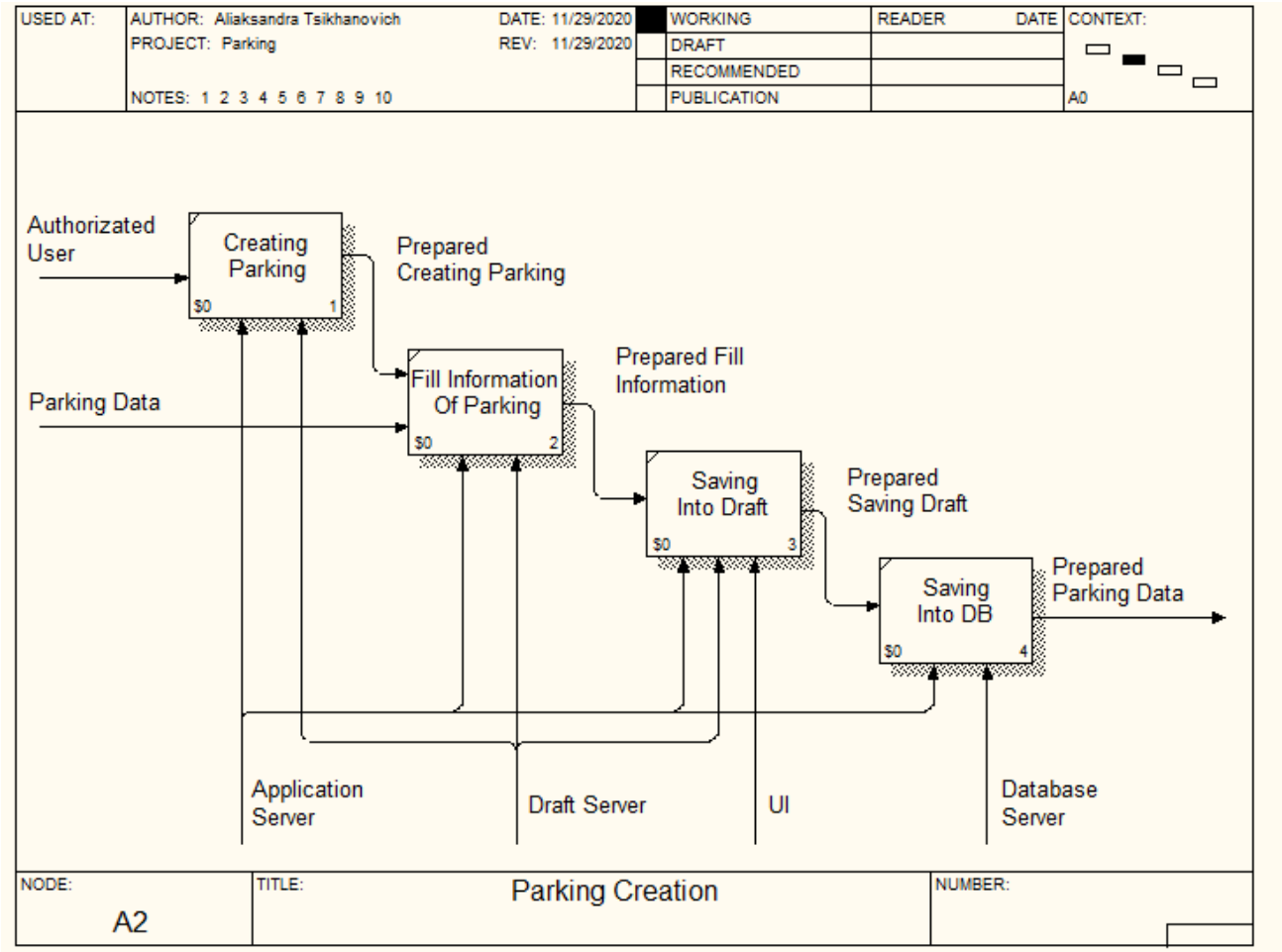



Рисунок 2.4 – Диаграмма декомпозиции работы «Создание записи автостоянки»

## 2.3 Методология *DFD*-диаграммы

Функциональная методология *DFD*. Диаграммы потоков, данных применяется главным образом для описания процессов документооборота и обработки информации. Диаграммы, построенные на основе методологии *DFD*, обычно используются как дополнение к *IDEF0*-диаграммам для построения моделей тех элементов объекта управления, основные функции которых связаны

с обработкой информации. На рисунке 2.5 представлена детализация блока «Создание записи аренды места для машины»

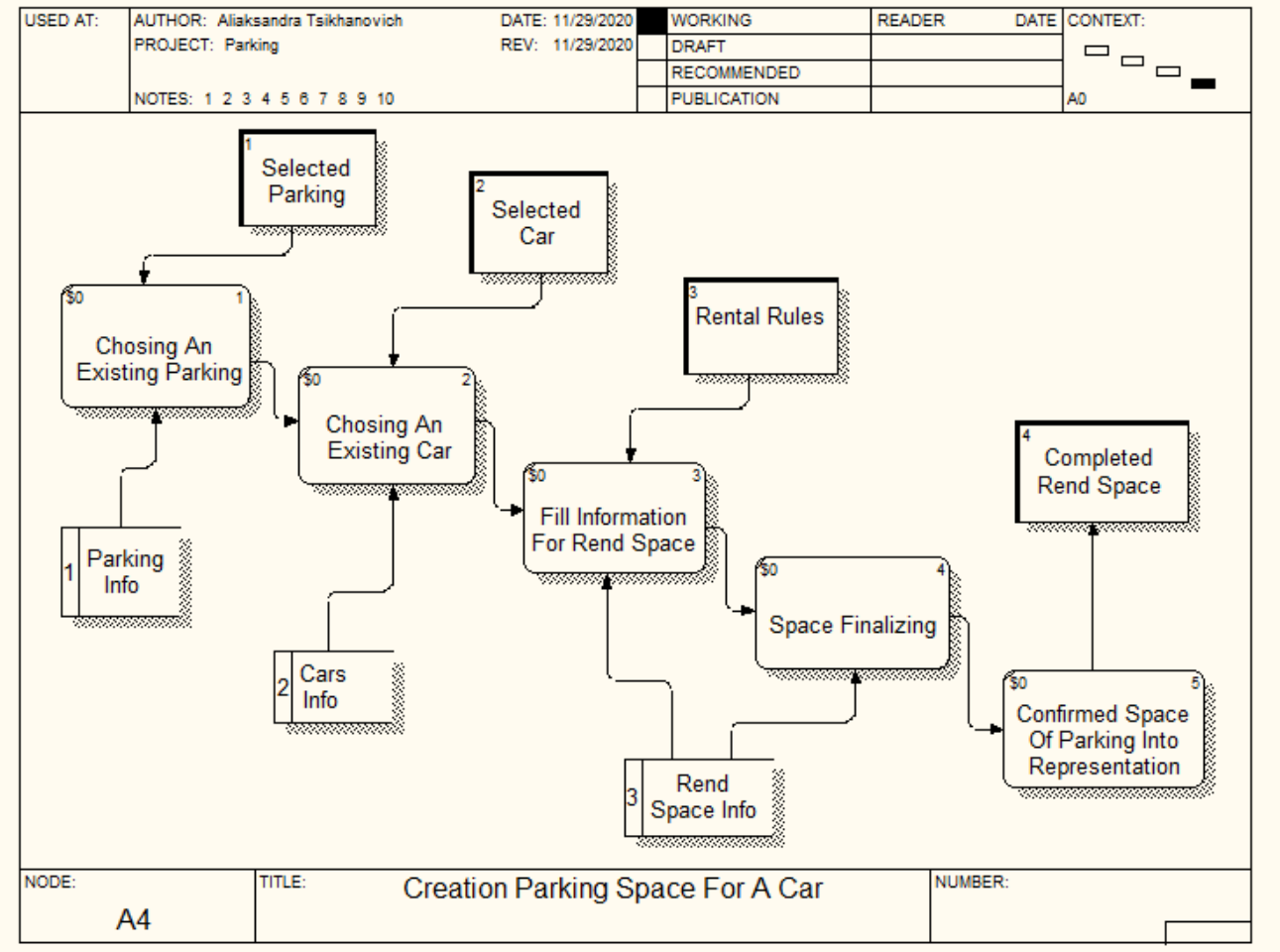



Рисунок 2.5 – Создание записи аренды места для машины

В качестве внешних источников данных используются данные: данные введенные работником и правила автостоянки. В качестве хранилища данных используются таблицы из базы данных, хранящие информацию о автостоянках, машинах и пользователях.

## 2.4 Методология *IDEF3*-диаграммы

Диаграммы *IDEF3*. Методология *IDEF3* применяется для описания взаимодействия процессов (работ), т.е. порядка их выполнения, а также

логических связей между ними. Диаграмма *IDEF3* построена для работы «Создание записи о машине», которая представлена на рисунке 2.6.

Первый этап – проверка введенных пользователем данных о документах владельца и информации о машине. Если проверка проходит успешно, то заполнение и проверка на подлинность информации. Если же такого пользователя не существует, или такой машины, то создание записи прерывается. Это отражено на перекрестке *J3*.

Перекрёсток *J2* означает, что выход на следующий этап верификации – создания записи о машине – возможен только в случае, если введенные пользователем данные пройдут проверку. В противном случае пользователь будет проинформирован о там, что именно не прошло верификацию. Также ему будет предложено исправить неверно введенные данные.

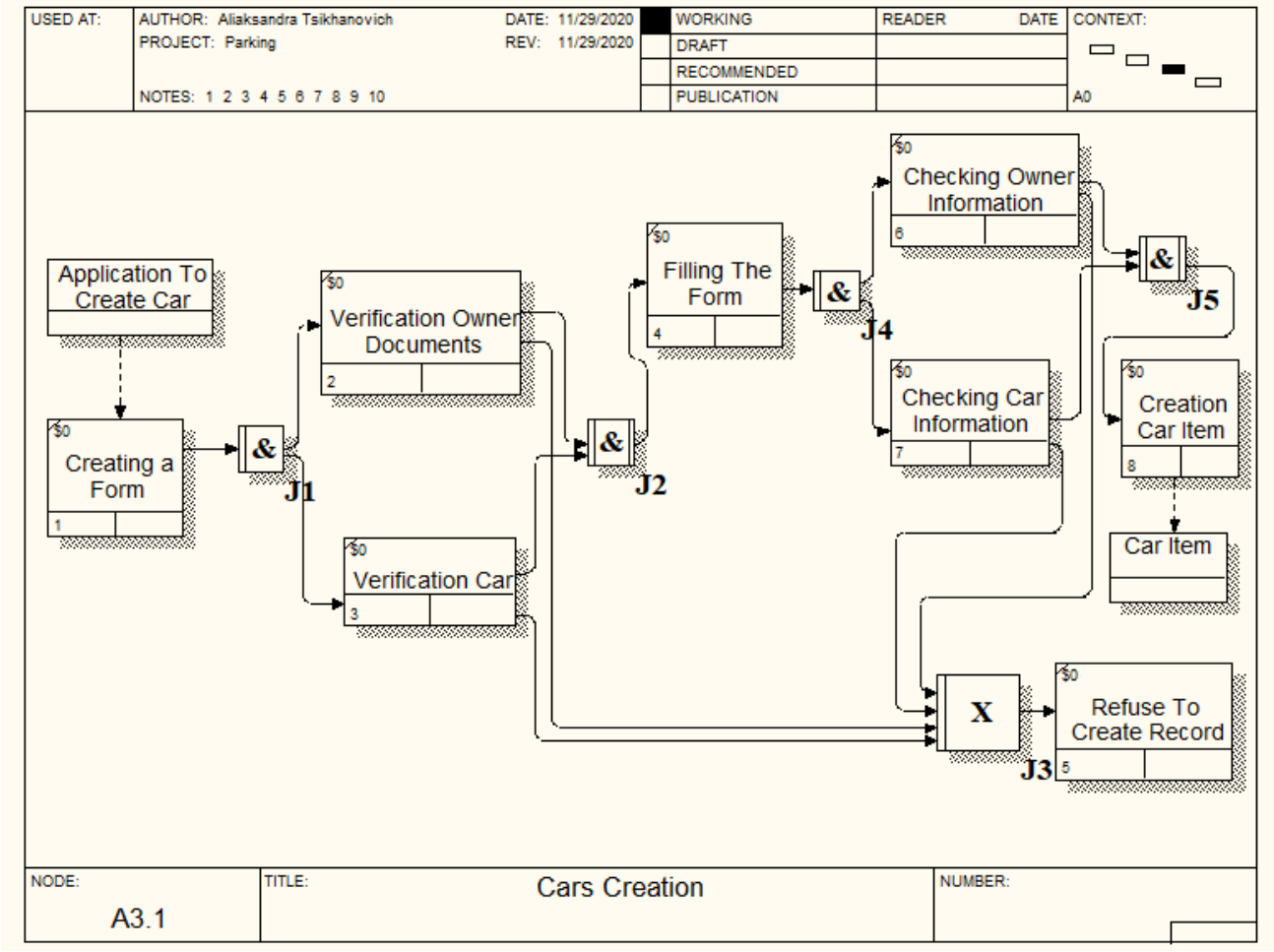



Рисунок 2.6 – *IDEF3* диаграмма «Создание записи о машине»

## 2.5 Диаграмма вариантов использования

Унифицированный язык моделирования (*UnifiedModelingLanguage, UML*) является графическим языком для визуализации, специфицирования, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению. С помощью *UML* можно разработать детальный план создаваемой системы, отображающий не только ее концептуальные элементы, такие как системные функции и бизнес-процессы, но и конкретные особенности реализации, в том числе классы, написанные на специальных языках программирования, схемы баз данных и программные компоненты многократного использования.

Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

На рисунке 2.7 представлена диаграмма вариантов использования «модуль статистического учета загруженности автостоянок».

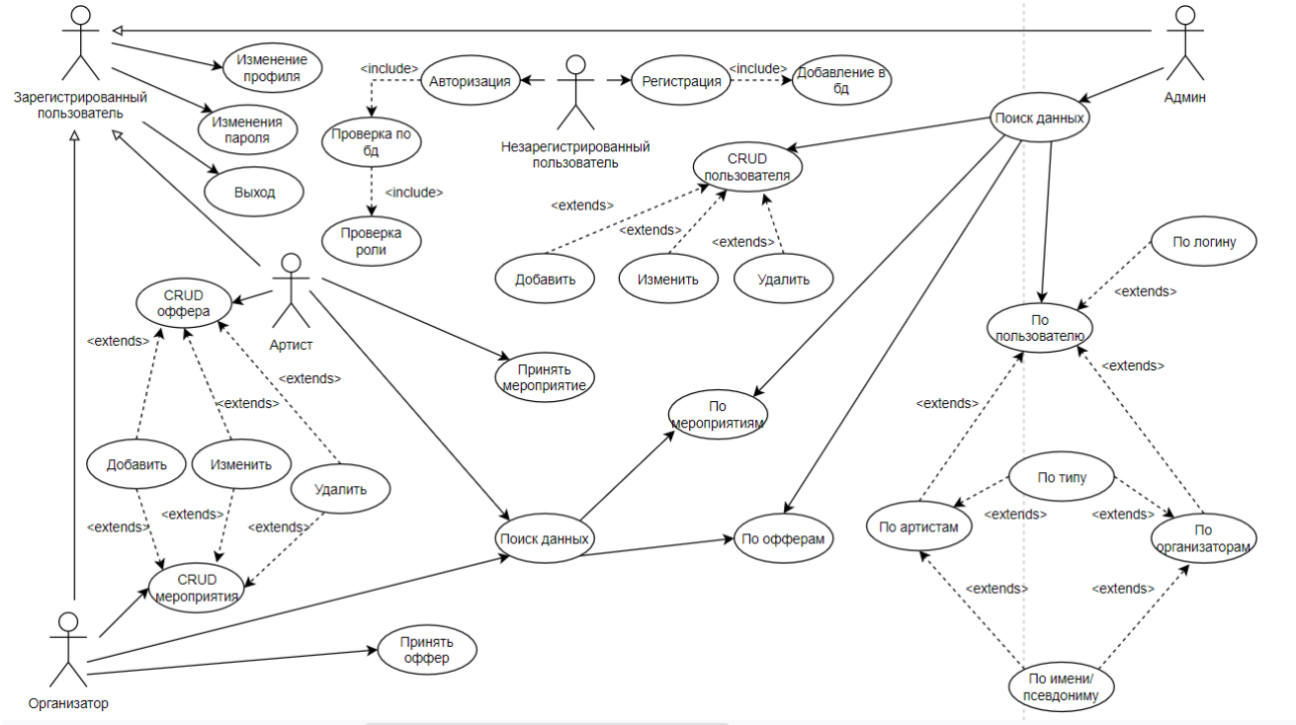



Рисунок 2.7 – Диаграмма вариантов использования «модуль статистического учета загруженности автостоянок»

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования. При этом пользователем (*actor*) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (*usecase*) служит для описания сервисов, которые система предоставляет пользователю. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с пользователем. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие пользователя с системой.

На *UML* диаграмме показаны наборы действий, предоставляемые спроектированным модулем, а также возможности взаимодействия менеджера или работника автостоянки с программным модулем.

## 2.6 Диаграмма взаимодействия

Диаграммы взаимодействия описывают поведение взаимодействующих групп объектов. Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках только одного варианта использования. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой.

Вариант использования «Создать автостоянку» представлен на рисунке 2.8.

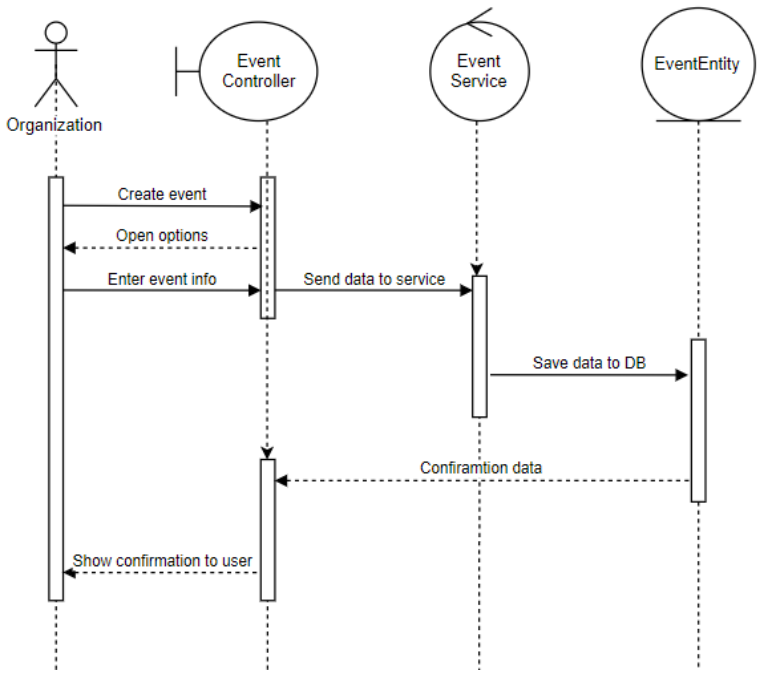



Рисунок 2.8 – Диаграмма последовательности «Создать автостоянку»

Пользователь (работник автостоянки) создает запрос на добавление новой автостоянки в базу данных. Далее пользователь заполняет информацию об автостоянке. После ввода данных на страницу они отправляются на сервер обрабатывающий сервис. После обработки данные записываются в соответствующую таблицу в базе данных. Ответом служит подтверждающее сообщение с основными данными об автостоянке, которое отображается на странице и передается пользователю.

## 2.7 Диаграмма состояний

Главное предназначение диаграммы состояний – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Чаще всего диаграммы состояний используются для описания поведения отдельных экземпляров классов (объектов), но они также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

На рисунке 2.9 представлена диаграмма состояний автостоянки

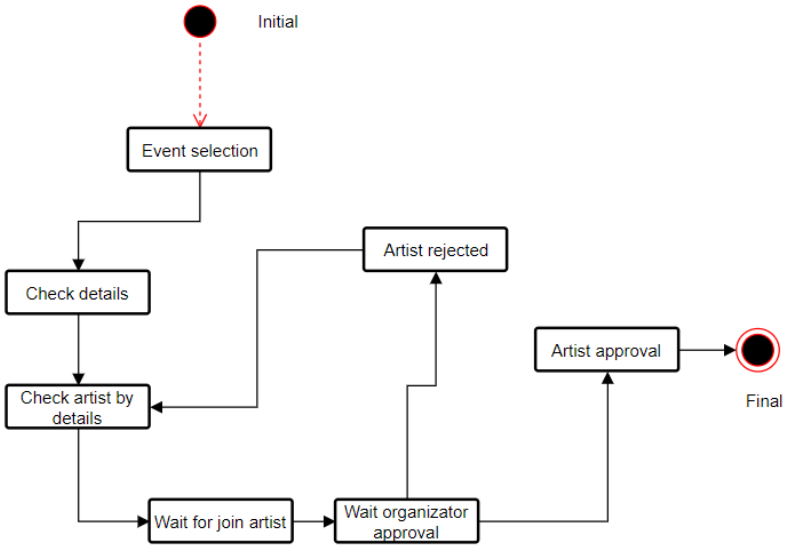



Рисунок 2.9 – Диаграмма состояний автостоянки

На данной диаграмме описаны состояния заказа в жизненном цикле приложения. Существует три последовательных состояния, описывающие наполненность автостоянки необходимыми данными. После получения всех необходимых данных и их отправки на сервер приложения мероприятие переходит в статус ожидания ответа. В случае успешного ответа автостоянка переходит в статус подтверждения, после чего в финальное состояние. В случае неуспешного ответа, автостоянка вернется в состояние подборки машины. Так же при полном наполнении автостоянка получается статус закрытой, в другом случае она открыта

## 2.8 Диаграмма деятельности

При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке *UML* используются так называемые диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов.

На рисунке 2.10 представлена диаграмма деятельности «Регистрация». На данной диаграмме показан процесс создания аннотации в рамках диаграммы деятельности.

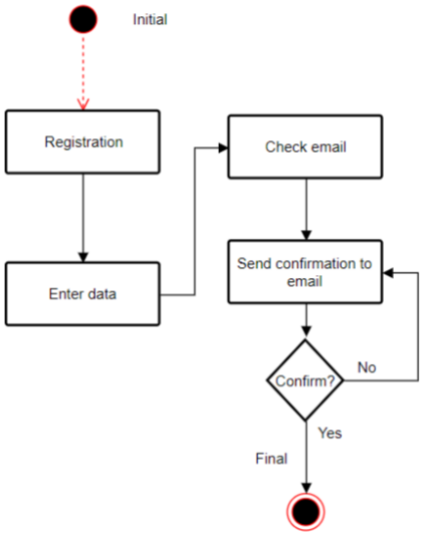



Рисунок 2.10 – Диаграмма деятельности «Регистрация»

Самым большим достоинством диаграмм деятельностей является поддержка параллелизма. Благодаря этому они являются мощным средством моделирования потоков работ и, по существу, параллельного программирования. Самый большой их недостаток заключается в том, что связи между действиями и объектами просматриваются не слишком четко.

Данный процесс можно описать так же, как он описан выше для предыдущих диаграмм, так как последовательность действий в нем не изменилась, а только отображена на другом виде диаграммы.

## 2.9 Диаграмма классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Диаграмма классов для сущностей представлена на рисунке 2.11.

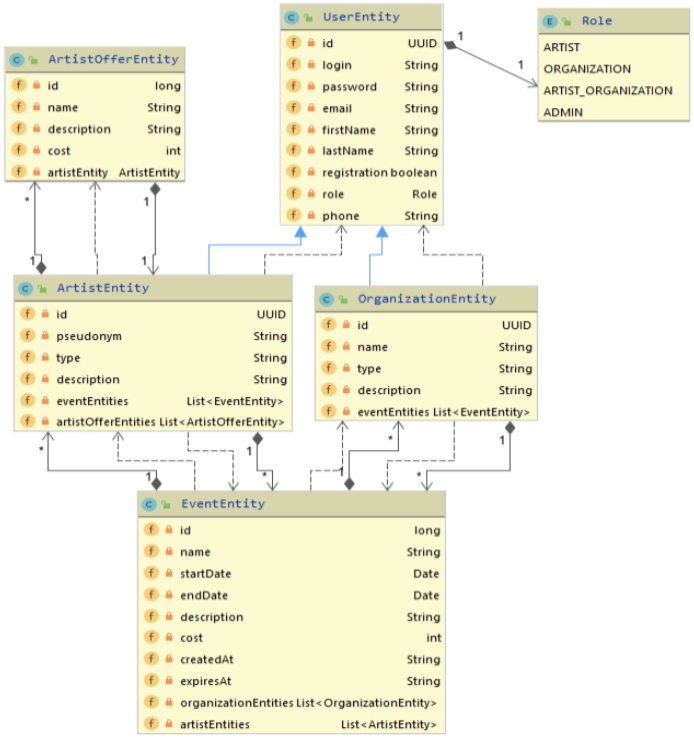



Рисунок 2.11 – Диаграмма классов сущностей

# 3 РЕАЛИЗАЦИОННАЯ ЧАСТЬ

## 3.1 Выбор и описание архитектуры системы

В данной системе используется трёхуровневая архитектура «клиент-сервер». Трёхуровневая архитектура – архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений).

Клиент (слой клиента) — это интерфейсный (обычно графический) компонент комплекса, предоставляемый конечному пользователю. Этот уровень не должен иметь прямых связей с базой данных (по требованиям безопасности и масштабируемости), быть нагруженным основной бизнес-логикой (по требованиям масштабируемости) и хранить состояние приложения (по требованиям надёжности). На этот уровень обычно выносится только простейшая бизнес-логика: интерфейс авторизации, алгоритмы шифрования, проверка вводимых значений на допустимость и соответствие формату, несложные операции с данными (сортировка, группировка, подсчёт значений), уже загруженными на терминал.

Сервер приложений (средний слой, связующий слой) располагается на втором уровне, на нём сосредоточена большая часть бизнес-логики. Вне его остаются только фрагменты, экспортируемые на клиента (терминалы), а также элементы логики, погруженные в базу данных (хранимые процедуры и триггеры). Реализация данного компонента обеспечивается связующим программным обеспечением. Серверы приложений проектируются таким образом, чтобы добавление к ним дополнительных экземпляров обеспечивало горизонтальное масштабирование производительности программного комплекса и не требовало внесения изменений в программный код приложения.

Сервер баз данных (слой данных) обеспечивает хранение данных и выносится на отдельный уровень, реализуется, как правило, средствами систем управления базами данных, подключение к этому компоненту обеспечивается только с уровня сервера приложений.

По сравнению с клиент-серверной или файл-серверной архитектурой трёхуровневая архитектура обеспечивает, как правило, большую масштабируемость (за счёт горизонтальной масштабируемости сервера

приложений и мультиплексирования соединений), большие возможности конфигураций (за счёт изолированности уровней друг от друга), более широкие возможности по обеспечению безопасности и отказоустойчивости. Кроме того, в сравнении с клиент-серверными приложениями, использующими прямые подключения к серверам баз данных, снижаются требования к скорости и стабильности каналов связи между клиентом и серверной частью.

Учитывая приведённое выше сравнение трёхуровневой архитектуры с другими видами и её преимущества, а также использование удалённой базы данных, большое количество пользователей, указывают на необходимость использования этого вида архитектуры. Также стоит учитывать возможность нестабильности сети интернет, при трёхуровневой архитектуре легче избежать ошибок, связанных с резкой потерей сети.

Трёхуровневая архитектура вынуждает создавать дополнительное связующее программное обеспечение (сервер), однако он позволяет перенести больше логики с клиентской части, тем самым улучшая производительность и быстродействие клиентского слоя. Также наличие сервера позволяет лучше обрабатывать ошибки, связанные с неправильными клиентскими действиями, что обеспечивает большую безопасность базы данных, так как некорректные данные не попадут в неё.

## 3.2 Обоснование выбора средств реализации

### 3.2.1 Выбор СУБД

Несмотря на то, что все системы управления базами данных выполняют одну и ту же основную задачу (т.е. дают возможность пользователям создавать, редактировать и получать доступ к информации, хранящейся в базах данных), сам процесс выполнения этой задачи варьируется в широких пределах. Кроме того, функции и возможности каждой СУБД могут существенно отличаться. Различные СУБД документированы по-разному: более или менее тщательно. По-разному предоставляется и техническая поддержка.

При сравнении различных популярных баз данных, следует учитывать, удобна ли для пользователя и масштабируема ли данная конкретная СУБД, а также убедиться, что она будет хорошо интегрироваться с другими продуктами,

которые уже используются. Кроме того, во время выбора следует принять во внимание стоимость системы и поддержки, предоставляемой разработчиком.

Сравниваются некоторые из популярных СУБД и на результате данного анализа выбирается подходящая для данного проекта.

«*Microsoft SQL Server*» - это система управления базами данных, движок которой работает на облачных серверах, а также локальных серверах, причем можно комбинировать типы применяемых серверов одновременно. Вскоре после выпуска *Microsoft SQL Server 2016, Microsoft* адаптировала продукт для операционной системы *Linux*, а на *Windows*-платформе он работал изначально. Одной из уникальных особенностей версии 2016 года является *temporal data support* (временная поддержка данных), которая позволяет отслеживать изменения данных с течением времени. Последняя версия *Microsoft SQL*-сервер поддерживает *dynamic data masking* (динамическую маскировку данных), которая гарантирует, что только авторизованные пользователи будут видеть конфиденциальные данные.

Достоинства:

– продукт очень прост в использовании;

– текущая версия работает быстро и стабильно;

– движок предоставляет возможность регулировать и отслеживать уровни производительности, которые помогают снизить использование ресурсов;

– вы сможете получить доступ к визуализации на мобильных устройствах;

– он очень хорошо взаимодействует с другими продуктами *Microsoft*.

Недостатки:

– цена для юридических лиц оказывается неприемлемой для большей части организаций;

– даже при тщательной настройке производительности корпорация SQL Server способен занять все доступные ресурсы;

– сообщается о проблемах с использованием службы интеграции для импорта файлов.

«*MongoDB*» - бесплатная база данных, которая имеет коммерческую версию - *MongoDB*, она предназначена для приложений, которые используют как структурированные, так и неструктурированные данные. Ядро является очень гибким и работает при подключении базы данных к приложениям через драйверы

*MongoDB*. Существует широкий выбор доступных драйверов, поэтому легко найти драйвер, который будет работать с требуемым языком программирования.

Поскольку изначально система *MongoDB* не была разработана для обработки моделей реляционных данных (хотя может это выполнять), могут возникнуть проблемы производительности, если вы попытаетесь использовать её таким образом. Однако, движок предназначен для обработки различных данных, которые нельзя отнести к реляционным, и может хорошо справляться там, где другие движки работают медленно или бессильны.

*MongoDB 3.2* - это последняя версия, и она имеет новую подключаемую систему движков хранения. Документы могут быть проверены в процессе обновления или выполнения вставок, а функции текстового поиска были улучшены. Новая способность частичного индексирования может привести к более высокой производительности, уменьшая размер индексов.

Достоинства:

– скорость и простота в использовании;

– движок поддерживает *json* и другие традиционные документы *NoSQL*;

– данные любой структуры могут быть сохранены/прочитаны быстро и легко.

Недостатки:

– *SQL* не используется в качестве языка запросов;

– инструменты для перевода *SQL*-запросов в *MongoDB* доступны, но их следует рассматривать именно как дополнение;

– программа установки может занять много времени.

«*MySQL*» - одна из самых популярных баз данных для веб-приложений. Фактически, является стандартом *de facto* для веб-серверов, которые работают под управлением операционной системы *Linux*. *MySQL* - это бесплатный пакет программ, однако новые версии выходят постоянно, расширяя функционал и улучшая безопасность. Существуют специальные платные версии, предназначенные для коммерческого использования. В бесплатной версии наибольший упор делается на скорость и надежность, а не на полноту функционала, который может стать и достоинством и недостатком - в зависимости от области внедрения.

Разработку и поддержку *MySQL* осуществляет корпорация *Oracle*, получившая права на торговую марку вместе с поглощённой *Sun Microsystems*, которая ранее приобрела шведскую компанию *MySQL AB*. Продукт

распространяется как под *GNU General Public License*, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

Эта СУБД позволяет выбирать различные движки для системы хранения, которые позволяют менять функционал инструмента и выполнять обработку данных, хранящихся в различных типах таблиц. Гибкость СУБД *MySQL* обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа *MyISAM*, поддерживающие полнотекстовый поиск, так и таблицы *InnoDB*, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД *MySQL* поставляется со специальным типом таблиц *EXAMPLE*, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и *GPL*-лицензированию, в СУБД *MySQL* постоянно появляются новые типы таблиц. Она также имеет простой в использовании интерфейс, и пакетные команды, которые позволяют удобно обрабатывать огромные объемы данных. Система невероятно надежна и не стремится подчинить себе все доступные аппаратные ресурсы.

Достоинства:

– распространяется бесплатно;

– прекрасно документирована;

– предлагает много функций, даже в бесплатной версии;

– пакет *MySQL* включен в стандартные репозитории наиболее распространённых дистрибутивов операционной системы *Linux*, что позволяет устанавливать её элементарно;

– поддерживает набор пользовательских интерфейсов;

– может работать с другими базами данных, включая *DB2* и *Oracle*.

Недостатки:

– придётся потратить много времени и усилий, чтобы заставить *MySQL* выполнять несложные задачи, хотя другие системы делают это автоматически, например: создавать инкрементные резервные копии;

– отсутствует встроенная поддержка *XML* или *OLAP;*

– для бесплатной версии доступна только платная поддержка;

– идеально подходит для: организаций, которым требуется надежный инструмент управления базами данных, но бесплатный.

«*CDS-view*» является одним из нескольких бесплатных и одновременно встроенных в *SAP Logon* вариантов СУБД, часто используется для ведения баз данных веб-сайтов. Это была одна из первых разработанных систем управления базами данных, поэтому в настоящее время она хорошо развита, и позволяет пользователям управлять как структурированными, так и неструктурированными данными. Может быть использован на большинстве основных платформ, включая *Linux*. Прекрасно справляется с задачами импорта информации из других типов баз данных с помощью собственного инструментария.

Движок БД может быть размещен в ряде сред, в том числе виртуальных, физических и облачных. Самая свежая версия, *CDS scc20*, предлагает обработку больших объемов данных и увеличение числа одновременно работающих пользователей.

Достоинства:

– является масштабируемым и способен обрабатывать терабайты данных;

– поддерживает формат *json*;

– существует множество предопределенных функций;

– доступен ряд интерфейсов.

Недостатки:

– документация туманна, поэтому, возможно, ответы на некоторые вопросы придется искать в интернете;

– конфигурация может смутить неподготовленного пользователя;

– скорость работы может падать во время проведения пакетных операций или выполнения запросов чтения;

– идеально подходит для организаций с ограниченным бюджетом, но квалифицированными специалистами, когда требуется возможность выбрать свой интерфейс и использовать *json*.

База данных данного проекта не требует очень сложных операций над данными, необходим удобный рабочий интерфейс для работы с базой данных, проект предусматривает создание реляционной базы данных. Учитывая приведённый выше анализ различных СУБД и опыт работы с несколькими СУБД разработчика системы, что так же является важным фактором, так как от этого зависит качество созданной базы данных и скорость её созданий, выбирается СУБД *CDS*.

### 3.2.2 Выбор языка программирования

Несмотря на то, что все языки программирования, направленные на работу с веб-технологиями выполняют практически одинаковые функции, сам процесс выполнения этой задачи варьируется в широких пределах. Кроме того, функции и возможности каждого языка программирования могут существенно отличаться. Различные языки программирования документированы по-разному: более или менее тщательно. По-разному предоставляется и техническая поддержка.

В больших компаниях для эффективной работы необходимо использовать сервисы и системы, разработанные с помощью новейший технологий. Для улучшения работы компании, возникла необходимость в создании веб-приложения для управления машинами на автостоянке. Для работы с серверной частью приложения были использованы следующие технологии:

– *CDS-view*;
– *BOPF*;
– *FPM*.

Начиная с версии *ABAP 7.4* стала доступна новая технология описания моделей данных в словаре – *ABAP CDS* (*CDS – Core Data Services*) [1]. Данная технология позволяет описывать модели данных на более продвинутом уровне, нежели это можно было делать стандартными словарными вьюшками. Она так же позволяет оптимизировать работу с данными за счёт вынесения вычислений на уровень СУБД (*Code-to-Data*), что актуально для *HANA*.

Рассмотрим основные понятия в контексте *ABAP CDS* (Рисунок 3.2.1):

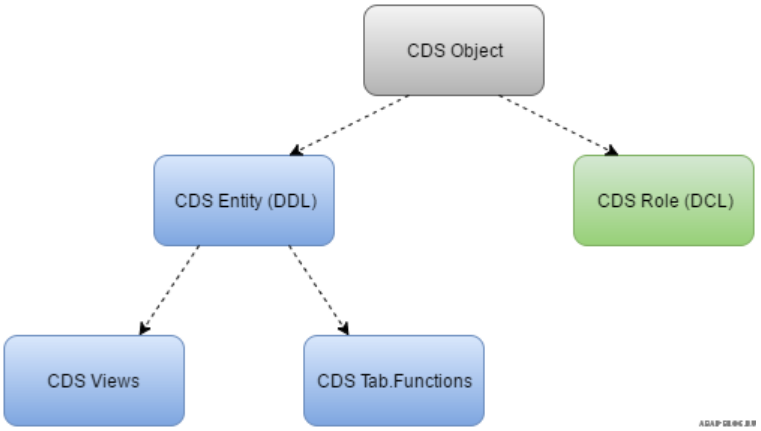



Рисунок 3.2.1 – Основные понятия *CDS View*

– *CDS Object* – обобщенное понятие объекта репозитория, создаваемого через исходный код *CDS*. К *CDS Object* относятся либо *CDS Entity* (*DDL*), либо *CDS Role* (*DCL*);

– *CDS Entity* – это *CDS* объект, управляемый *ABAP* словарём, создаваемый путём описания на специальном языке – *Data Defenition Language* (*DDL*). Объект не переносится транспортом, вместо этого он генерируется в системе во время активации исходного кода *DDL* с которым он связан;

– *CDS View* – это основной объект с которым мы имеем дело в *ABAP* при чтении данных из словаря. На *CDS View* можно ссылаться при объявлении *ABAP* переменных, однако этот объект нельзя использовать при описании словарных объектов. Каждая новая *CDS View* должна быть описана в отдельном *DDL* объекте. Имя *CDS View* определяется сразу после ключевого слова *DEFINE VIEW* в *DDL* (пример ниже), рекомендуется использовать имя такое же как у *DDL* объекта. Создаются они на базе существующих источников данных из словаря – таблиц (поддерживаются только прозрачные таблицы, пул таблиц и кластерные таблицы не поддерживаются), классических словарных вьюшек и других *CDS*. *CDS Entity* не доступны из *Native SQL*, а *OpenSQL* поддерживает только операции чтения;

– *CDS Role* – объекты описывающие необходимые проверки полномочий для доступа *к CDS View*. *CDS Role* создаются путём их описания в *DCL* (*Data control language*) объекте;

– *CDS Database View* – объект словаря, который как и *CDS View* генерируется во время активации *DDL* объекта. В *ABAP* можно считать данные напрямую из *CDS DB View*, однако это не рекомендуется делать и данный способ уже считается устаревшим в *ABAP* 7.5. В отличие от *CDS VIew, CDS DB View* можно использовать при описании объектов словаря.

*BOPF Framework* - среда разработки *ABAP*, которая используется в командах разработчиков *SAP*. Эта технология позволяет делать пользовательскую разработку быстрее, стабильнее, а приложения, разработанные с помощью *BOPF*, имеют схожую архитектуру. *BOPF* является частью *Business Suite Foundation*.

На рисунке 3.2.2 показана структура *BOPF*.

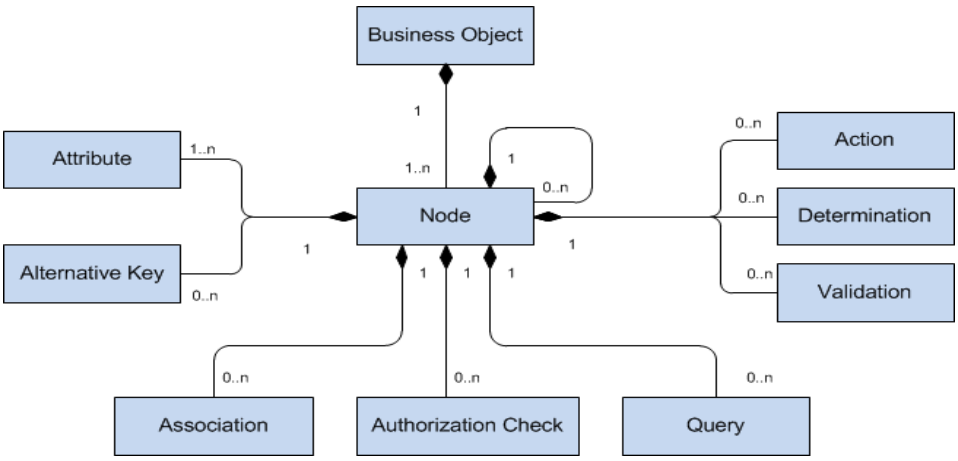



Рисунок 3.2.2 – Структура *BOPF*

В качестве базы данных была выбрана *SAP HANA* и технология *OData* для ее управления. Сервис *OData* организует данные в форме сущностей с набором свойств, объединённых взаимосвязями. Эти элементы напоминают элементы моделей данных *CDS*, поэтому модели данных *CDS* являются идеальным средством вывода сервисов *OData*[2].

Чтобы понять структуру сервиса *OData* необходимо посмотреть на сервисный документ и документ метаданных сервиса. Сервисный документ содержит список сущностей или ресурсов, к которым можно обращаться посредством данного сервиса или запрашивать посредством /*sap*/*opu*/*odata*/*sap*/<*OData_service_name*>/. Кроме того, определить, позволяет ли сервис создавать, изменять или удалять сущности, можно по атрибутам *sap:creatable*, sap:updatable и *sap:deletable* тега <*app:collection*>, который также содержит относительную ссылку на набор сущностей через его атрибут *href* (см. рис. 3.2.3).

Сервисный документ метаданных содержит намного более подробные данные, чем сервисный документ, и отражает все метаданные сервиса. Для его запроса используется опция *$metadata*:/*sap*/*opu*/*odata*/*sap*/<*OData_service_name*>/*$metadata*. С её помощью можно просмотреть все сущности сервиса вместе с их свойствами и связями.

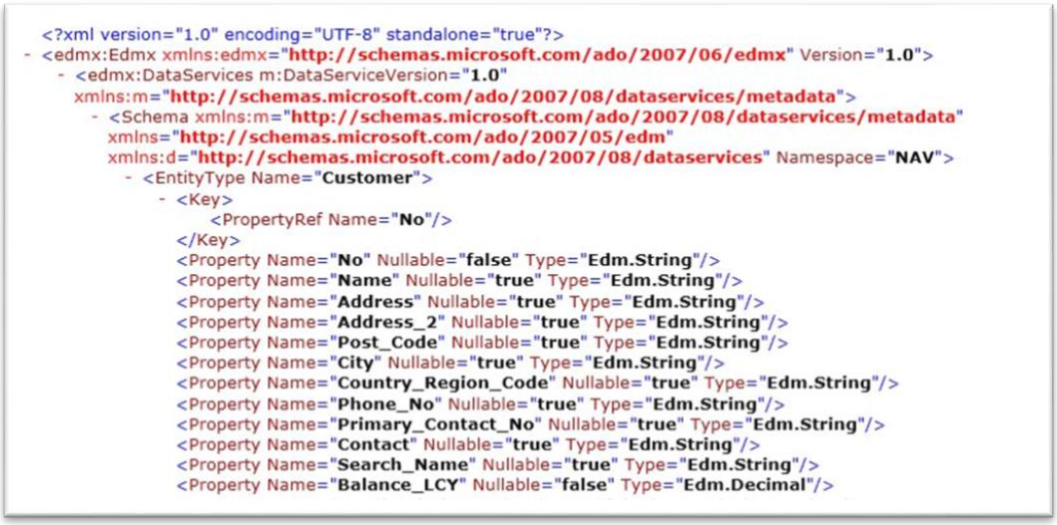



Рисунок 3.2.3 – Сервисный документ *OData*

*OData* использует команды *REST HTTP POST, GET, PUT* и *DELETE* для создания, считывания, обновления и удаления сущностей (*CRUD*). *OData* также определяет простой, но мощный язык запросов для ограничения набора результатов, предоставляемого *SAP Gateway*.

Основным ядром в *SAP HANA* является компонент СУБД, позволяющий обрабатывать большие объёмы данных с помощью технологии *In-Memory* и на базе языкового инструмента *SQL*[3]. В основе СУБД *SAP HANA* используется реляционная модель данных, но также существует возможность обращения к данным с помощью «графового» языка запросов *WIPE*. *SAP HANA* включает в себя четыре компонента-продукта: *SAP Data Hub, SAP HANA, SAP Enterprise Architecture Designer* и *SAP Cloud Platform Big Data Services*[4].

Сочетание этих решений позволяет создать целостную структуру управления данными с следующими функциями:

- отслеживание происхождения данных;
- отслеживание изменений в данных и их структуре;
- комплексное понимание метаданных;
- поддержка необходимого уровня безопасности;
- централизованный мониторинг.

Фундаментальная характеристика объектно-реляционной базы данных — это поддержка пользовательских объектов и их поведения, включая типы данных, функции, операции, домены и индексы. Это делает *SAP HANA*

невероятно гибким и надежным. Среди прочего, он умеет создавать, хранить и извлекать сложные структуры данных.

Существует обширный список типов данных, которые поддерживает *SAP HANA. SAP HANA* поддерживает 7 категорий типов данных *SQL*, и это зависит от типа данных, которые вы должны хранить в столбце:

- числовой;
- персонаж / Строка;
- логический;
- дата Время;
- двоичный;
- большие объекты;
- *multi*-значных.

Функционал данного проекта будет реализован на языке программирования Java. Бизнес логика будет реализована с помощью фреймворка Spring. Так же в качестве интерфейса взаимодействия пользователя с приложением будет использоваться фреймворк Swagger. Ниже приведены описания данных технологий, и их преимущества.

## 3.3 Описание базы данных

Базой данных (БД) называется организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность сведений об объектах, процессах, событиях или явлениях, относящихся к некоторой предметной области, теме или задаче. Она организована таким образом, чтобы обеспечить информационные потребности пользователей, а также удобное хранение этой совокупности данных, как в целом, так и любой ее части.

Реляционная база данных представляет собой множество взаимосвязанных таблиц (сущностей), каждая из которых содержит информацию об объектах определенного вида. Каждая строка таблицы содержит данные об одном объекте, а столбцы таблицы содержат различные характеристики этих объектов - атрибуты.

База данных предназначена для хранения данных о пользователях (организаторах, артистах, администраторах). Функции системы, связанные с использованием БД: регистрация пользователей;

Для реализации *web*-приложения «Управления автостоянкой машин» были выявлены такие сущности, как *Park* (автостоянка), *Car* (машина). На рисунке 3.3.1 показана структура проекта автостоянки.

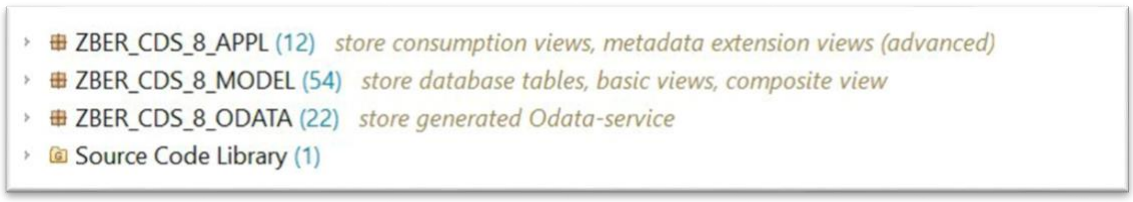
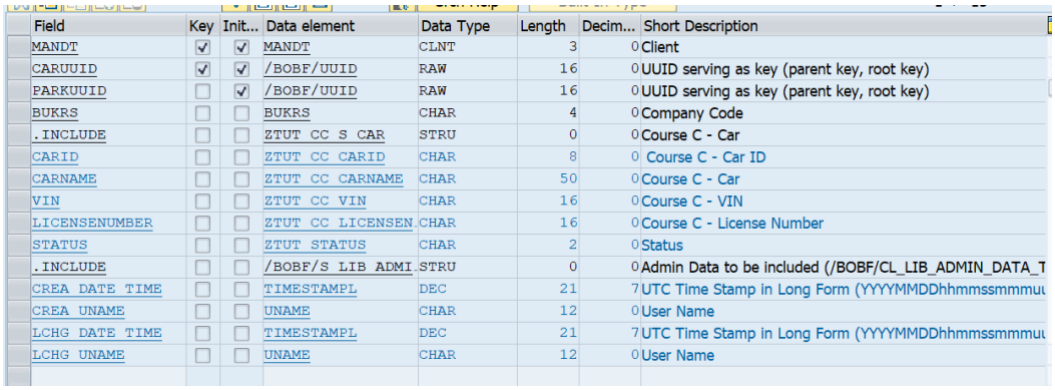



Рисунок 3.3.1 – Структура проекта автостоянки

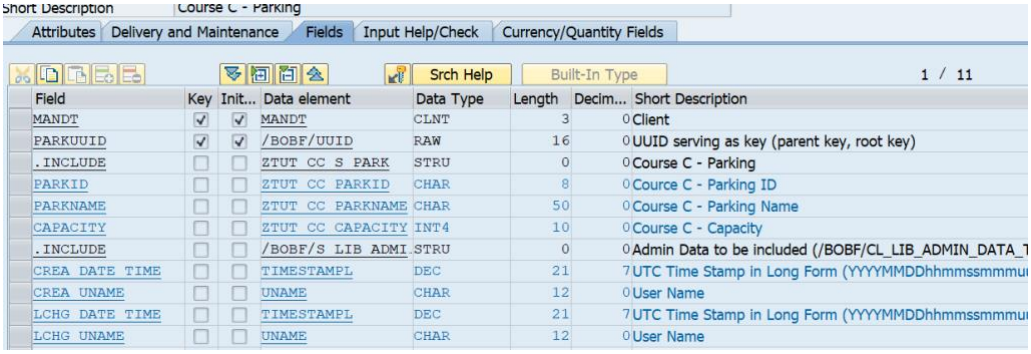
В таблице *Car* есть следующие поля:



| Field | Key | Init... | Data element | Data Type | Length | Decim... | Short Description |
|---|---|---|---|---|---|---|---|
| MANDT | ☑ | ☑ | MANDT | CLNT | 3 | 0 | Client |
| CARUUID | ☑ | ☑ | /BOBF/UUID | RAW | 16 | 0 | UUID serving as key (parent key, root key) |
| PARKUUID | ☐ | ☑ | /BOBF/UUID | RAW | 16 | 0 | UUID serving as key (parent key, root key) |
| BUKRS | ☐ | ☐ | BUKRS | CHAR | 4 | 0 | Company Code |
| .INCLUDE | ☐ | ☐ | ZTUT_CC_S_CAR | STRU | 0 | 0 | Course C - Car |
| CARID | ☐ | ☐ | ZTUT_CC_CARID | CHAR | 8 | 0 | Course C - Car ID |
| CARNAME | ☐ | ☐ | ZTUT_CC_CARNAME | CHAR | 50 | 0 | Course C - Car |
| VIN | ☐ | ☐ | ZTUT_CC_VIN | CHAR | 16 | 0 | Course C - VIN |
| LICENSENUMBER | ☐ | ☐ | ZTUT_CC_LICENSEN | CHAR | 16 | 0 | Course C - License Number |
| STATUS | ☐ | ☐ | ZTUT_STATUS | CHAR | 2 | 0 | Status |
| .INCLUDE | ☐ | ☐ | /BOBF/S_LIB_ADMI | STRU | 0 | 0 | Admin Data to be included (/BOBF/CL_LIB_ADMIN_DATA_T |
| CREA_DATE_TIME | ☐ | ☐ | TIMESTAMPL | DEC | 21 | 7 | UTC Time Stamp in Long Form (YYYYMMDDhhmmssmmmuu |
| CREA_UNAME | ☐ | ☐ | UNAME | CHAR | 12 | 0 | User Name |
| LCHG_DATE_TIME | ☐ | ☐ | TIMESTAMPL | DEC | 21 | 7 | UTC Time Stamp in Long Form (YYYYMMDDhhmmssmmmuu |
| LCHG_UNAME | ☐ | ☐ | UNAME | CHAR | 12 | 0 | User Name |

Рисунок 3.3.2 – Структура сущности *Car*

В таблице *Park*:



Short Description Course C - Parking

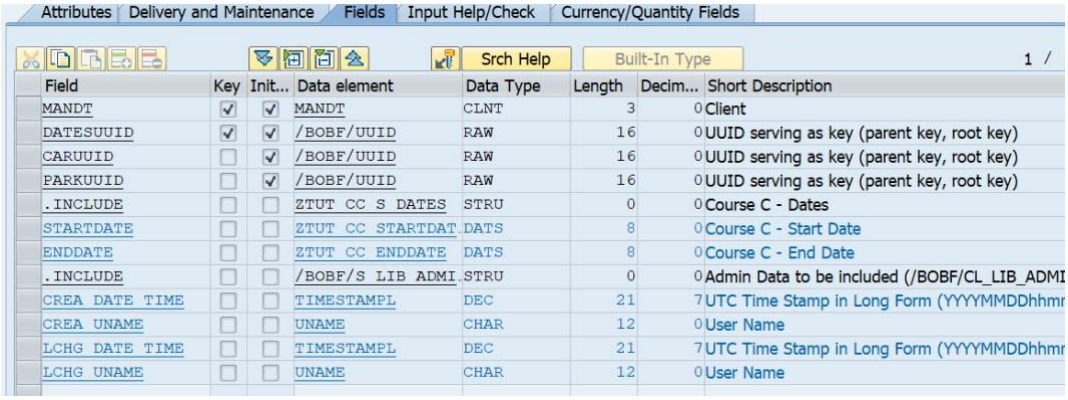
Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields

Srch Help | Built-In Type | 1 / 11

| Field | Key | Init... | Data element | Data Type | Length | Decim... | Short Description |
|---|---|---|---|---|---|---|---|
| MANDT | ☑ | ☑ | MANDT | CLNT | 3 | 0 | Client |
| PARKUUID | ☑ | ☑ | /BOBF/UUID | RAW | 16 | 0 | UUID serving as key (parent key, root key) |
| .INCLUDE | ☐ | ☐ | ZTUT_CC_S_PARK | STRU | 0 | 0 | Course C - Parking |
| PARKID | ☐ | ☐ | ZTUT_CC_PARKID | CHAR | 8 | 0 | Cource C - Parking ID |
| PARKNAME | ☐ | ☐ | ZTUT_CC_PARKNAME | CHAR | 50 | 0 | Course C - Parking Name |
| CAPACITY | ☐ | ☐ | ZTUT_CC_CAPACITY | INT4 | 10 | 0 | Course C - Capacity |
| .INCLUDE | ☐ | ☐ | /BOBF/S_LIB_ADMI | STRU | 0 | 0 | Admin Data to be included (/BOBF/CL_LIB_ADMIN_DATA_T |
| CREA_DATE_TIME | ☐ | ☐ | TIMESTAMPL | DEC | 21 | 7 | UTC Time Stamp in Long Form (YYYYMMDDhhmmssmmmuu |
| CREA_UNAME | ☐ | ☐ | UNAME | CHAR | 12 | 0 | User Name |
| LCHG_DATE_TIME | ☐ | ☐ | TIMESTAMPL | DEC | 21 | 7 | UTC Time Stamp in Long Form (YYYYMMDDhhmmssmmmuu |
| LCHG_UNAME | ☐ | ☐ | UNAME | CHAR | 12 | 0 | User Name |

Рисунок 3.3.3 – Структура таблицы *Parks*

Поля сущности *Dates*:




| Field | Key | Init... | Data element | Data Type | Length | Decim... | Short Description |
|---|---|---|---|---|---|---|---|
| MANDT | ☑ | ☑ | MANDT | CLNT | 3 | 0 | Client |
| DATESUUID | ☑ | ☑ | /BOBF/UUID | RAW | 16 | 0 | UUID serving as key (parent key, root key) |
| CARUUID | ☐ | ☑ | /BOBF/UUID | RAW | 16 | 0 | UUID serving as key (parent key, root key) |
| PARKUUID | ☐ | ☑ | /BOBF/UUID | RAW | 16 | 0 | UUID serving as key (parent key, root key) |
| .INCLUDE | ☐ | ☐ | ZTUT_CC_S_DATES | STRU | 0 | 0 | Course C - Dates |
| STARTDATE | ☐ | ☐ | ZTUT_CC_STARTDAT | DATS | 8 | 0 | Course C - Start Date |
| ENDDATE | ☐ | ☐ | ZTUT_CC_ENDDATE | DATS | 8 | 0 | Course C - End Date |
| .INCLUDE | ☐ | ☐ | /BOBF/S_LIB_ADMI | STRU | 0 | 0 | Admin Data to be included (/BOBF/CL_LIB_ADMI |
| CREA_DATE_TIME | ☐ | ☐ | TIMESTAMPL | DEC | 21 | 7 | UTC Time Stamp in Long Form (YYYYMMDDhhmr |
| CREA_UNAME | ☐ | ☐ | UNAME | CHAR | 12 | 0 | User Name |
| LCHG_DATE_TIME | ☐ | ☐ | TIMESTAMPL | DEC | 21 | 7 | UTC Time Stamp in Long Form (YYYYMMDDhhmr |
| LCHG_UNAME | ☐ | ☐ | UNAME | CHAR | 12 | 0 | User Name |

Рисунок 3.3.4 – Структура *Dates*

Требуется соблюсти соответствие базы данных трём нормальным формам. Нормальная форма – требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц).

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа (ПК).

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых относится к нескольким записям таблицы, в отдельные таблицы.

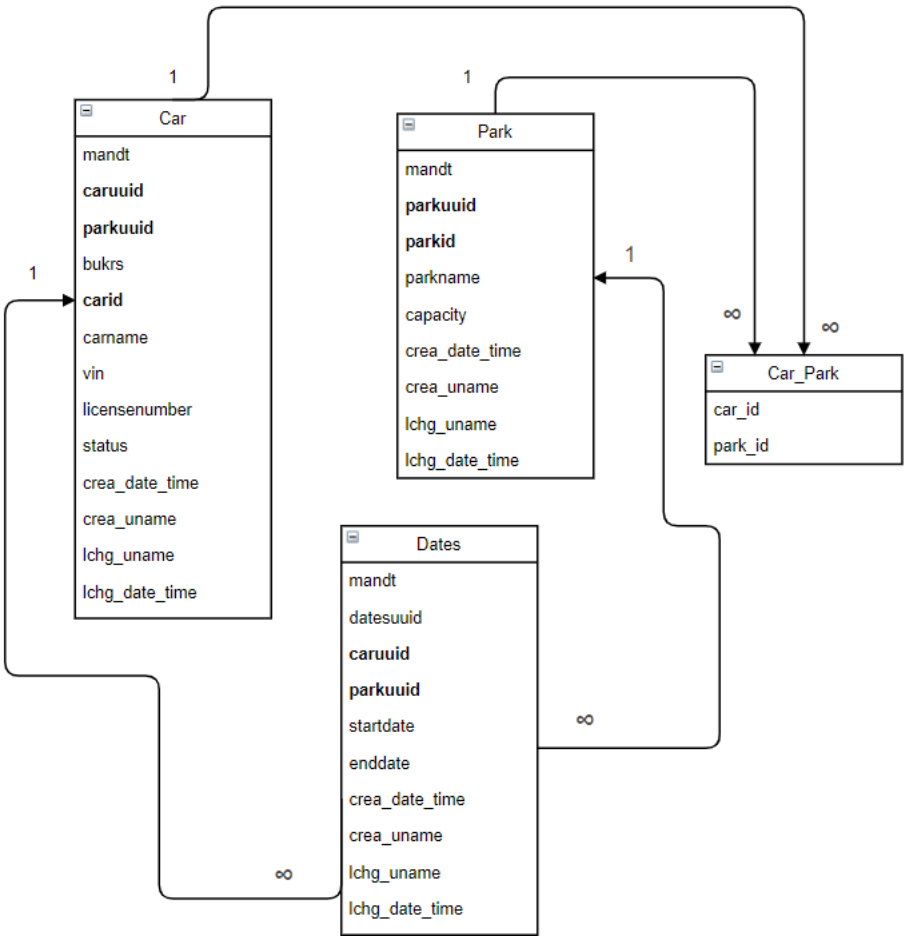



Рисунок 3.3.5 – Схема базы данных

### 3.4 Описание программы

На уровне данных были созданы *Basic Views* для этих сущностей (Рисунок 3.4.1). Создаётся база данных при помощи технологии *CDS View*. Для созданий таблиц, их ключей и внешних ключей не требуется прописывать код вручную.

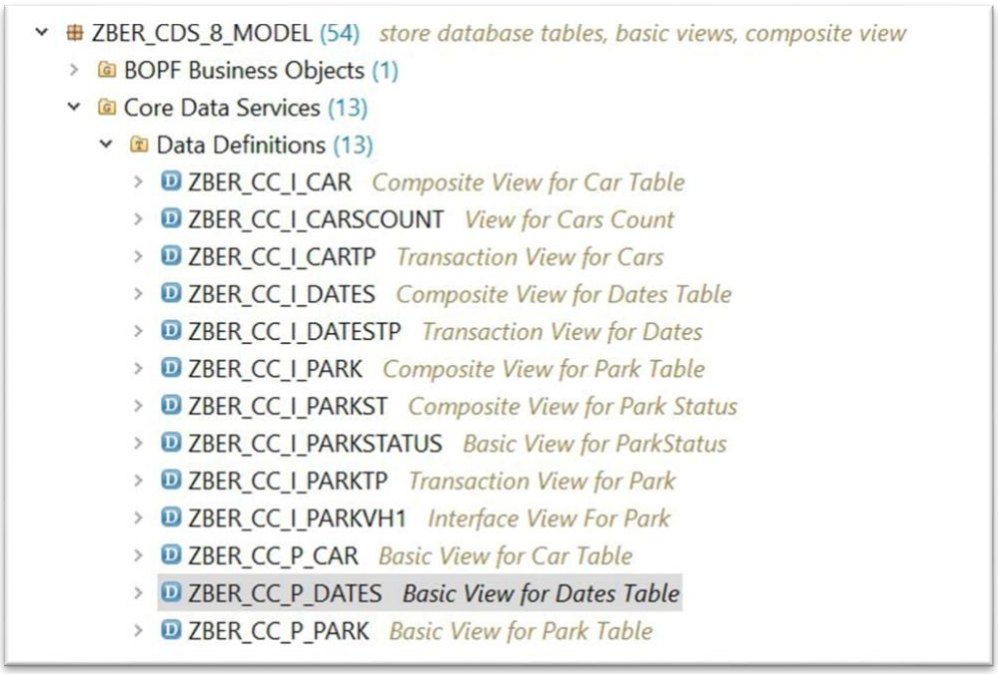



Рисунок 3.4.1 – *Basic CDS Views* для главных сущностей.

На рисунке 3.4.2 продемонстрирован код для *basic view* для сущности *Park*.

```
@AbapCatalog.sqlViewName: 'ZBERCCPPARK'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Basic View for Park Table'

/* @VDM: {
  viewType: #BASIC
}
*/

define view ZBER_CC_P_PARK
  as select from zber_cc_park
{

  key parkuuid        as ParkUUID,
      parkid          as ParkID,
      parkname        as ParkName,
      capacity        as Capacity,
      crea_date_time,
      crea_uname,
      lchg_date_time,
      lchg_uname

}
```

Рисунок 3.4.2 – *Basic View* для сущности *Park*

На рисунке 3.4.3 продемонстрирован код для *basic view* для сущности *Car*.

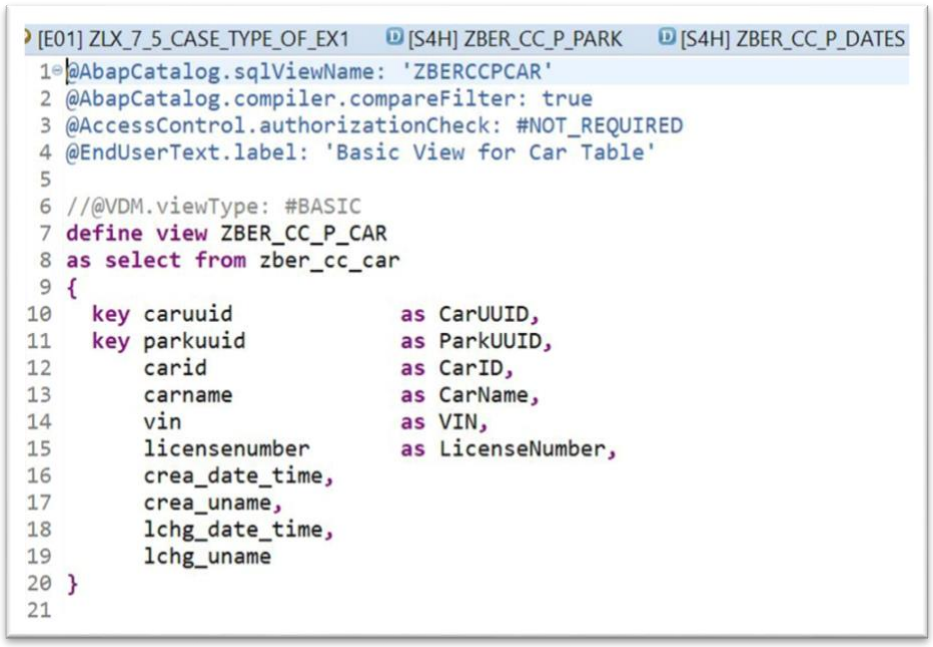



Рисунок 3.4.3 – *Basic View* для сущности *Car*.

Также были разработаны *Consumption View*, *Composite View* и *Transaction View*. На рисунке 4.1.4 продемонстрирован часть кода *Consumption View*.

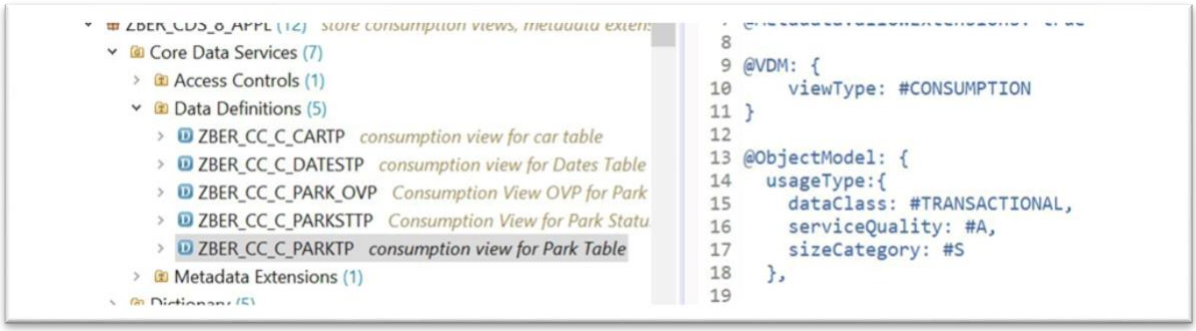



Рисунок 3.4.4 – *Consumption View* для сущности *Park*

На основе *CDS Views* был сгенерирован *OData Service* – бэкэнд часть приложения. На рисунке 3.4.5. можно увидеть *Entity Types* и *Entity Sets*, полученные в результате созданных *CDS Views.*

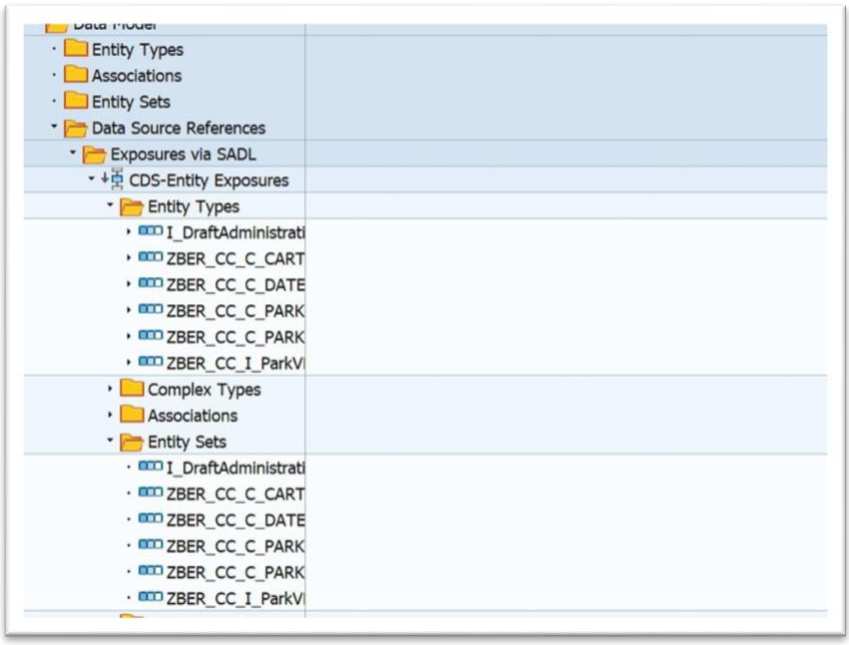



Рисунок 3.4.5 – Структура *OData* сервиса

Также был сгенерирован *BOPF* объект на основе *CDS View*. Для этого необходимо перейти к транзакции *BOBX*:

Мы видим здесь три категории *BO*, а именно. Пользовательские улучшения бизнес-объектов, *SAP Business Object* и *Business Object*, которые не требуют пояснений (рисунок 3.4.6).

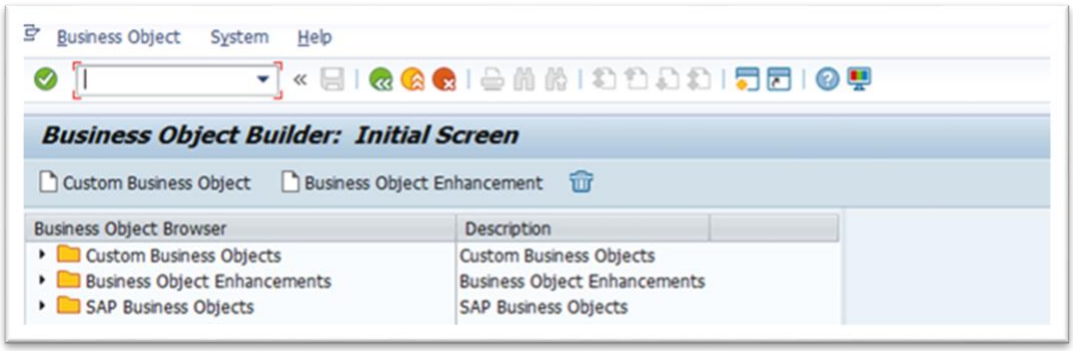



Рисунок 3.4.6 – Категории бизнес объекта

Создали пользовательский бизнес-объект: мастер помогает нам создать собственный бизнес-объект (рисунок 3.4.7).

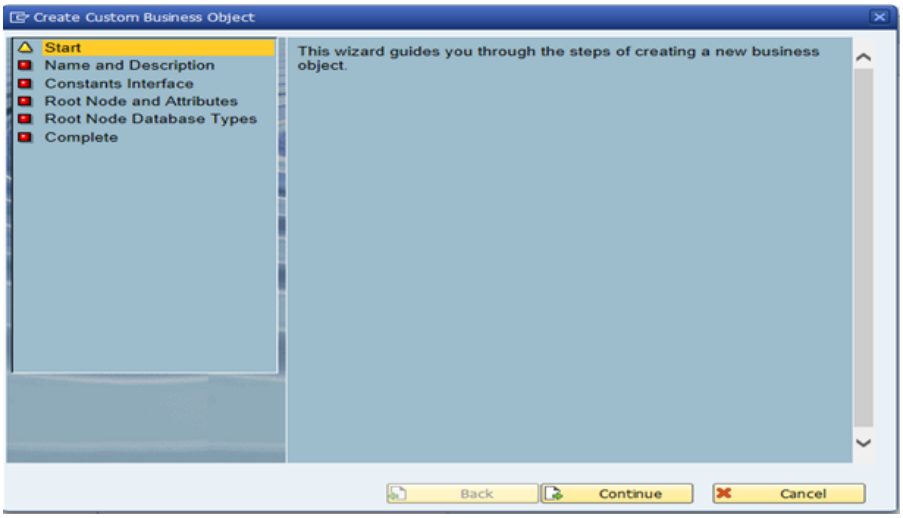



Рисунок 3.4.7 – Мастер создания бизнес объекта

При нажатии на кнопку *Complete*, *BO* будет создан (рисунок 3.4.8).

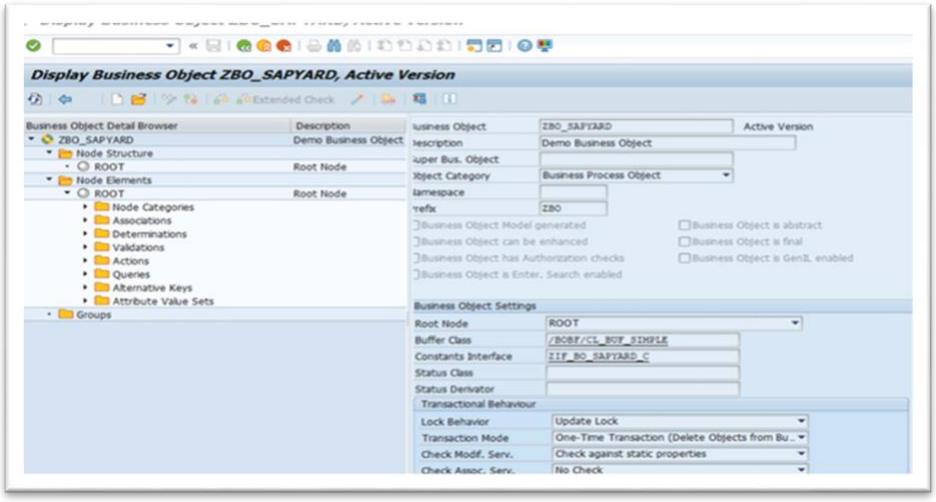



Рисунок 3.4.8 – Созданный бизнес объект

При нажатии на *BO* можно увидеть все детали бизнес-объекта, а именно структурную модель, процедурную модель (элементы узла), постоянный интерфейс и т.д. (рисунок 3.4.9)

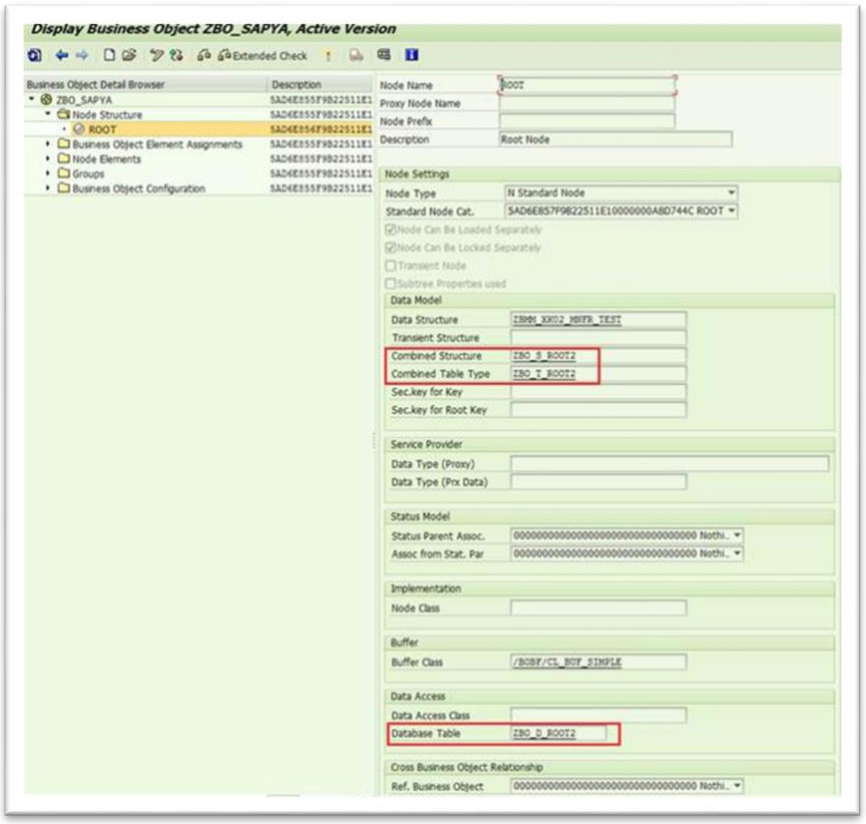



Рисунок 3.4.9 – Детали бизнес объекта

В транзакции *BOBF* демонстрируется созданный бизнес объект. (рисунок 3.4.10)

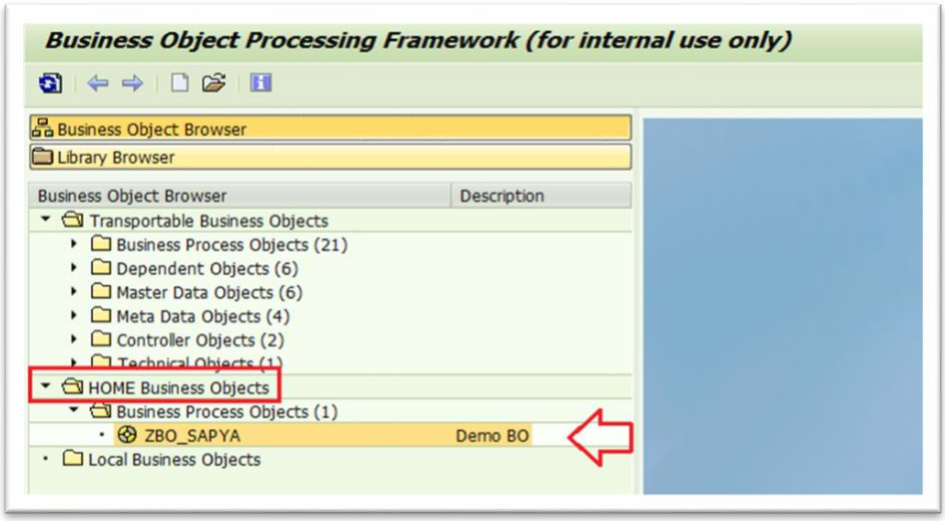



Рисунок 3.4.10 – Транзакция *BOBF* и созданный бизнес объект

Для дальнейшего использования БО для работы с базой данных, были созданы запросы, валидации, детерминации, ассоциации и действия. Все эт помогает отслеживать корректность данных и автоматическое заполнение полей. Рисунок 3.4.11.

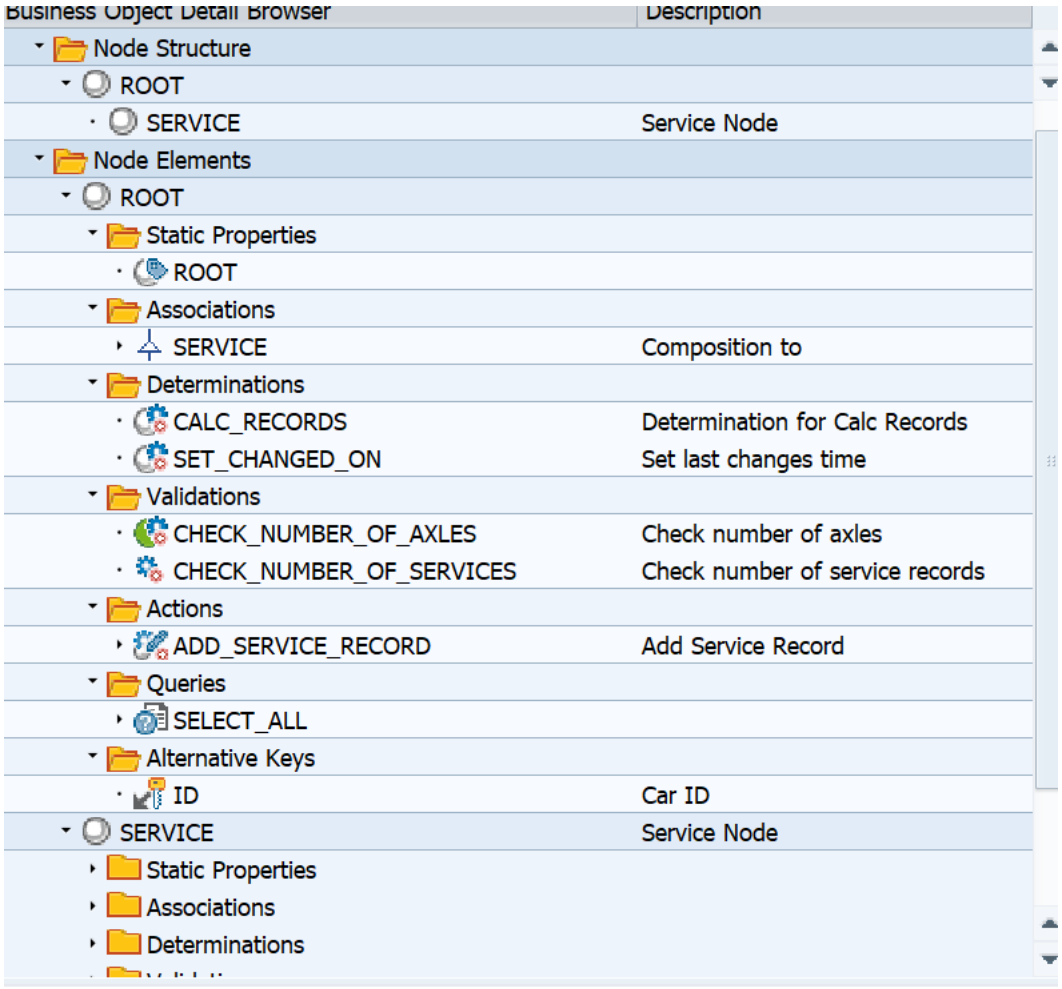



Рисунок 3.4.11 – Структура созданного бизнес объекта

### 3.5 Руководство пользователя

Ниже приведен алгоритм действий пользователя при работе с готовым продуктом.

При входе на сайт пользователь попадает на стартовую страницу (рисунок 3.5.1).

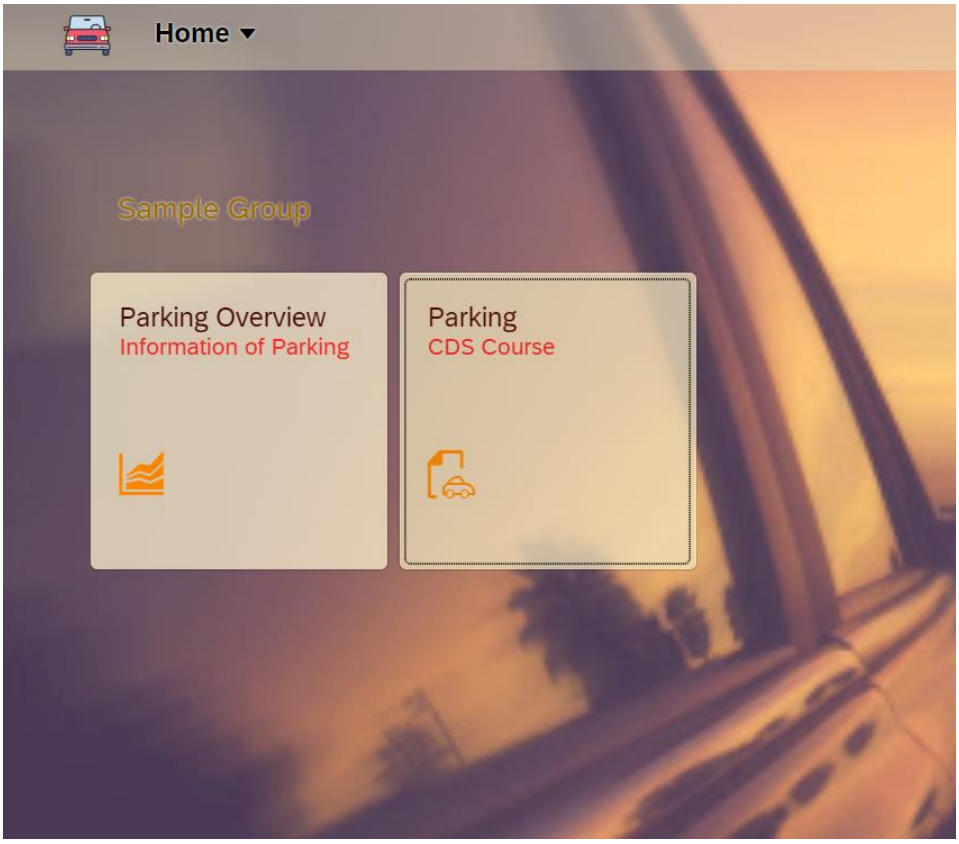



Рисунок 3.5.1 – Домашняя страница сайта

При выполнении любого действия, появится форма для авторизации (рисунок 3.5.2).

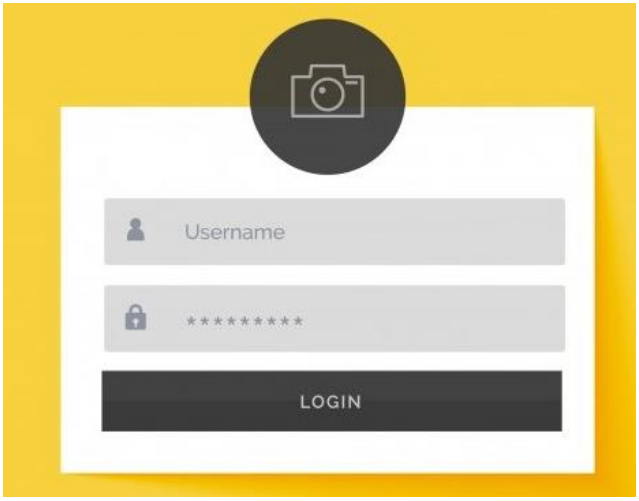



Рисунок 3.5.2 – Форма авторизации

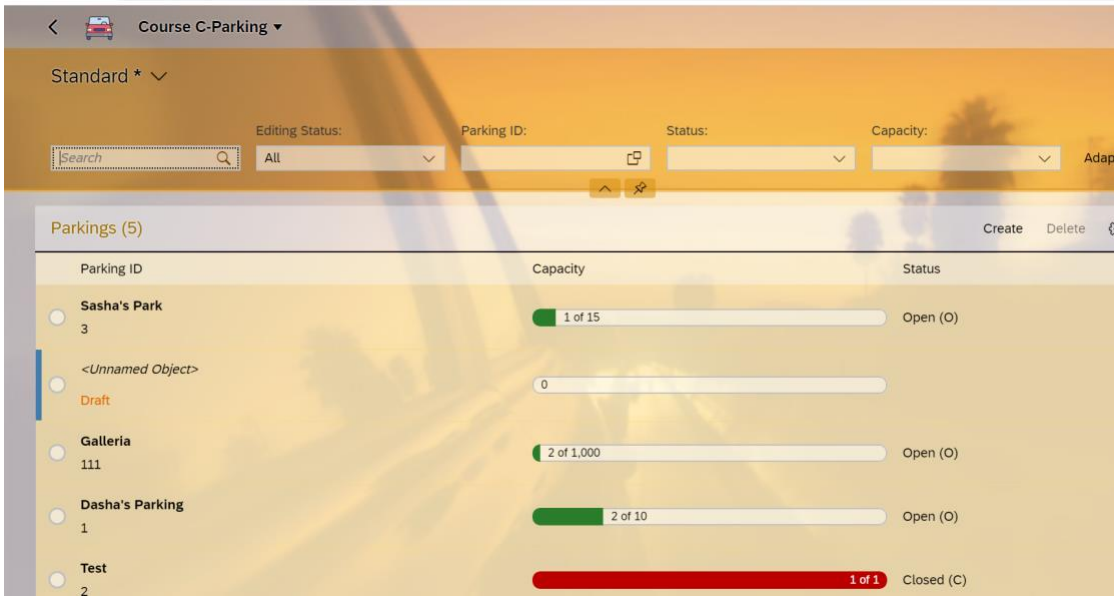



Рисунок 3.5.3 – Отображение всех автостоянок

Также реализованы функции создания и удаления автостоянкой для машин. Второй блок содержит отображение всех выбранных автостоянок, включая наглядное представления заполненности автостоянки машинами.

При нажатии на строчку конкретной автостоянки, осуществляется переход на вторую страницу, где можно увидеть информацию о автостоянку, машины, находящиеся в нем, а также внести новые машины на автостоянку(рисунок 3.5.4).

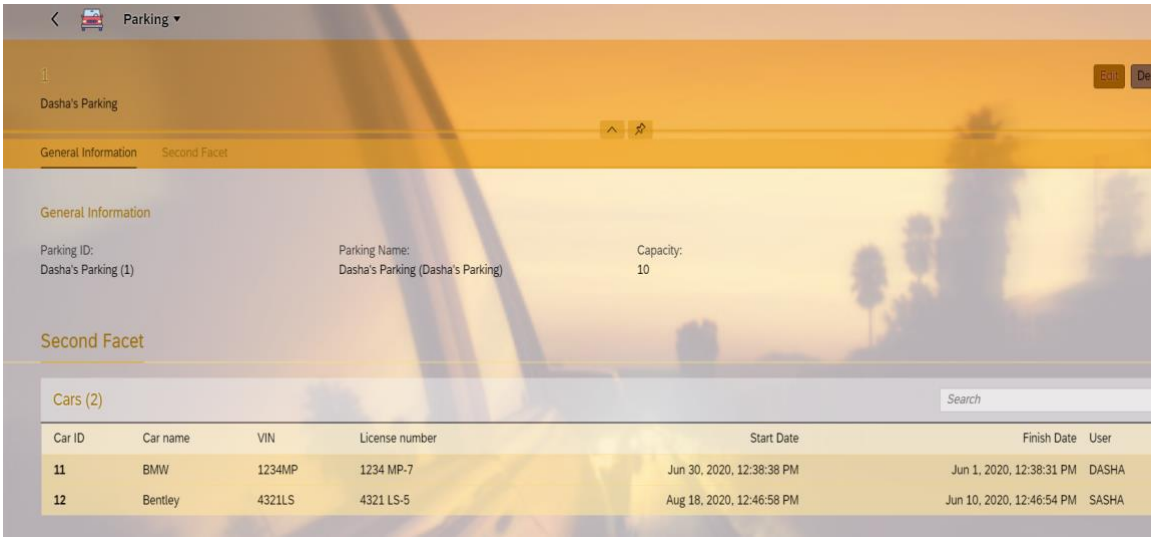



Рисунок 3.5.4– Информация о конкретной автостоянке

При нажатии на машину, мы можем увидеть подробную информацию о ней (рисунок 3.5.5)

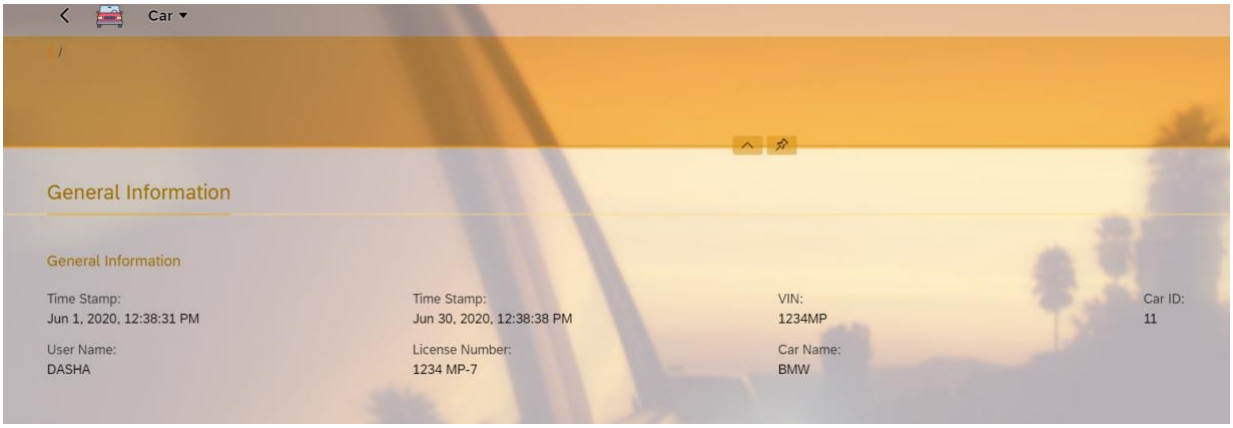



Рисунок 3.5.5 – Отображение информации о конкретной машине

Страница графического отображения информации об автостоянке с помощью *CDS View* и *OData Service*. На этой странице приложения отображаются графики заполненности автостоянок машинами, круговые и столбчатые диаграммы, отсортированные по количеству машин в автостоянке (рисунок 3.5.6)

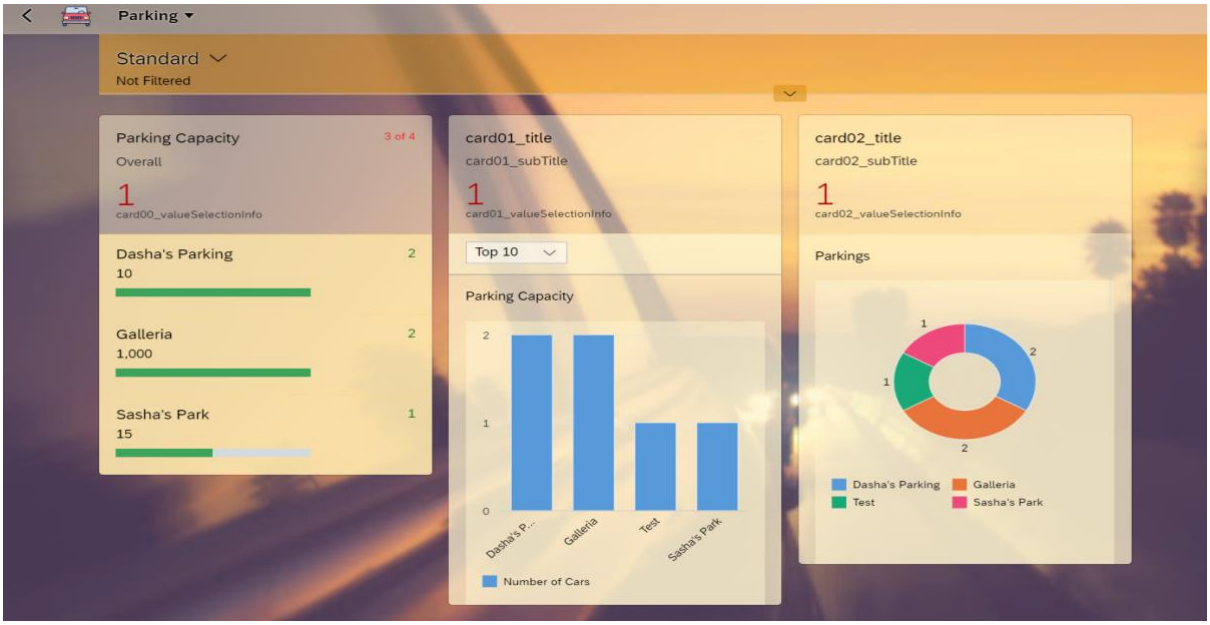



Рисунок 3.5.6 – Страница графического отображения

При нажатии на любой из столбиков диаграммы, пользователь переходит на запись выбранной автостоянки. (рисунок 3.5.7)


Рисунок 3.5.7 – Отображение выбранной автостоянки

Также есть часть, которая демонстрирует работу с автостоянками для машин, и показывает загруженность автостоянки машинами.

При переходе с вкладки *HOME* (рисунок 3.5.1) на вкладку *CARS*, мы можем работать с машинами. На странице находится поле для ввода *Id* машины. (рисунок 3.5.8).

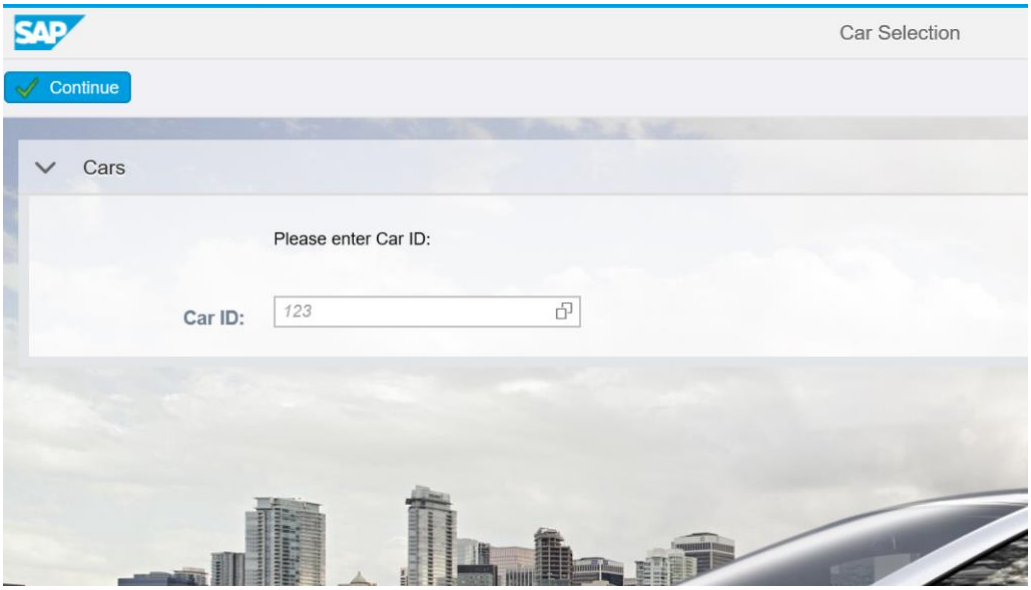



Рисунок 3.5.8 – Страница выбора машины

Если пользователь осуществил ввод *Id* машины, который не находится в базе, то покажется сообщение (рисунок 3.5.9)

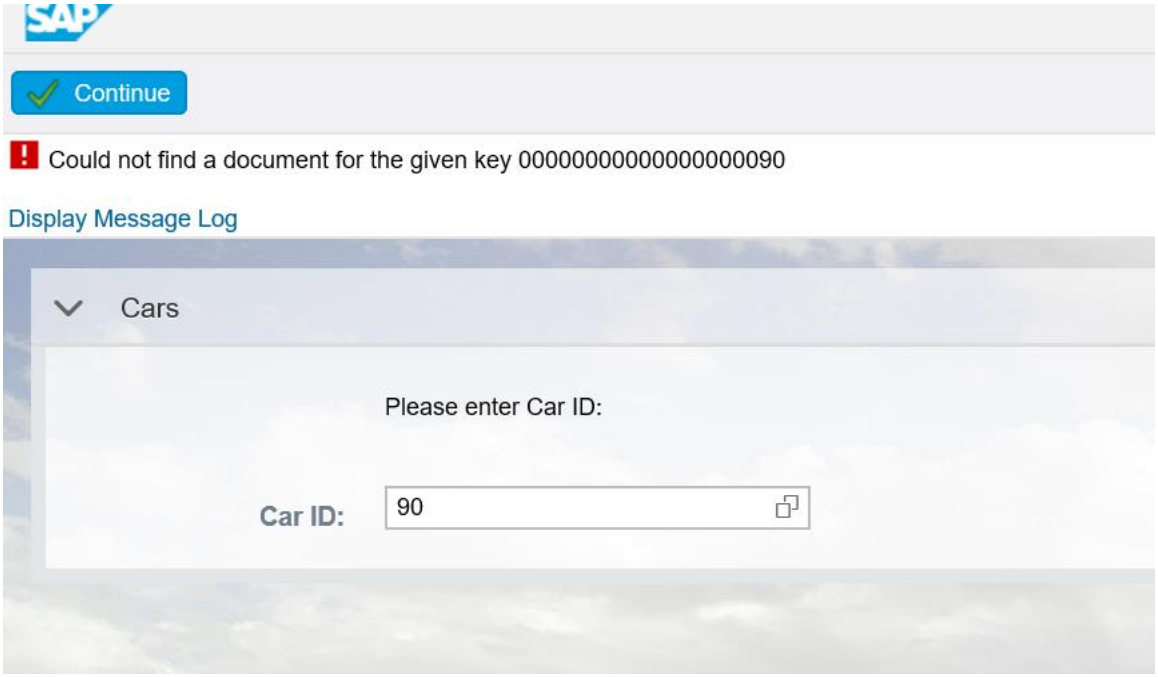



Рисунок 3.5.9– Ошибка ввода пользователя

При вводе корректного *Id*, осуществляется переход пользователя на следующую страницу приложения, где отображаются три вкладки: главная информация, график, дополнительная информация.

Главная информация содержит в себе ключевые поля таблицы базы данных, которые несут в себе смысловую информацию, а также картинку машины. Данная картинка не содержится в БД, так как сохранение изображения в БД не предусмотрено базой данных *SAP HANA*. Эта функция больше служит для простого и наглядного отображения машины пользователя. (рисунок 3.5.10).

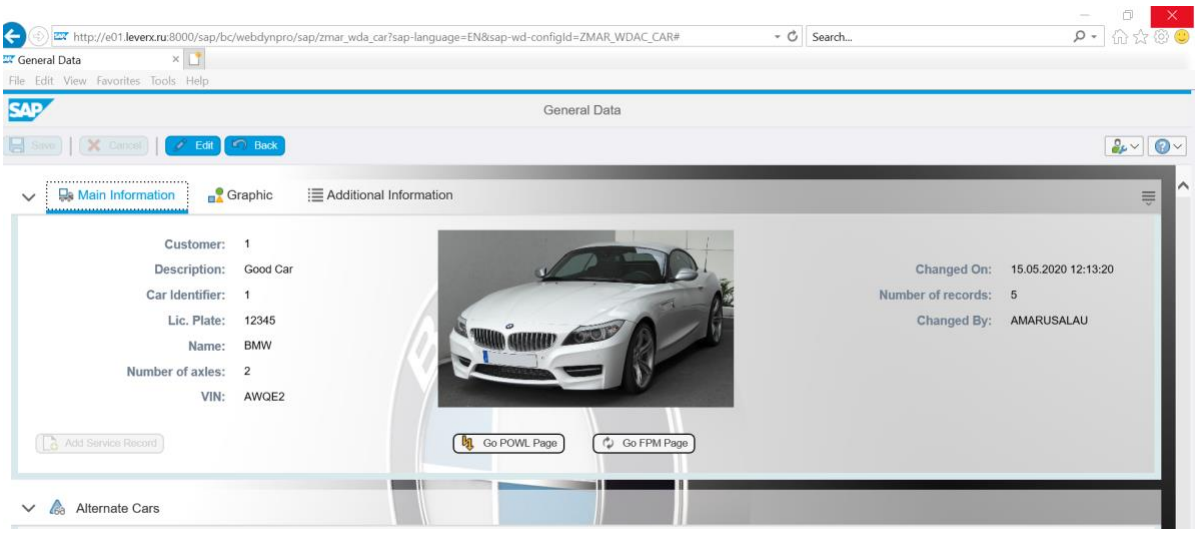



Рисунок 3.5.10– Главная информация о машине

Также на главное странице отображается информация и о остальных машинах, которые присутствуют в БД. (рисунок 3.5.11)

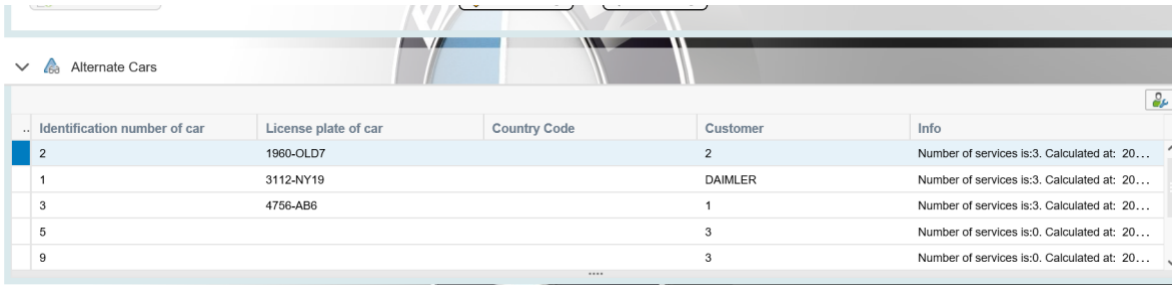



Рисунок 3.5.11– Остальные машины БД

Для отображения статусов машин в БД предусмотрен график, который показывает, какие машины в процессе стоянки, а какие уже выехали (рисунок 3.5.12)

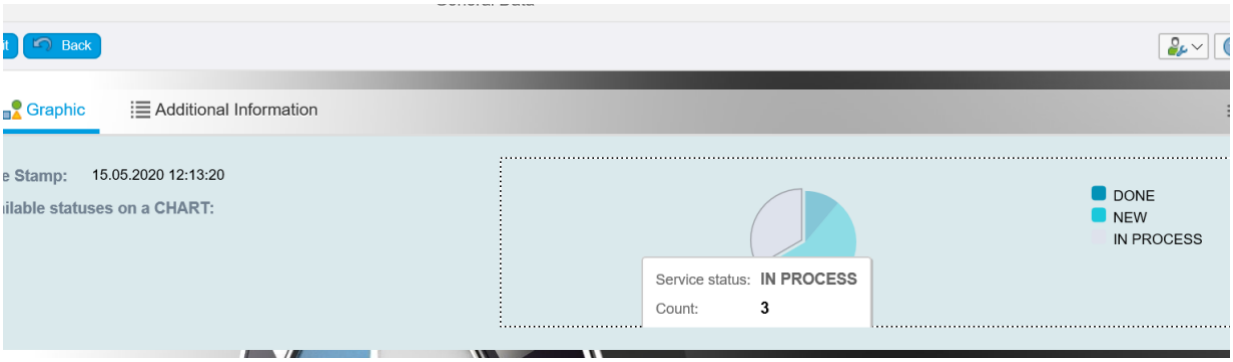



Рисунок 3.5.12– Статусы машин. График

И третья вкладка как раз таки и позволяет управлять статусами машин, с помощью нажатия кнопок: *Set in Process*, *Set as New*, *Set as Done*. (рисунок 3.5.13).

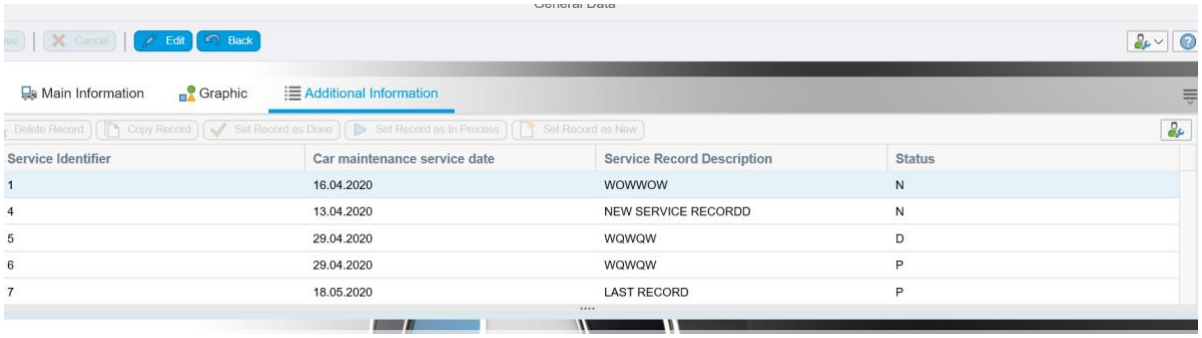



Рисунок 3.5.13– Вкладка для изменения статусов машин

Также, при нажатии на поля, отображается список доступных объектов в таблице базы данных. Это позволяет пользователю лучше ориентироваться и не вводить недоступные ключи при поиске нужного объекта. (рисунок 3.5.14)

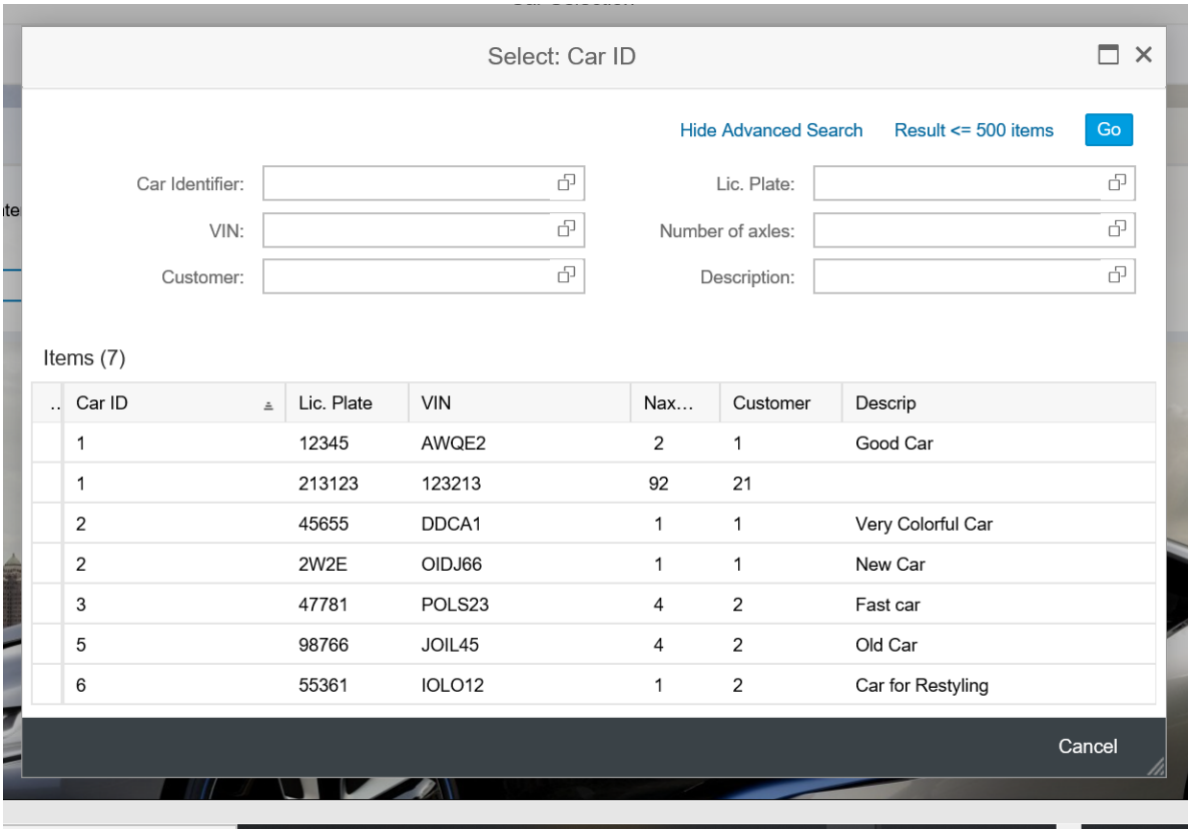



Рисунок 3.5.14– Отображение объектов базы данных в таблице

Таким образом, работа с сайтом достаточно удобная и интуитивно понятная даже неопытным пользователям.

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы был создан ресурс, с помощью которого можно создать автостоянку с участием количества машин, просмотреть статистику об автостоянках, просмотреть, заполнить и изменить информацию о машинах. Данный проект представляет собой пользовательский интерфейс, предназначенный для просмотра и изменения данных из базы. Бизнес-логика создана в соответствии с требованиями, представленными выше.

Соблюдены правила создания базы данных, в следствие чего предупреждены возможности неожиданного исчезновения или редактирования данных системой.

Так же соблюдались предписания по созданию пользовательского интерфейса, интерфейс получился интуитивно понятным и лояльным.

Использовались современные технологии по созданию веб-ресурсов, что обеспечило широкое отображение ресурсов.

Пояснительная записка оформлена в соответствии с требованиями стандарта СТП 01-2017.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] *ABAP CDS* [Электронный ресурс] / *ABAP CDS Development User Guide*– Электрон. дан. – Режим доступа: *https*://*help*.*sap*.*com*/*viewer*/*f*2*e*545608079437*ab*165*c*105649*b*89*db*/7.51.3/*enUS*/, свободный– Загл. С экрана – Яз. англ.

[2] *OData Service* [Электронный ресурс] / *Generating and Creating OData Services* – Электрон. дан. – Режим доступа: *https*://*help*.*sap*.*com*/*viewer*/*cc*0*c*305*d*2*fab*47*bd*808*adcad*3*ca*7*ee*9*d*/7.51.6/*en*-/, свободный – Загл. С экрана – Яз. рус.

[3] *Sap Cloud* [Электронный ресурс] / *SAP Cloud Platform Connector*– Электрон. дан. – Режим доступа: *https*://*tools*.*hana*.*ondemand*.*com*/#*cloud*, свободный – Загл. С экрана – Яз. рус., англ.

[4] *SAP HANA* [Электронный ресурс] / *SAP HANA Reference Scenario*– Электрон. дан. – Режим доступа: *https*://*blogs*.*sap*.*com*/2013/10/21/*abap-for-sap-hana-reference-scenario-video-tutorials*– Загл. С экрана – Яз. англ.

[5] *ABAP Core* [Электронный ресурс] / *ABAP Core Data Services*– Электрон. дан. – Режим доступа: *https*://*blogs*.*sap*.*com*/2017/09/09/*abap-core-data-services-introduction-abap-cds-view*/– Загл. С экрана – Яз. англ.