Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Технологии разработки программного обеспечения

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к проекту
на тему

КУРЬЕРСКАЯ СЛУЖБА

БГУИР КП 1-40 04 01 009 ПЗ

Студент: гр. 953501
Дубровский К. Л.

Руководитель: ассистент кафедры информатики Гриценко Н. Ю.

Минск 2022

СОДЕРЖАНИЕ

1 ПРЕДНАЗНАЧЕНИЯ И ФОРМУЛИРОВКА ЦЕЛЕЙ

1.1 Целевая аудитория

Целевой аудиторией являются:

– пользователи, которым необходима быстрая доставка интересующего их предмета из точки А в точку Б (до четырех последовательных адресов);

– пользователи, планирующие переезд в новую квартиру/дом в пределах страны, планирующие ремонт, и им необходима доставка стройматериалов;

– пользователи, которые приобрели крупный товар в интернет-магазине, и им нужна доставка товара из пункта выдачи.

1.2 Цели документа

Целями данного документа являются:

– установка рабочего плана;

– распределение ролей и задач в команде;

– постановка задач.

1.3 Риски

Создание данного продукта может быть связано со следующими рисками:

– риск произвести неправильную оценку сложности проекта и не уложиться в поставленные сроки;

– риск наличия неквалифицированных сотрудников;

– риск не учесть некоторые аспекты предметной области;

– риск создать не конкурентоспособный продукт.

1.4 Анализ существующих аналогов

В качестве аналогичных программных продуктов рассматривались другие курьерские службы доставки, такие как:

– www.dostavka24.by (см. рис. 1.1);

– www.cdek.by;

– www.deliver.by (см. рис. 1.2).

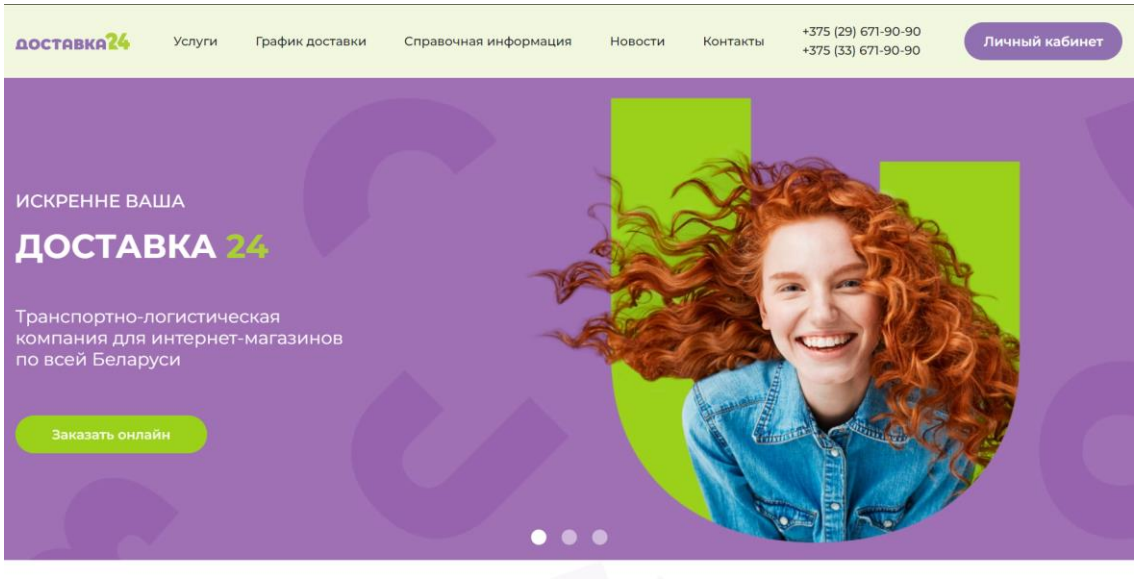

Рисунок 1.1 – Сайт курьерской службы dostavka24.by

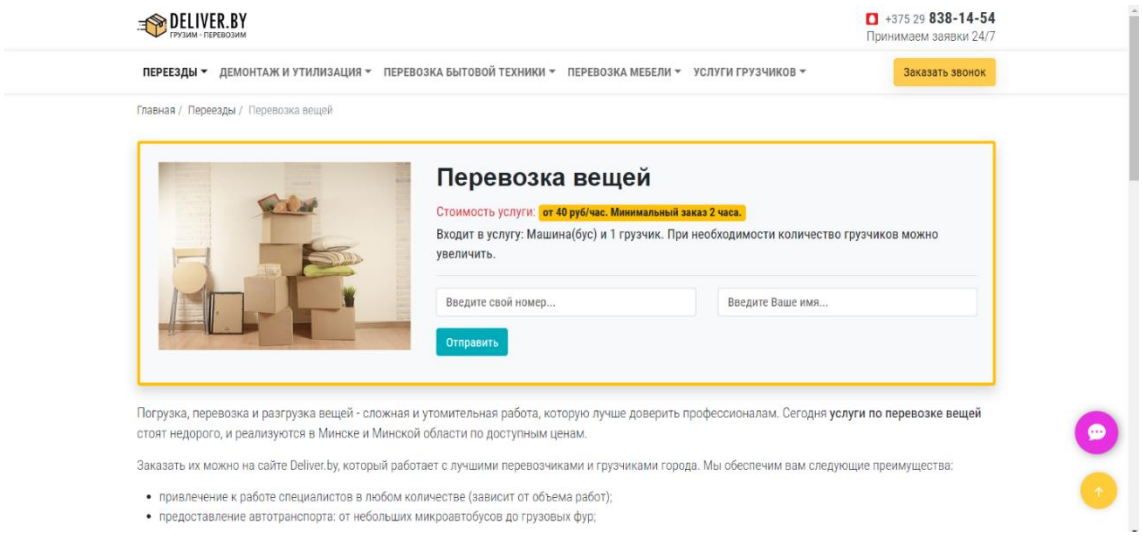

Рисунок 1.2 – Сайт курьерской службы deliver.by

В ходе изучения и анализа конкурентов были выделены следующие преимущества:

1 Предлагаемый нами продукт представляет из себя систему приложений, которая позволяет решить некоторые проблемы:

а) Строгое разделение ролей (с помощью разделения функционала по приложениям);

б) В каждом приложении присутствует только функционал, отведённый конкретной роли таким образом, что, например, роль “Водитель” даже теоретически (при знании данных учётной записи роли “Администратор”) не получит доступ к функционалу роли “Администратор”.

2 Клиент не предоставляет никаких персональных данных, кроме мобильного телефона.

3 При составлении заказа клиент пользуется удобной интернет-картой.

4 Максимальное упрощение оформления заказа клиентом путём избавления от процедуры регистрации и аутентификации (на сайтах конкурентов из-за процедуры регистрации у клиента уйдёт больше времени на формирование заказа).

2 ТРЕБОВАНИЯ К ПРОГРАММНОМУ СРЕДСТВУ

2.1 Общие требования

Сервис включает в себя:

А. Сайт для формирования заказа клиентом;

Б. Приложение, реализующее необходимый функционал для работы администратора;

В. Приложение, реализующее необходимый функционал для работы водителя.

Общие требования (далее ОТ) для сайта:

– ОТ А-1: Стартовая страница сайта должна содержать следующую информацию:

а) информацию об услугах курьерской службы доставки;

б) информацию о стоимости будущего заказа;

в) информацию о возможности связи с администратором.

– ОТ А-2: Пользователь должен иметь возможность сформировать заказ на доставку при помощи формы. Форма регистрации заказа пользователя-клиента должна содержать следующие поля:

а) имя клиента;

б) номер телефона клиента;

в) пункты А и Б доставки груза;

г) ожидаемую дату и время заказа;

д) информацию о грузе.

Поля для заполнения информации о грузе могут быть дополнительно добавлены в форму, если клиент планирует перевести груз, состоящий из нескольких позиций.

– ОТ А-3: После формирования заказа пользователь должен получить информацию о стоимости заказа до его регистрации в системе заказов.

Общие требования для приложения администратора:

– ОТ В-1: Администратор должен иметь возможность просматривать, редактировать, удалять записи о заказах, водителях, грузчиках, тарифах и автомобилях в таблицах.

– ОТ В-2: Администратор должен иметь возможность добавить новые записи о водителях, грузчиках, автомобилях заполнив соответствующие формы.

– ОТ В-3: Администратор должен иметь возможность просматривать записи в таблицах архива, но не должен иметь возможности их изменять.

Общие требования для приложения водителя:

– ОТ С-1: Водитель должен иметь установленным собственное приложение для водителей, вход в которое должен осуществляться по аутентификационным данным аккаунта, предоставленного ему администратором.

– ОТ С-2: Водитель должен иметь возможность просматривать записи о заказах, которые свободны и могут быть выбраны водителем.

– ОТ С-3: Водитель должен иметь возможность выбрать заказ из свободных и закрепить его за собой, выбрав также грузчика(-ков) и автомобиль от компании.

– ОТ С-4: Водитель должен иметь возможность просмотреть указанный в заказе маршрут по интернет-карте.

2.2 Функциональная карта

Функциональная карта ПС представлена на рисунке 2.1.

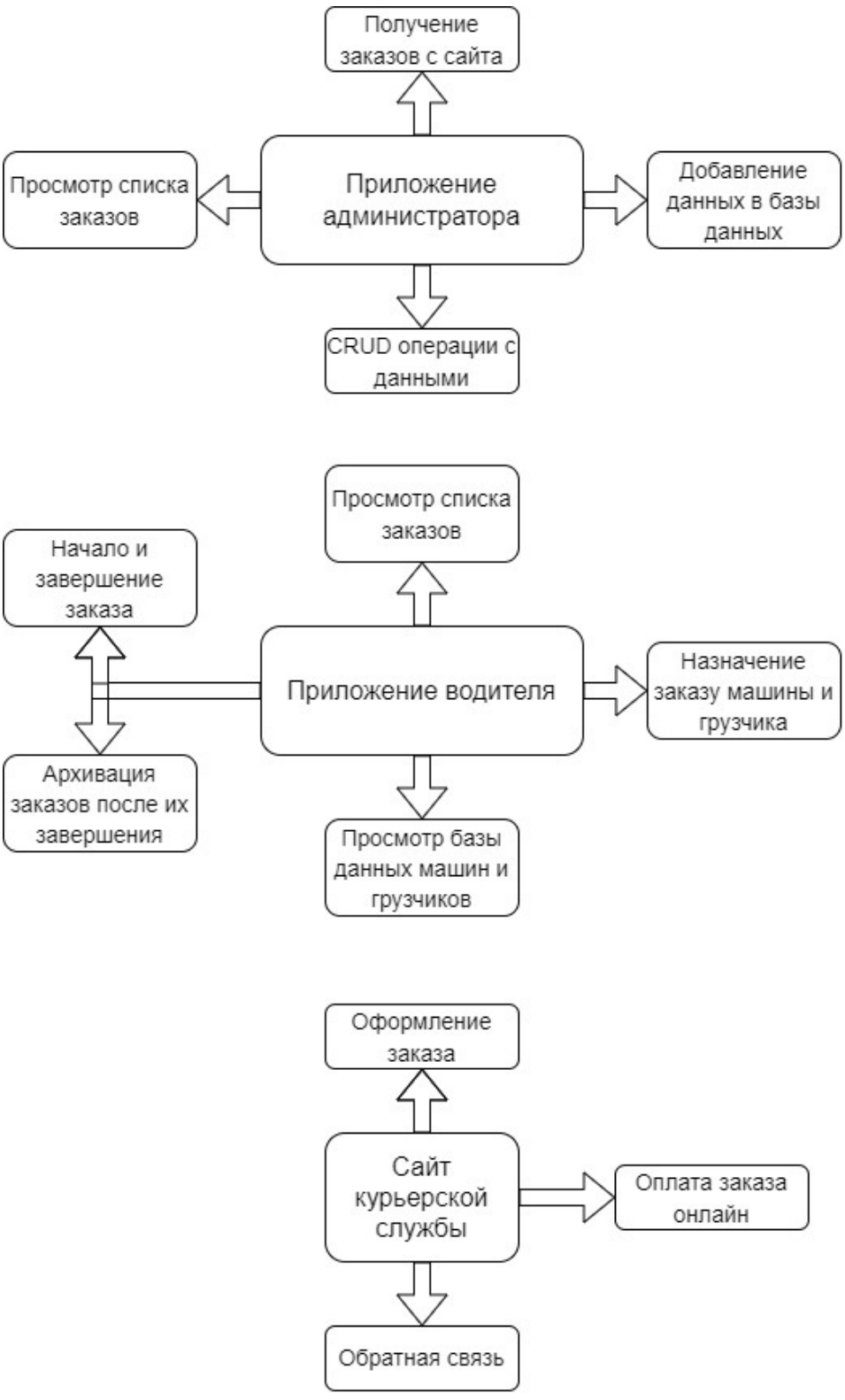

Рисунок 2.1 – Функциональная карта ПС

2.3 Варианты использования программного средства

Диаграмма вариантов использования (англ. use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. Основное назначение диаграммы – описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Диаграмма маршрута путешествия (англ. user flow diagram) – это визуальное представление последовательности действий, которые пользователь выполняет для достижения своей цели.

Диаграммы вариантов использования и маршрута путешествия программного средства соответственно представлены на рисунках 2.2 и 2.3.

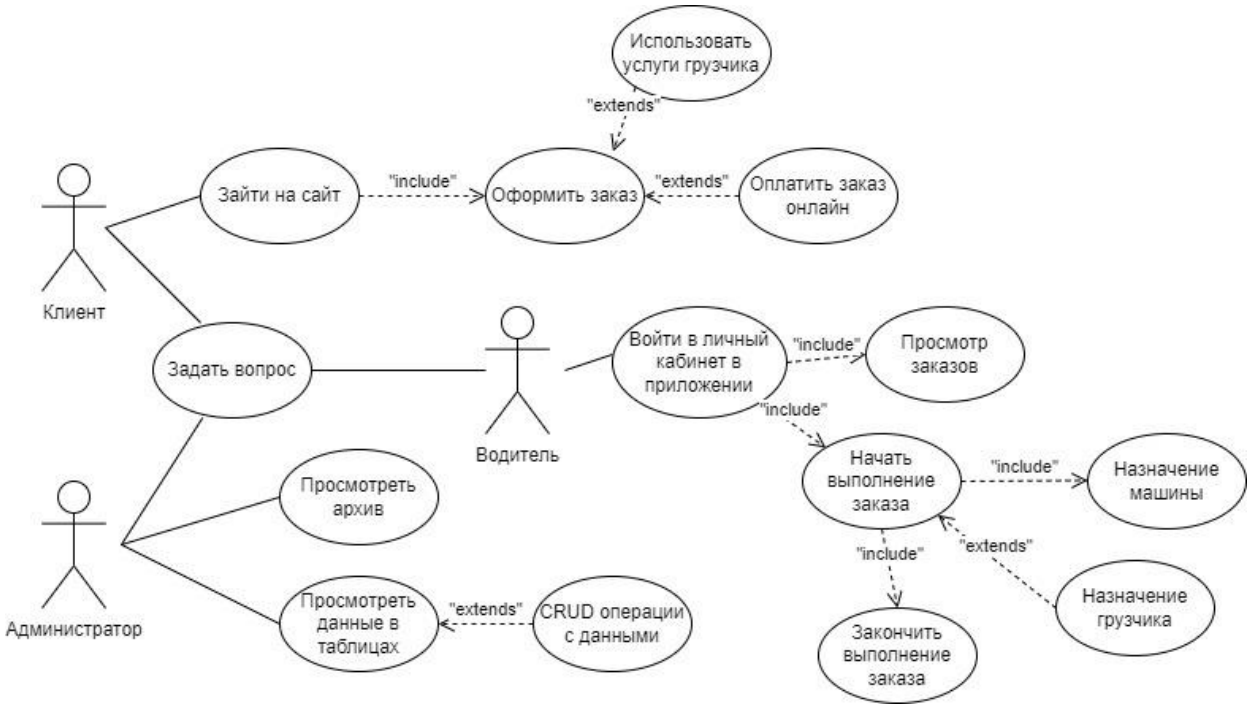

Рисунок 2.2 – Диаграмма вариантов использования ПС (use-case)

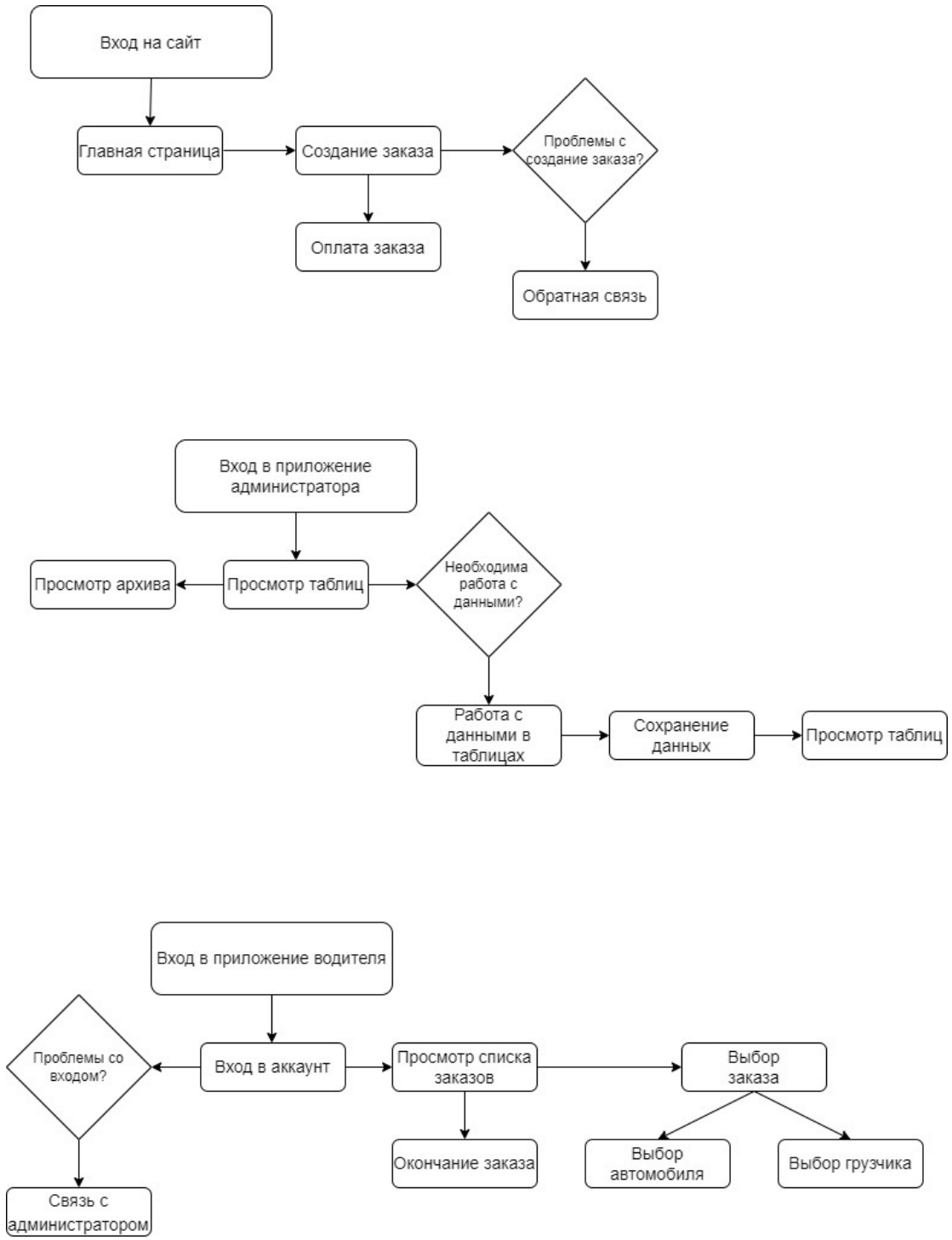

Рисунок 2.3 – Диаграмма вариантов использования ПС (User flow)

2.4 База данных

Для работы сервиса необходимо спроектировать базу данных.

База данных должна храниться на интернет-сервере. Требуется, чтобы к ней имели доступ сайт (добавление записей о новом заказе), приложение администратора (просмотр, добавление, удаление и редактирование записей), приложение водителя (просмотр записей о заказе, изменение статуса заказа на «выполняется» и «выполнен»).

Модель базы данных представлена на рисунке 2.4.

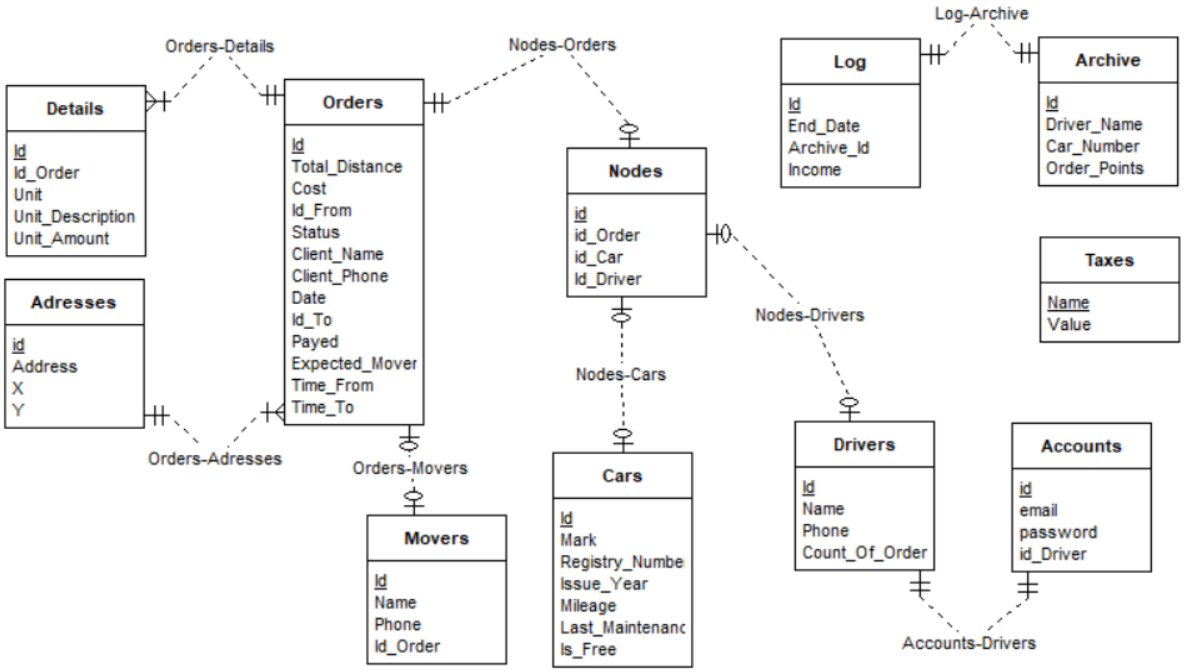

Рисунок 2.4 – ER диаграмма базы данных

База данных состоит из 11 таблиц. Таблицы Drivers и Accounts должны хранить информацию о водителях. Таблицы Movers и Cars должны хранить данные о грузчиках и автомобилях кампании соответственно. За добавление и удаление записей в этих таблицах отвечает администратор.

Таблицы Orders, Details Addresses должны содержать записи о заказе, причём одной записи в таблице Orders допускается соответствие нескольких записей из Details. Также адресам (записям) из таблицы Addresses должны соответствовать пары записей из Orders, хранящие информацию о пункте старта и завершения маршрута доставки. В этих таблицах записи должны добавляться после ввода клиентом данных на сайте. Записи в таблице Nodes формируются, когда водитель принимает заказ, выбрав дополнительно автомобиль и грузчиков.

После того, как заказ выполнен, данные о нем необходимо удалить из предыдущих таблиц и занести в таблицы Archive и Log, которые представляют собой архив записей заказов.

В таблице Taxes необходимо хранить некоторые значения, которые могут быть полезны при добавлении новых заказов и могут изменятся (например, зависимость стоимости доставки от расстояния, максимальное расстояние, минимальная стоимость и т.д.).

База данных должна находиться в третьей нормальной форме, то есть необходимо выполнение следующих пунктов:

– все атрибуты должны быть простые (1-я НФ),

– каждый не ключевой атрибут должен неприводимо завесить от первичного ключа, т.е. в составе потенциального ключа должно отсутствовать меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость (2-я НФ),

– каждый не ключевой атрибут должен нетранзитивно зависеть от первичного ключа, т.е. все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы, должны быть вынесены в отдельные таблицы (3-я НФ).

2.5 Диаграмма потока данных

Диаграмма потока данных (англ. Data Flow Diagram), отображает потоки данных между системами, базами данных. Ключевыми элементами являются входные/выходные данные, системы, точки хранения и сбора данных. Диаграмма потока данных программного средства представлена на рисунке 2.5.

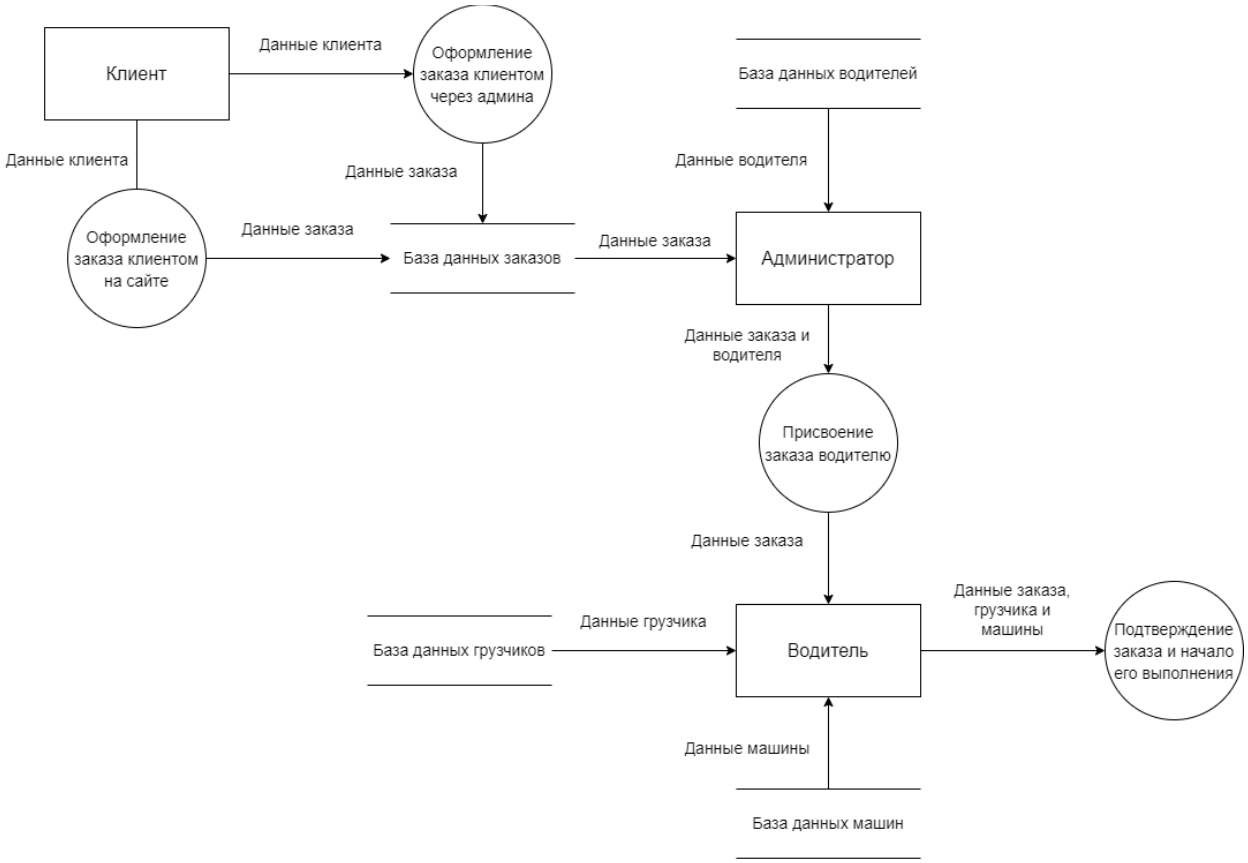

Рисунок 2.5 – Диаграмма потока данных (Data flow)

2.6 Интерфейс

Программные средства должны использовать приятный пользователю и интуитивно понятный графический интерфейс. Интерфейс должен реагировать на любое действие пользователя. Схемы интерфейса для интернет-сайта, приложения администратора и приложения водителя представлены на рисунках 2.6, 2.7 и 2.8 соответственно.

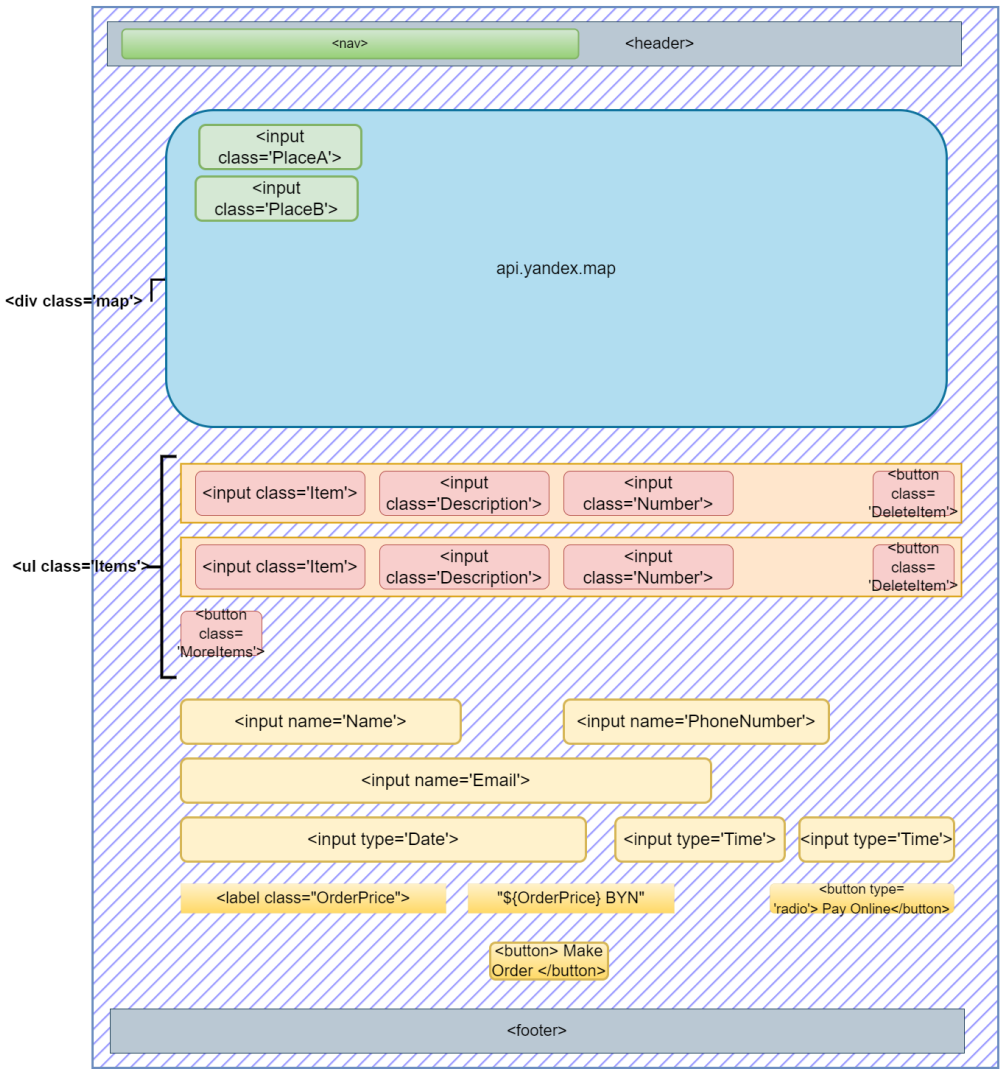

Рисунок 2.6 – Интерфейс сайта заказа

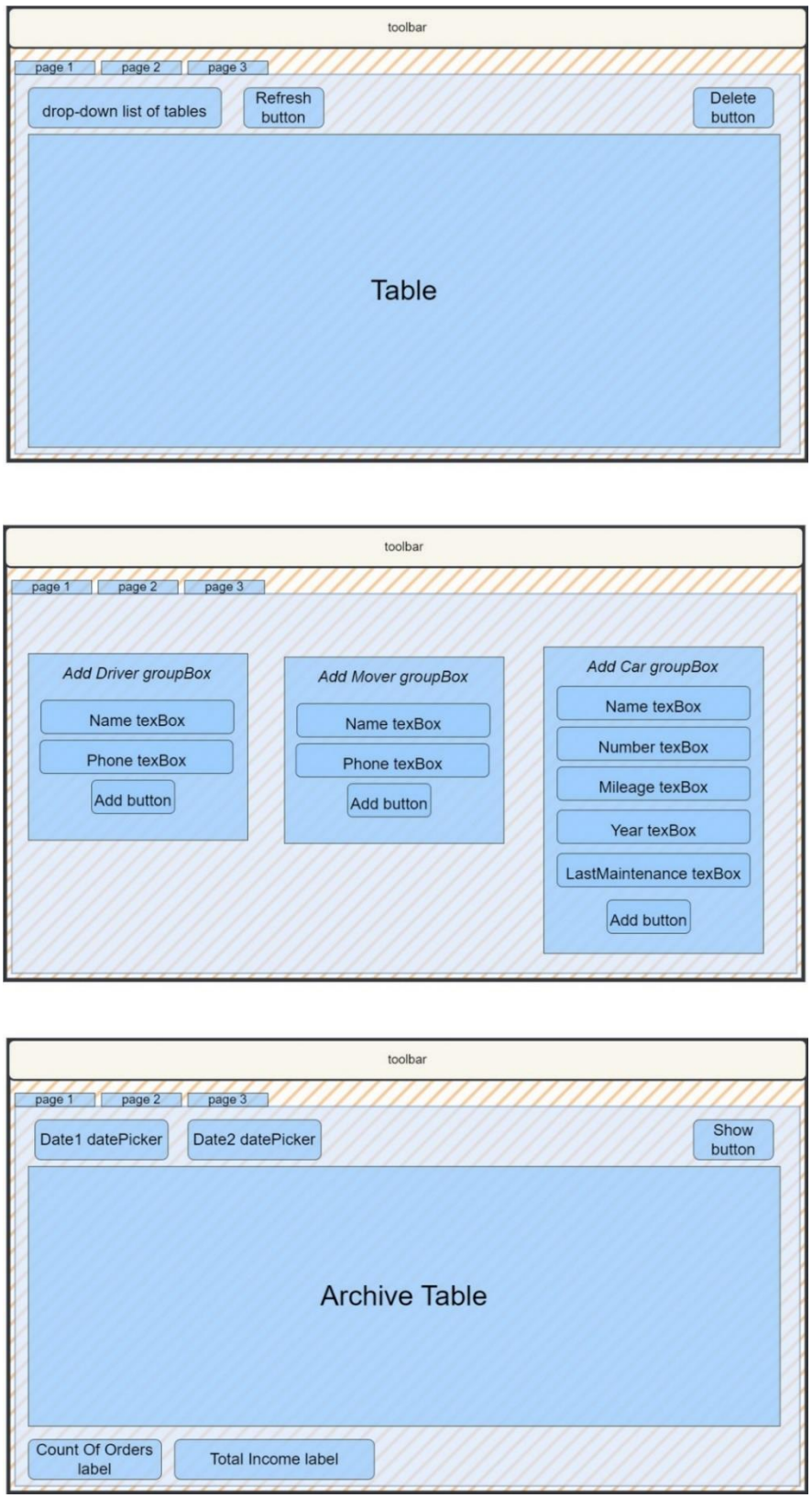

Рисунок 2.7 – Интерфейс приложения администратора (3 страницы)

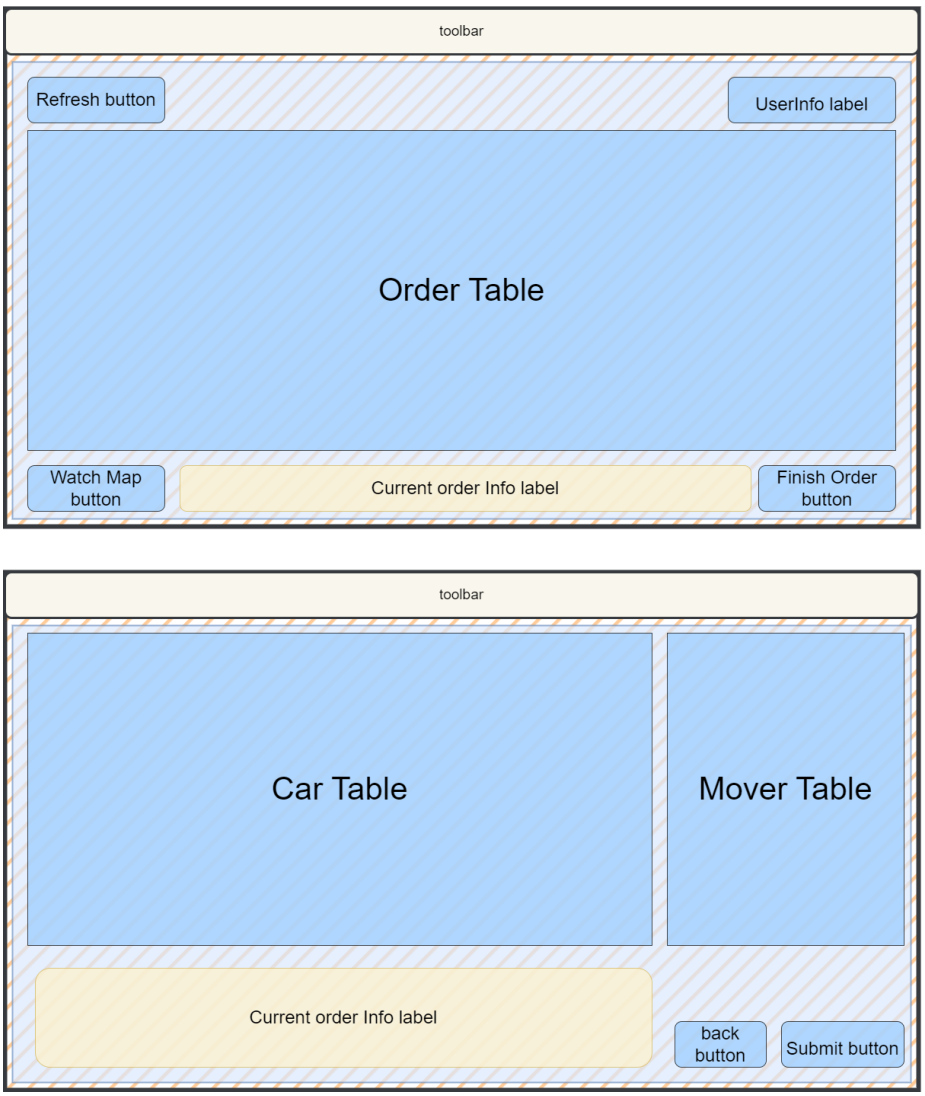

Рисунок 2.8 – Интерфейс приложения водителя (2 страницы)

2.7 Системные характеристики

Приложение должно обладать следующими системными характеристиками (СХ):

– СХ-1: Сервис должен использовать базу данных;

– СХ-2: При написании приложений должны быть использованы технологии .NET Core, .NET Framework, ASP.NET и SQL Server;

– СХ-3: Рекомендуемой системой для работы сервиса является ОС Windows.

2.8 Атрибуты качества

Атрибутами качества (АК) приложения являются следующие пункты:

– АК-1: Приложение должно обеспечивать скорость обработки данных 5 МБ/сек со следующим (или эквивалентным оборудованием): CPU i7, RAM 4 GB, средняя скорость чтения/записи диска 30 МБ/сек;

– АК-2: Приложение должно обрабатывать вводимые данные на русском и английском языках;

– АК-3: Приложение должно корректно вести себя при неправильно введенных входных данных.

2.9 Детальные спецификации

При разработке должны быть учтены следующие детальные спецификации (ДС):

– ДС-1: Приложение должно завершать свою работу только корректным образом;

– ДС-2: Приложение должно корректно отрабатывать при возникновении ошибок и исключительных ситуаций.

3 ВЫБОР ИНСТРУМЕНТОВ РАЗРАБОТКИ

На основании выдвинутых к программному средству основных функциональных и нефункциональных требований, обзора существующих аналогов, было принято решение о необходимости проектировании сети приложений, использующих одну базу данных в сети Интернет, разделенных функциональными требованиями на три части: веб-сайт для клиента, приложение для администратора и приложение для водителя.

3.1 Программная технология .NET

Microsoft.NET - программная технология, предназначенная для создания как обычных программ, так и веб-приложений (в качестве платформы для разработок впервые предложена корпорацией Microsoft).

Одной из основных идей Microsoft.NET является совместимость различных служб, написанных на разных языках. Например, служба, написанная на C++ для Microsoft.NET, может обратиться к методу класса из библиотеки, написанной на Delphi; на C# можно написать класс, наследованный от класса, написанного на Visual Basic.NET, а исключение, созданное методом, написанным на C#, может быть перехвачено и обработано в Delphi. Каждая библиотека (сборка) в.NET имеет сведения о своей версии, что позволяет устранить возможные конфликты между разными версиями сборок.

.NET является патентованной технологией корпорации Microsoft. Тем не менее, после заключения договоренности с компанией Novell, была признана технология Mono как реализация.NET на Unix-подобных системах (GNU/Linux, Mac OS X). Однако договорённость касается Novell и клиентов Novell, также технологии ASP.NET, ADO.NET и Windows.Forms не были стандартизированы ECMA/ISO и использование их в Mono находится под угрозой претензий со стороны Microsoft. Mono предоставляет реализацию ASP.NET, ADO.NET и Windows.Forms, но в то же время рекомендует обходить эти API.

Средами разработки.NET-приложений являются:

– Microsoft Visual Studio (C#, Visual Basic.NET, Managed C++);
– SharpDevelop;
– MonoDevelop;
– Eclipse;
– Borland Developer Studio (Delphi for.NET, C#);
– PascalABC.NET и т. д.

Приложения также можно разрабатывать в текстовом редакторе и использовать консольный компилятор.

Так же, как и технология Java, среда разработки.NET создаёт байт-код, предназначенный для исполнения виртуальной машиной. Входной язык этой машины в.NET называется MSIL (Microsoft Intermediate Language), или CIL (Common Intermediate Language, более поздний вариант), или просто IL. Применение байт-кода позволяет получить кроссплатформенность на уровне

скомпилированного проекта (в терминах.NET: сборка), а не только на уровне исходного текста, как, например, в C. Перед запуском сборки в среде исполнения CLR байт-код преобразуется встроенным в среду JIT-компилятором (just in time, компиляция на лету) в машинные коды целевого процессора. Также существует возможность скомпилировать сборку в родной (native) код для выбранной платформы с помощью поставляемой вместе с .NET Framework утилиты NGen.exe.

Технология .Net обладает следующими преимуществами:

1 Полные возможности взаимодействия с существующим кодом. Существующие двоичные компоненты COM отлично работают вместе с двоичными файлами .NET.

2 Полное и абсолютное межъязыковое взаимодействие. В отличие от классического COM, в.NET поддерживаются межъязыковое наследование, межъязыковая обработка исключений и межъязыковая отладка.

3 Общая среда выполнения для любых приложений .NET, вне зависимости от того, на каких языках они были созданы. Один из важных моментов при этом - то, что для всех языков используется один и тот же набор встроенных типов данных.

4 Библиотека базовых классов, которая обеспечивает сокрытие всех сложностей, связанных с непосредственным использованием вызовов API, и предлагает целостную объектную модель для всех языков программирования, поддерживающих.NET.

5 Отсутствие сложности, присущей COM. IClassFactory, IUnknown, код IDL и VARIANT-совместимые типы данных (BSTR, SAFEARRAY и остальные) не используются в коде программ.NET.

6 Действительное упрощение процесса развертывания приложения. В .NET нет необходимости регистрировать двойные типы в системном реестре. Более того, .NET позволяет разным версиям одного и того же модуля DLL мирно сосуществовать на одном компьютере.

Технологии CLR, CTS и CLS очень важны для понимания смысла платформы.NET. С точки зрения программиста.NET вполне можно рассматривать просто как новую среду выполнения и новую библиотеку базовых классов. Среда выполнения.NET как раз и обеспечивается с помощью Common Language Runtime (CLR, стандартная среда выполнения для языков). Главная роль CLR заключается в том, чтобы обнаруживать и загружать типы.NET и производить управление ими в соответствии с вашими командами. CLR берет на себя всю низкоуровневую работу - например, автоматическое управление памятью, межъязыковым взаимодействием, развертывание (с отслеживанием версий) различных двоичных библиотек.

Еще один строительный блок платформы.NET - это Common Type System (CTS, стандартная система типов). CTS полностью описывает все типы данных, поддерживаемые средой выполнения, определяет, как одни типы данных могут взаимодействовать с другими и как они будут представлены в формате метаданных.NET.

Важно понимать, что не во всех языках программирования.NET обязательно должны поддерживаться все типы данных, которые определены в CTS. Common Language Specification (CLS) - это набор правил, определяющих подмножество общих типов данных, в отношении которых гарантируется, что они безопасны при использовании во всех языках.NET.

Помимо спецификаций CLR и CTS/CLS платформа.NET предоставляет в распоряжение программиста также и библиотеку базовых классов, доступную из любого языка программирования.NET. Библиотека базовых классов не только прячет обычные низкоуровневые операции, такие как файловый ввод-вывод, обработка графики и взаимодействие с оборудованием компьютера, но и обеспечивает поддержку большого количества служб, используемых в современных приложениях.

3.2 Технология разработки ASP.NET

ASP.NET – это новая технология для создания мощных сценариев, которые выполняются на сервере. Она предоставляет разработчикам службы, необходимые для создания .NET-приложений. Компания Microsoft первоначально планировала назвать свой продукт ASP+ -- как усовершенствование ASP. Но после создания платформы .NET ASP+ была переименована в ASP.NET и вошла в состав пакета среды разработки приложений Visual Studio.NET.

Непосредственно взаимодействуя с операционной системой, среда .NET Framework предоставляет интерфейс ASP-приложениям. Новая технология ASP.NET позволяет создавать приложения на нескольких языках программирования, например, на VisualBasic .NET, C# и JScript .NET. Благодаря этому приложениям предоставляются возможности .NET, такие как работа в среде CLR, безопасность типов и наследование. Наиболее важными усовершенствованиями, добавленными в ASP.NET, являются серверные элементы управления (ServerControls), новые возможности работы в Web, кэширование Web-страниц и новая объектная модель.

Технология ASP.NET является новой средой разработки Web-приложений. Технология ASP базировалась на использовании языков сценариев. В основу ASP.NET положена работа в среде CLR, что позволяет создавать Web-приложения на любом языке, поддерживаемом платформой .NET. Независимо от языка программирования, использованного при создании приложения ASP, его код компилируется в код на промежуточном языке IL. Это немаловажное преимущество, так как теперь возможности одного языка могут использоваться в другом языке без необходимости написания дополнительного кода. Таким образом достигается высокая степень повторного использования кода.

Файлы страниц, создаваемых в рамках технологии ASP.NET, могут иметь различные расширения. Файл стандартной ASP.NET-страницы имеет расширение .aspx. Файл Web-службы имеет расширение .asmx, а файл пользовательского элемента управления -- расширение .ascx. Поддержка различных форматов файлов позволяет одновременно использовать ASP.NET- и ASP-страницы. В зависимости от расширения файла сервер IIS вызывает соответствующий ISAPI-фильтр для управления выполнением задачи. Архитектура ASP.NET позволяет различать управляемый и неуправляемый код. На коде, управляемом средой CLR, написаны .NET-приложения, что позволяет использовать возможности .NET Framework. Например, с помощью функции отсоединенного доступа технология ASP.NET поддерживает работу с серверами IIS 4.0, ISS 5.0 или InternetExplorer 5.5.

ASP.NET поддерживает две модели программирования: Web-формы и Web-службы. Web-формы позволяют создавать Web-страницы с помощью форм. Элементы управления Web-форм доступны на панели инструментов General (Стандартная) и могут использоваться для создания элементов пользовательского интерфейса Web-форм. Элементы управления Web-форм могут

использоваться многократно, что значительно упрощает задачу написания кода для Web-страниц.

3.3 Средство управления базами данных MS SQL Server

В качестве средства управления базами данных (далее СУБД) для хранения данных системы была выбрана СУБД MS SQL Server.

SQL Server – это комплексная платформа управления данными и бизнес-аналитики. Она обладает первоклассной масштабируемостью, возможностью создавать хранилища данных, продвинутыми средствами анализа и достаточной безопасностью, что позволяет использовать ее как основу для критически важных бизнес-приложений. Эта редакция позволяет консолидировать серверы и выполнять крупномасштабные OLTP-операции и создание отчетности.

Среди достоинств SQL Server можно выделить следующие:

1 Высокий уровень доступности. Технологии, которые защищают данные от дорогостоящих человеческих ошибок и максимально сокращают сроки аварийного восстановления, помогут обеспечить непрерывность ведения бизнеса.

2 Производительность и масштабируемость. Инфраструктура, поставившая официальный рекорд в обработке больших объемов данных и пиковых нагрузок.

3 Безопасность. Встроенные средства защиты от несанкционированного доступа позволят решить вопрос конфиденциальности и соответствия нормативным требованиям.

4 Управляемость. Автоматические диагностика, калибровка и настройка инфраструктуры позволяют снизить издержки на управление, сократить потребность в обслуживании и притом управлять огромными объемами данных.

5 Бизнес-аналитика. Большие объемы данных из хранилищ или киосков легко запрашиваются и анализируются, преобразуясь в практически значимый результат, ведущий к принятию верного решения.

СУБД SQL Server обладает большим рядом преимуществами:

1 Управление большими и постоянно растущими таблицами станет более эффективным благодаря прозрачному разбиению таблиц на управляемые блоки данных по технологии параллелизма в секционированных таблицах.

2 Улучшения в создании отказоустойчивых кластеров в ОС Windows Server.

3 Поврежденные страницы данных можно восстановить с зеркального сервера благодаря улучшенному зеркалированию баз данных.

4 Новые узлы в одноранговую репликацию можно добавлять во время работы, не отключая репликацию.

5 Регулятор ресурсов позволяет осуществлять упреждающий контроль приоритетности рабочей нагрузки и выделения ресурсов.

6 Сжатие резервных копий позволяет сократить время, требуемое на восстановление. Это также позволит уменьшить количество занимаемого резервными копиями пространства.

7 Средство сбора данных о производительности дает возможность осуществлять тонкую настройку экземпляров SQL Server по всему предприятию, а также устранять неполадки и производить мониторинг.

8 Общекорпоративные средства шифрования становятся возможными благодаря расширенному управлению ключами и аппаратным модулям безопасности.

3.4 Среда разработки MS Visual Studio

Среда разработки Microsoft Visual Studio – это набор инструментов и средств, предназначенных для помощи разработчикам программ любого уровня квалификации в решении сложных задач и создания новаторских решений. Разработчикам программного обеспечения часто приходится решать ряд проблем, чтобы создавать удачные программы. Роль Visual Studio заключается в том, чтобы улучшить процесс разработки и упростить разработку высокоэффективных программ.

В Visual Studio постоянно улучшается процесс разработки. Для этого проводится работа в следующих направлениях:

1 Производительность. Средства Visual Studio позволяют разработчикам работать с большей отдачей и затрачивать меньше усилий на повторяющиеся задачи. Следует отметить высокопроизводительные редакторы кода, поддержку технологии IntelliSense, мастеров и различных языков кодирования в одной интегрированной среде разработки (IDE), а также продукты управления жизненным циклом приложений (ALM) в Microsoft Visual Studio Team System. В новых версиях Visual Studio постоянно появляются новые средства, позволяющие разработчикам сосредоточиться на решении основных проблем, а не на рутинной работе.

2 Интеграция. Разработчики, применяющие Visual Studio, получают в свое распоряжение интегрированный продукт, включающий инструменты, серверы и службы. Продукты Visual Studio отлично работают вместе - не только один с другим, но и с прочими программами Майкрософт, включая серверные продукты и приложения Microsoft Office.

3 Комплексность. В Visual Studio содержатся инструменты для всех этапов разработки программного обеспечения (разработка, тестирование, развертывание, интеграция и управления) и для разработчиков любого уровня квалификации, от новичков до опытных специалистов. Visual Studio поддерживает разработку для различных типов устройств - ПК, серверов, сетевых и мобильных устройств.

4 Надежность. Visual Studio разрабатывается таким образом, чтобы обеспечить высокую надежность и совместимость. Visual Studio обладает удачным сочетанием безопасности, масштабируемости и взаимодействия. В Visual Studio всегда поддерживаются новейшие технологии, но везде, где это возможно, обеспечивается обратная совместимость.

3.5 Среда разработки MS SQL Server Management Studio

Так как имеющаяся база данных была в формате MS SQL Server, поэтому для развертывания использовался продукт Microsoft SQL Server Management Studio. Среда Microsoft SQL Server Management Studio (SSMS) — это интегрированная среда для доступа, настройки, администрирования, разработки всех компонентов SQL Server и управления ими. Служба SSMS сочетает в себе обширную группу графических инструментов с рядом отличных редакторов сценариев для обеспечения доступа к службе SQL Server для разработчиков и администраторов всех профессиональных уровней.

3.6 Набор сервисов Yandex Map API

API Яндекс Карт — это картографическая платформа, содержащая набор сервисов, которые позволяют использовать картографические данные и технологии Яндекса в проектах.

API Яндекс Карт делится на несколько составляющих:

– Адреса и организации;

– Карты;

– Сервисы для решения логистических задач;

– Геолокация.

3.6.1 Адреса и организации.

Геокодер позволяет узнать координаты по адресу пользователя, например, чтобы наиболее точно определить местоположение пользователя и сориентировать его по возможностям доставки в его район.

Поиск по организациям позволит найти все организации по заданному запросу.

3.6.2 Карты.

JavaScript API — позволяет встроить интерактивные Яндекс Карты себе на сайт, а также предоставляет возможность работы с базовыми картографическими сервисами Яндекса в браузере.

MapKit — позволяет встроить интерактивные Яндекс Карты в своё мобильное приложение на базе iOS или Android, а также иметь доступ к базовым картографическим сервисам Яндекса из мобильного приложения.

Tiles API — позволяет получить изображения подложки Яндекс Карт в ответ на http-запрос.

Static API — вернёт актуальное статическое изображение нужного фрагмента карты в ответ на http-запрос.

3.6.3 Сервисы для решения логистических задач.

Матрица расстояний — сервис для расчета попарных маршрутов между многими точками с учётом текущей ситуации на дорогах, а также прогноза на заданное время.

Получение деталей маршрута — сервис для получения геометрии и деталей маршрута по заданному набору точек.

3.6.4 Геолокация.

API Локатора — для определения местоположения датчиков и смартфонов по Wi-Fi, IP и вышкам сотовой связи — без использования GPS.

ПРИЛОЖЕНИЕ А
(обязательное)
Отчет к лабораторной работе №1

Разработка логической и физической модели БД на языке IDEF1.X

В ходе анализа предметной области были продуманы основные сущности для представления базы данных. Диаграммы логической и физической моделей представлены на рисунках А.1 и А.2 соответственно.

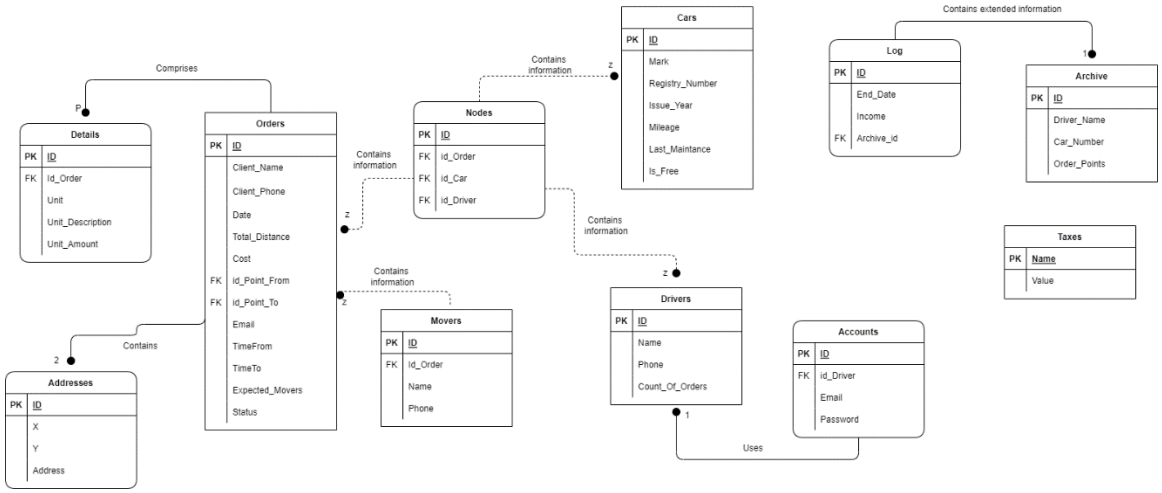

Рисунок А.1 – Диаграмма логический модели (IDEF1.X)

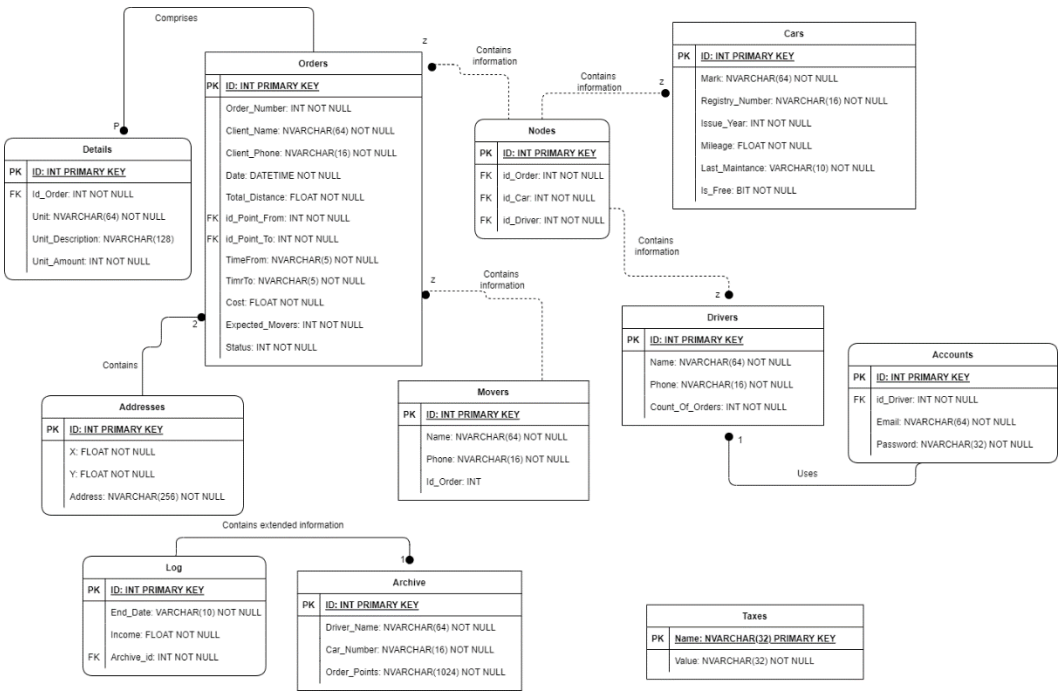

Рисунок А.2 – Диаграмма физический модели (IDEF1.X)

Написание DDL скриптов. Создание базы данных

```
USE CourierService;
go
CREATE TABLE Details (
	Id INT IDENTITY(1,1) PRIMARY KEY,
	Id_Order INT NOT NULL,
	Unit NVARCHAR(64) NOT NULL,
	Unit_Description NVARCHAR(128),
	Unit_Amount INT NOT NULL
);
go
CREATE TABLE Addresses (
	Id INT IDENTITY(1,1) PRIMARY KEY,
	X FLOAT Not null,
	Y FLOAT not null,
	[Address] NVARCHAR(64) NOT NULL,
);
go
CREATE TABLE Orders (
	Id INT IDENTITY(1,1) PRIMARY KEY,
	Total_Distance FLOAT NOT NULL,
	Cost FLOAT NOT NULL,
	Payed BIT NOT NULL,
	[Status] INT NOT NULL,
	Client_Name NVARCHAR(64) NOT NULL,
	Client_Phone NVARCHAR(17) NOT NULL,
	Client_Email NVARCHAR(64),
	Expected_Num_Of_Movers INT NOT NULL,
	Id_Point_From INT NOT NULL,
	Id_Point_To INT NOT NULL,
	Time_From VARCHAR(5) NOT NULL,
	Time_To VARCHAR(5) NOT NULL,
	[Date] VARCHAR(10) NOT NULL
);
go
CREATE TABLE Nodes (
	Id INT IDENTITY(1,1) PRIMARY KEY,
	Id_Order INT NOT NULL,
	Id_Car INT NOT NULL,
	Id_Driver INT NOT NULL,
);
go
CREATE TABLE Movers (
	Id INT IDENTITY(1,1) PRIMARY KEY,
	Id_Order Int,
	[Name] NVARCHAR(64) NOT NULL,
	Phone NVARCHAR(17) NOT NULL,
);
go
CREATE TABLE Cars (
	Id INT IDENTITY(1,1) PRIMARY KEY,
	Mark NVARCHAR(64) NOT NULL,
	Registry_Number NVARCHAR(16) NOT NULL,
	Issue_Year INT NOT NULL,
	Mileage FLOAT NOT NULL,
	Last_Maintenance VARCHAR(10) NOT NULL,
	Is_Free BIT NOT NULL,
);
go
```

```
	CREATE TABLE Drivers (
		Id INT IDENTITY(1,1) PRIMARY KEY,
		[Name] NVARCHAR(64) NOT NULL,
		Phone NVARCHAR(17) NOT NULL,
		Count_of_Order INT NOT NULL,
	);
	go
	CREATE TABLE Accounts (
		Id INT IDENTITY(1,1) PRIMARY KEY,
		Email NVARCHAR(64) NOT NULL,
		[Password] NVARCHAR(32) NOT NULL,
		Id_Driver INT NOT NULL,
	);
	go
	CREATE TABLE [Log] (
		Id INT IDENTITY(1,1) PRIMARY KEY,
		Order_Number INT NOT NULL,
		End_Date VARCHAR(10) NOT NULL,
		Archive_id INT NOT NULL,
		Income INT NOT NULL
	);
	go
	CREATE TABLE Archive (
		Id INT IDENTITY(1,1) PRIMARY KEY,
		Driver_name NVARCHAR(64) NOT NULL,
		Car_Number NVARCHAR(16) NOT NULL,
		Order_Points NVARCHAR(1024) NOT NULL,
	);
	go
	CREATE TABLE Taxes (
		[Name] NVARCHAR(32) NOT NULL,
		[Value] NVARCHAR(32) NOT NULL,
	);
	go
	ALTER TABLE Details
	ADD CONSTRAINT FK_Details_To_Orders FOREIGN KEY(Id_Order) REFERENCES Orders(Id);
	go
	ALTER TABLE Orders
	ADD CONSTRAINT FK_Orders_To_Addresses_From FOREIGN KEY(Id_Point_From) REFER-
ENCES Addresses(Id),
		CONSTRAINT FK_Orders_To_Addresses_To FOREIGN KEY(Id_Point_To) REFERENCES
Addresses(Id);
	go
	ALTER TABLE Nodes
	ADD CONSTRAINT FK_Nodes_To_Orders FOREIGN KEY(Id_Order) REFERENCES Orders(Id),
		CONSTRAINT FK_Nodes_To_Cars FOREIGN KEY(Id_Car) REFERENCES Cars(Id),
		CONSTRAINT FK_Nodes_To_Drivers FOREIGN KEY(Id_Driver) REFERENCES Driv-
ers(Id);
	go
	ALTER TABLE Accounts
	ADD CONSTRAINT FK_Accounts_To_Drivers FOREIGN KEY(Id_Driver) REFERENCES Driv-
ers(Id) ON DELETE CASCADE;
	go
	ALTER TABLE [Log]
	ADD CONSTRAINT FK_Log_To_Archive FOREIGN KEY(Archive_Id) REFERENCES Archive(Id);
	go
	ALTER TABLE Movers
	ADD CONSTRAINT FK_Mover_To_Order FOREIGN KEY(Id_Order) REFERENCES Orders(Id);
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Отчет к лабораторной работе №2

Разработка функциональной модели

В ходе анализа предметной области была разработана следующая функциональная модель приложения (см. рис. Б.1).

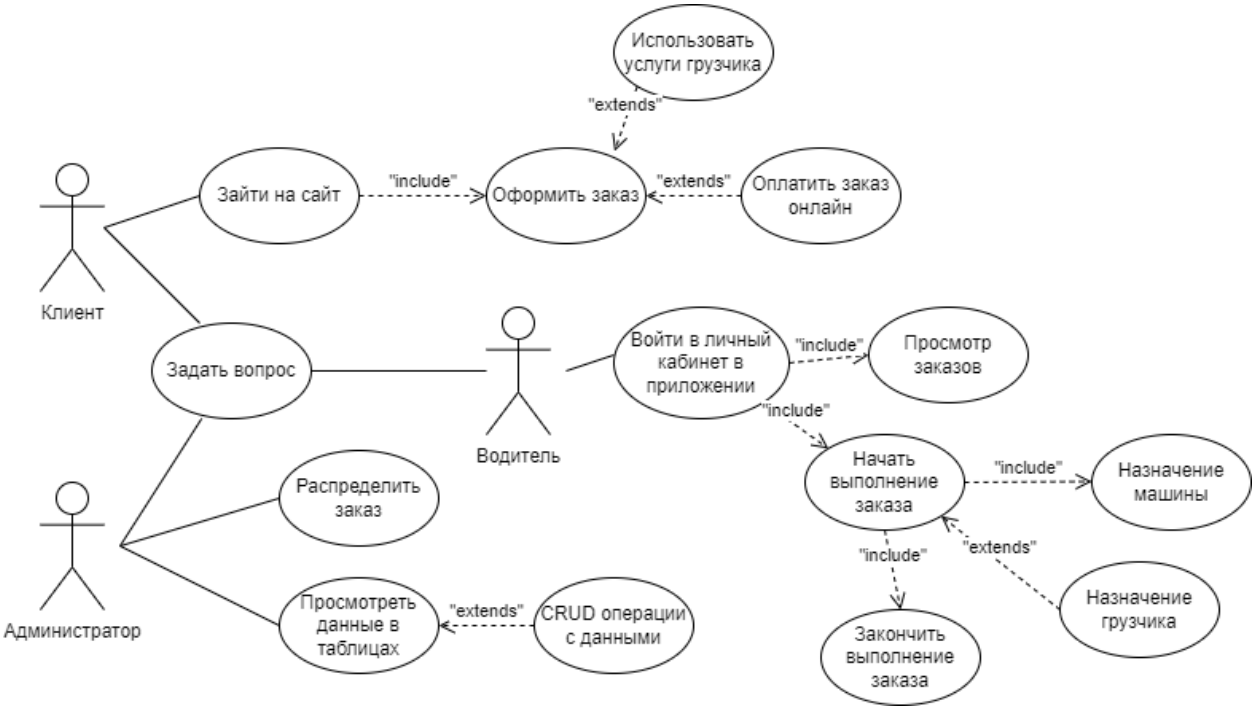

Рисунок Б.1 – Диаграмма функциональной модели.

Функциональная модель состоит из набора диаграмм потока данных, которые показывают потоки значений от внешних входов через операции и внутренние хранилища данных к внешним выходам. Функциональная модель описывает смысл операций объектной модели и действий динамической модели, а также ограничения на объектную модель. Функциональная модель позволяет наглядно показать, какие действия будут доступны конкретным ролям в разрабатываемом приложении.

Разработка модели данных

В ходе анализа предметной области также была продумана и разработана следующая модель данных (см. рис Б.2).

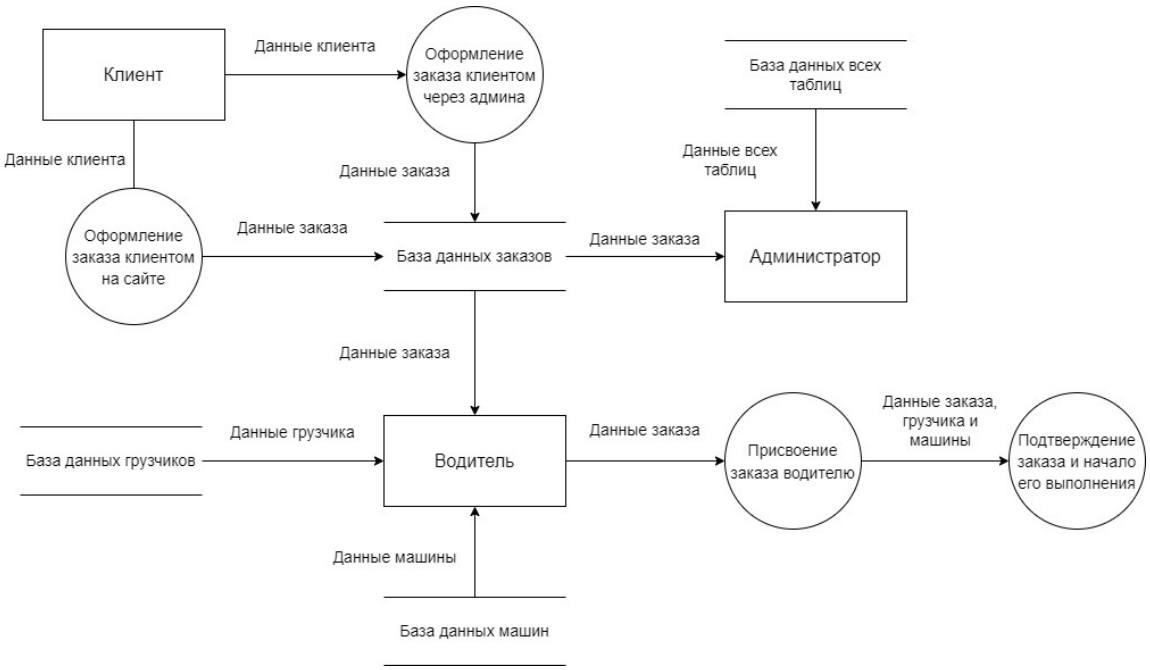

Рисунок Б.2 – Диаграмма модели данных.

Модель данных – это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных.

Проектирование приложения

Исходя из разработанной функциональной модели, модели данных, логической модели БД можно перейти к проектированию самого приложения. Для этого были продуманы и разработаны на языке UML следующие диаграммы:

– диаграммы классов;
– диаграммы состояний;
– диаграмма активности;
– диаграмма компонентов.

Диаграмма классов – это структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их кооперацией, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними (см. рис. Б.3). Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

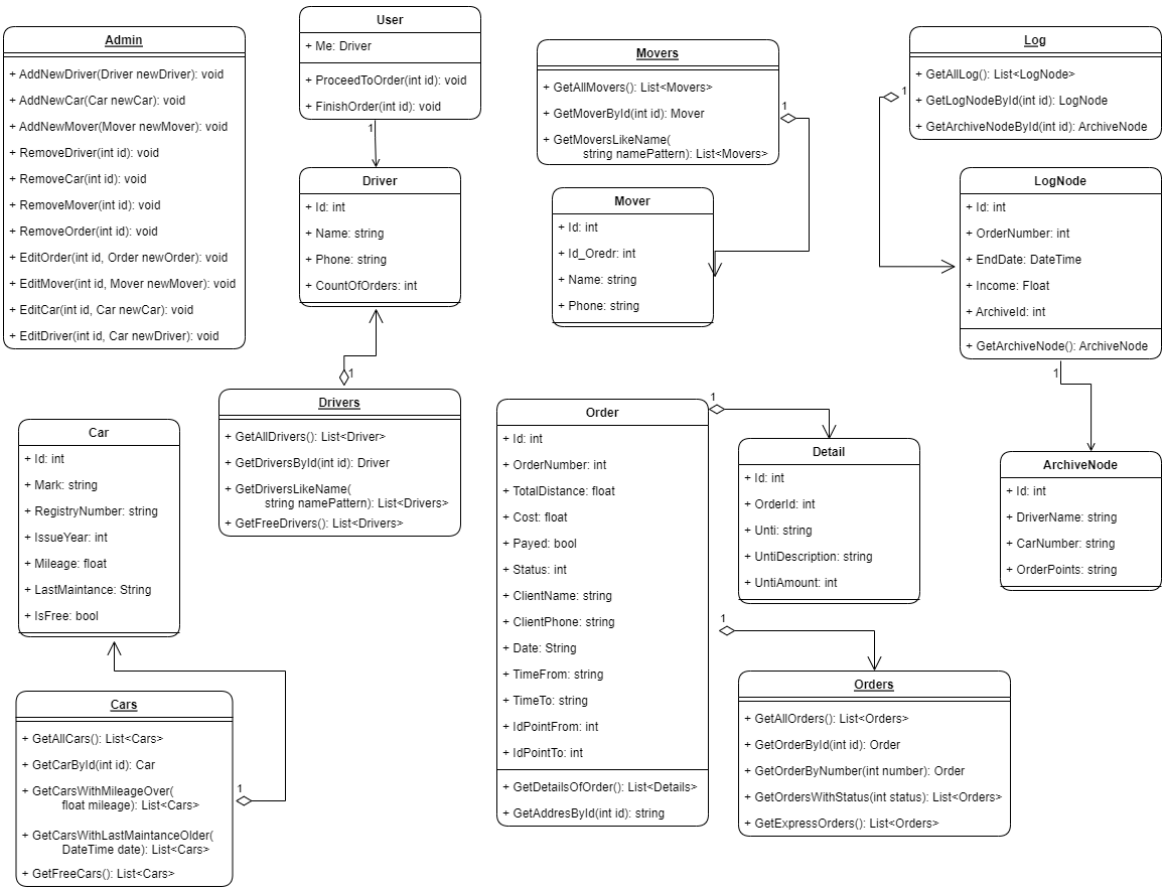

Рисунок Б.3 – Диаграмма классов (UML).

Основными классами в нашей программе будут классы, отражающие объекты и сущности базы данных. Это классы Order, Detail, Driver, Car, Mover, LogNode и ArchiveNode. Помимо этого, необходимы классы, которые будут управлять этими объектами: создавать их, собирать в коллекции, получать и отправлять их в базу данных. Таковыми являются классы Cars, Orders, Movers, Drivers, Log, Admin и User.

Диаграмма состояний (UML) – это, по существу, диаграмма состояний из теории автоматов со стандартизированными условными обозначениями, которая может определять множество систем от компьютерных программ до бизнес-процессов (см. рис. Б.4). Используются следующие условные обозначения:

1 Круг, обозначающий начальное состояние.

2 Окружность с маленьким кругом внутри, обозначающая конечное состояние (если есть).

3 Скруглённый прямоугольник, обозначающий состояние. Верхушка прямоугольника содержит название состояния. В середине может быть горизонтальная линия, под которой записываются активности, происходящие в данном состоянии.

5 Стрелка, обозначающая переход. Название события (если есть), вызывающего переход, отмечается рядом со стрелкой.

6 Толстая горизонтальная линия с либо множеством входящих линий и одной выходящей, либо одной входящей линией и множеством выходящих. Это обозначает объединение и разветвление соответственно.

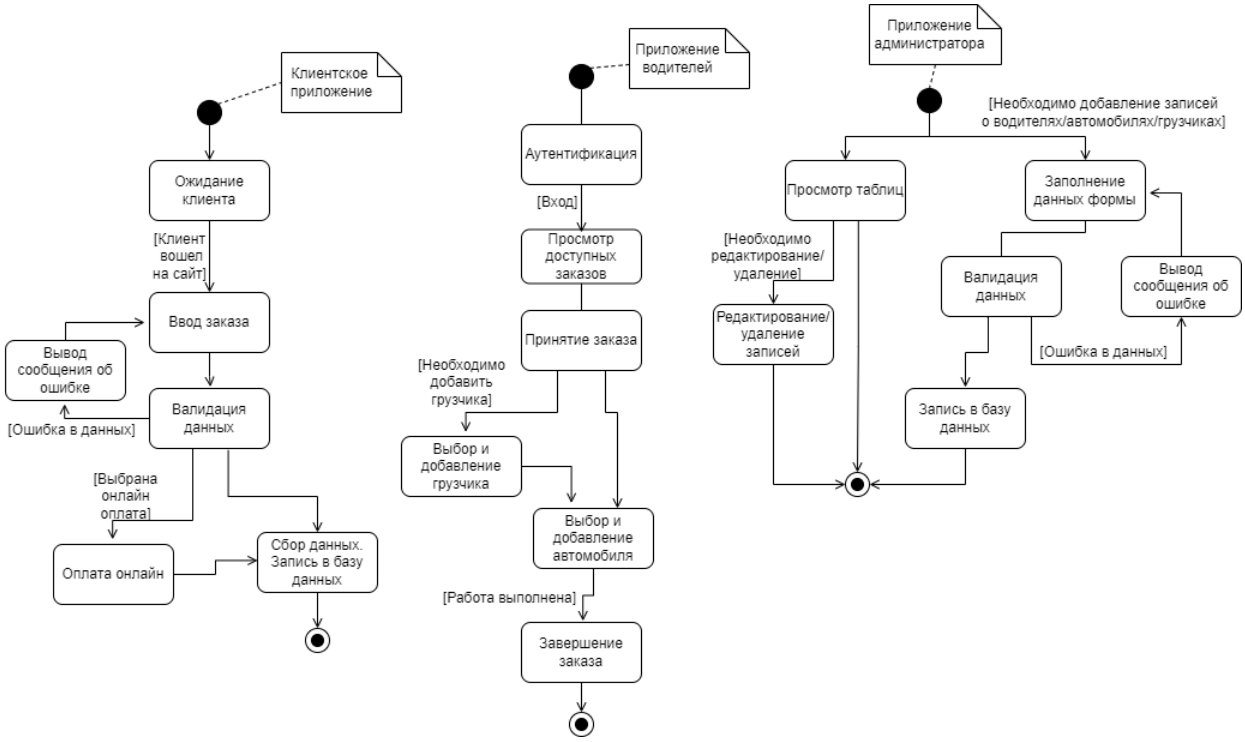

Рисунок Б.4 – Диаграмма состояний (UML).

На диаграмме, представленной на рисунке Б.4, продемонстрированы основные состояния нашей программы. Исходя из особенностей нашего приложения, имеются три точки входа и три точки выхода, т.к. приложения для клиента, водителя и администратора будут реализовывать свой функционал и находится на разных носителях.

Диаграмма активностей – это UML-диаграмма, на которой показаны действия, состояния которых описаны на диаграмме состояний (см. рис. Б.5). Под деятельностью (англ. activity) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий англ. action, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграммы деятельности состоят из ограниченного количества фигур, соединённых стрелками. Основные фигуры (узлы):

1 Прямоугольники с закруглениями — действия (операция). Узел управления (control node) — это абстрактный узел действия, которое координирует потоки действий.

2 Ромбы — решения. Узел решения предназначен для определения правила ветвления и различных вариантов дальнейшего развития сценария. В точку ветвления входит ровно один переход, а выходит — два или более.

3 Широкие полосы — начало (разветвление) и окончание (схождение) ветвления действий. Узел объединения имеет два и более входящих узла и один исходящий.

4 Чёрный круг — начало процесса (начальный узел). Начальный узел деятельности (или начальное состояние деятельности) (activity initial node) является узлом управления, в котором начинается поток (или потоки) при вызове данной деятельности извне.

5 Чёрный круг с обводкой — окончание процесса (финальный узел). Конечный узел деятельности (или конечное состояние деятельности) (activity final node) является узлом управления, который останавливает все потоки данной диаграммы деятельности. На диаграмме может быть более одного конечного узла.

Стрелки идут от начала к концу процесса и показывают потоки управления или потоки объектов (данных).

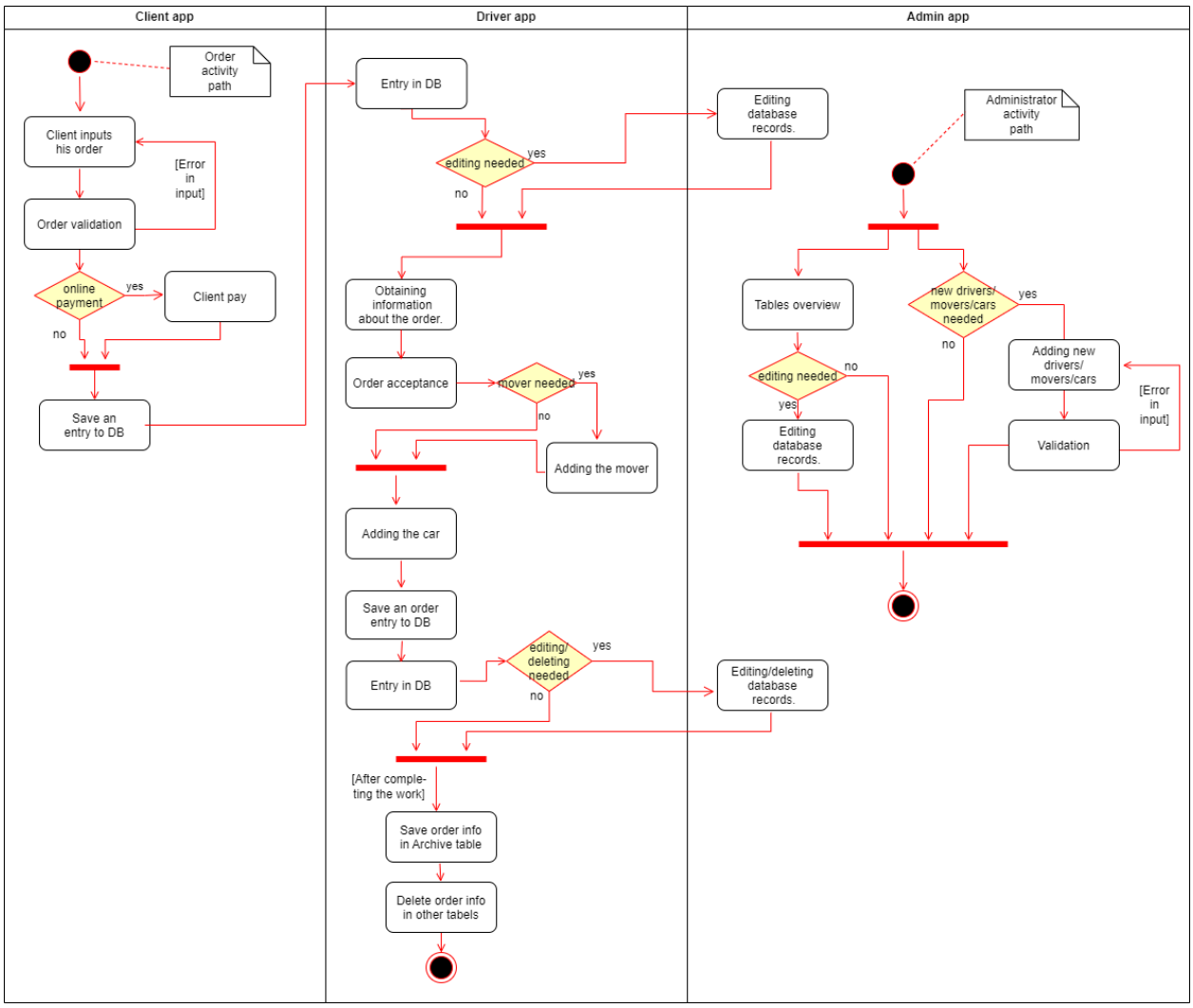

Рисунок Б.5 – Диаграмма активностей (UML).

Диаграмма активностей, представленная на рисунке Б.5, показывает путь активности всей сети приложений для одного отдельно взятого заказа. Для наглядности она также поделена на три секции, отражающие приложение клиента, водителя и администратора.

Диаграмма компонентов (UML) – это элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами (см. рис. Б.6). В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

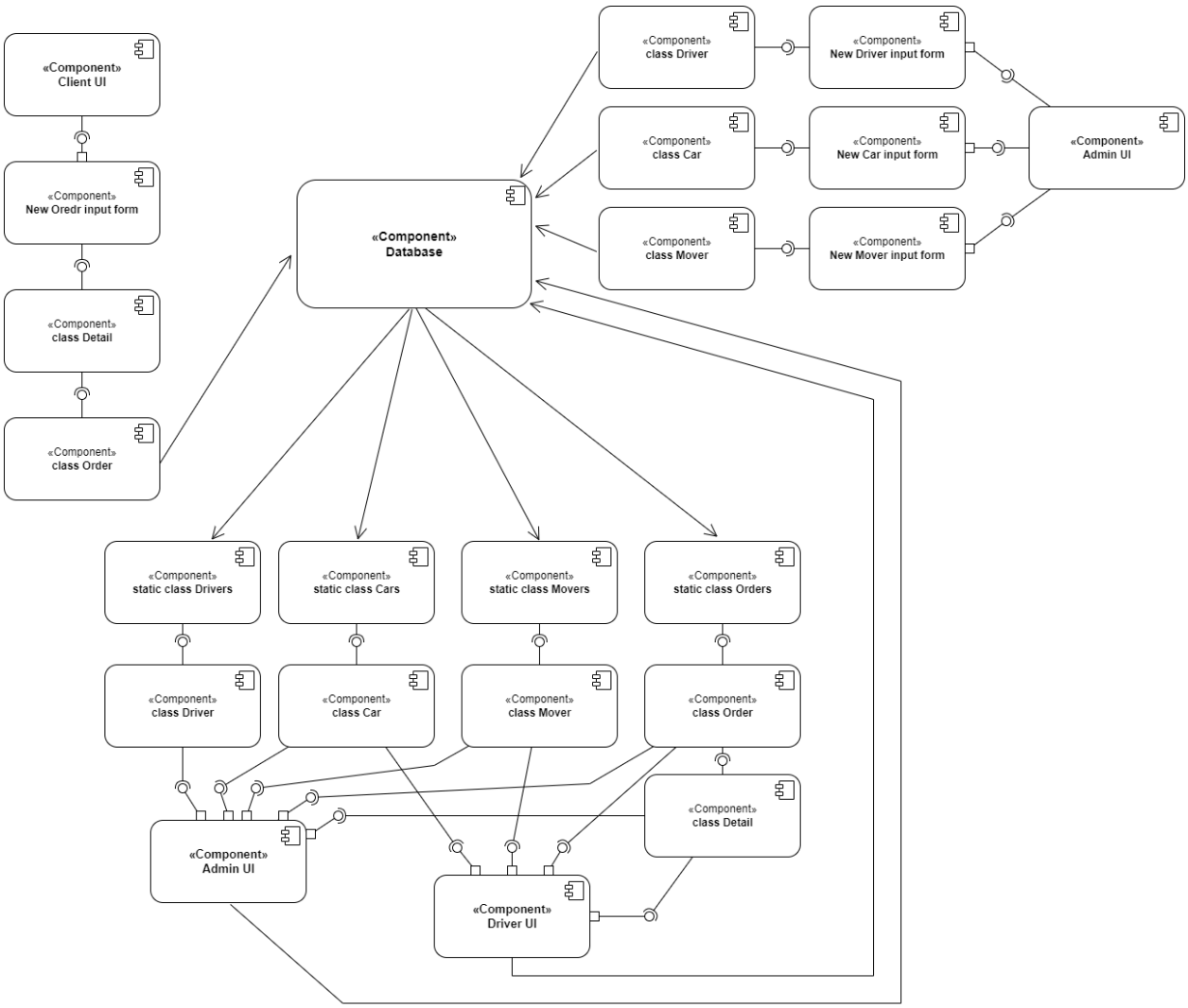

Рисунок Б.6 – Диаграмма компонентов (UML).

Диаграмма компонентов показывает, каким образом происходит взаимодействие компонентов пользовательского интерфейса и главного «сердца» приложения – компонента базы данных, а также всех связующих их компонентов.

ПРИЛОЖЕНИЕ В
(обязательное)
Отчет к лабораторной работе №3

Теоретические сведения

Методология IDEF3 является одним из стандартов семейства IDEF и довольно широко используется при декомпозиции моделей IDEF0 для моделирования процессов более низкого уровня, поскольку с его помощью можно смоделировать технологические процессы, происходящие на предприятии, т.е. описать возможные сценарии реализации процессов, в рамках которых происходит последовательное изменение свойств объекта. Данная методология позволяет показывать возможные разветвления в процессе. Например, когда результат одного действия может инициировать запуск нескольких действий или наоборот, чтобы начать какое-то действие, необходимо завершить несколько предыдущих действий. Модели IDEF3 можно отнести к классу WFD-диаграмм, поскольку с их помощью также описывается взаимосвязанная последовательность действий, которые осуществляются в рамках реализации процесса.

В рамках стандарта IDEF3 выделяют два типа диаграмм, позволяющих описать процесс с разных точек зрения:

– диаграмма описания последовательности этапов процесса (Process Flow Description Diagrams — PFDD), с помощью которой моделируется последовательность действий, реализуемых в рамках бизнес-процесса;

– диаграмма состояния и трансформации объекта в процессе (Object State Transition Network — OSTN), с помощью которой описываются изменения, происходящие с объектом в ходе его обработки.

Основными элементами диаграммы PFDD IDEF3 (далее — IDEF3) являются:

– функциональный блок;

– элемент;

– стрелка;

– перекресток.

Функциональный элемент (элемент поведения, единица работы) используется для обозначения действия, работы или события. Он отражается в виде прямоугольника, в центре которого указывается название действия (глагол или отглагольное существительное). Внизу блока указывается номер действия с учетом номера родительской диаграммы.

Стрелка (линия) используется для отражения последовательности выполнения работ (действий) и связей между ними. Все стрелки показывают движение в одну сторону: слева направо, таким образом, визуально соблюдая идею демонстрации последовательного выполнения операций процесса. Они могут выходить и входить с любой стороны блока, но предпочтение лучше

отдавать их горизонтальному расположению. Существуют три типа стрелок: временное предшествование, объектный поток, нечеткое отношение.

Стрелка типа "Временное предшествование" показывает, что действие, из которого она выходит, должно завершиться до того, как начнется действие, в которое она входит. Результат исходного действия не обязательно является инициатором для действия, куда входит стрелка. Главное значение данной стрелки — показать временную связь между действиями, т.е. показать, что одно действие не может начаться до того, пока предыдущее не закончится, независимо от результата его завершения. Такая связь обозначается простой стрелкой.

Стрелка типа "Объектный поток" показывает, что результат действия, из которого она выходит, является инициатором действия, в которое оно входит. Соответственно действие, в которое входит стрелка, не может начаться до тех пор, пока не закончится действие, из которого стрелка выходит. Такая связь обозначается стрелкой с двойным наконечником. В названии стрелки должно быть приведено название объекта, который передается от одной операции к другой.

Стрелка типа "Нечеткое отношение" показывает, что тип связи между двумя действиями задается индивидуально, может иметь переменчивый или уникальный характер. Такая связь обозначается пунктирной стрелкой. специальных требований по ее наименованию нет. Такое изображение связей используется, когда нельзя применить связи, типа "Временное предшествование" и "Объектный поток". Перекресток (условные символы ветвления) используется для отражения логики движения потоков между функциональными элементами (операциями).

Перекресток позволяет указать события, которые могут или должны произойти для того, чтобы началось выполнение следующего действия. На диаграмме IDEF3 перекресток представляет собой прямоугольник с индикатором "J" и номером данного перекрестка на диаграмме. Существуют перекрестки, используемые для отражения слияния стрелок, и перекрестки, используемые для отражения разветвления стрелок. Стоит отметить, что один перекресток не может одновременно использоваться для слияния и для разветвления.

В методологии IDEF3 выделяют: разворачивающиеся соединения, используемые для отражения связей, где завершение одного процесса инициирует запуск нескольких других процессов: сворачивающиеся соединения, используемые для отражения связей, где завершение нескольких процессов приводит к запуску следующего одного процесса.

Разворачивающиеся и сворачивающиеся соединения могут быть также нескольких типов:

– «и» (обозначается квадратом с символом «&»);

– «исключающее “или”» (обозначается квадратом с символом «X»);

– «или» (обозначается квадратом с символом «O»).

Диаграмма информационных потоков

Диаграммы информационных потоков для сайта, приложения администратора и приложения водителя указаны на рисунках В.1, В.2, В.3 соответственно.

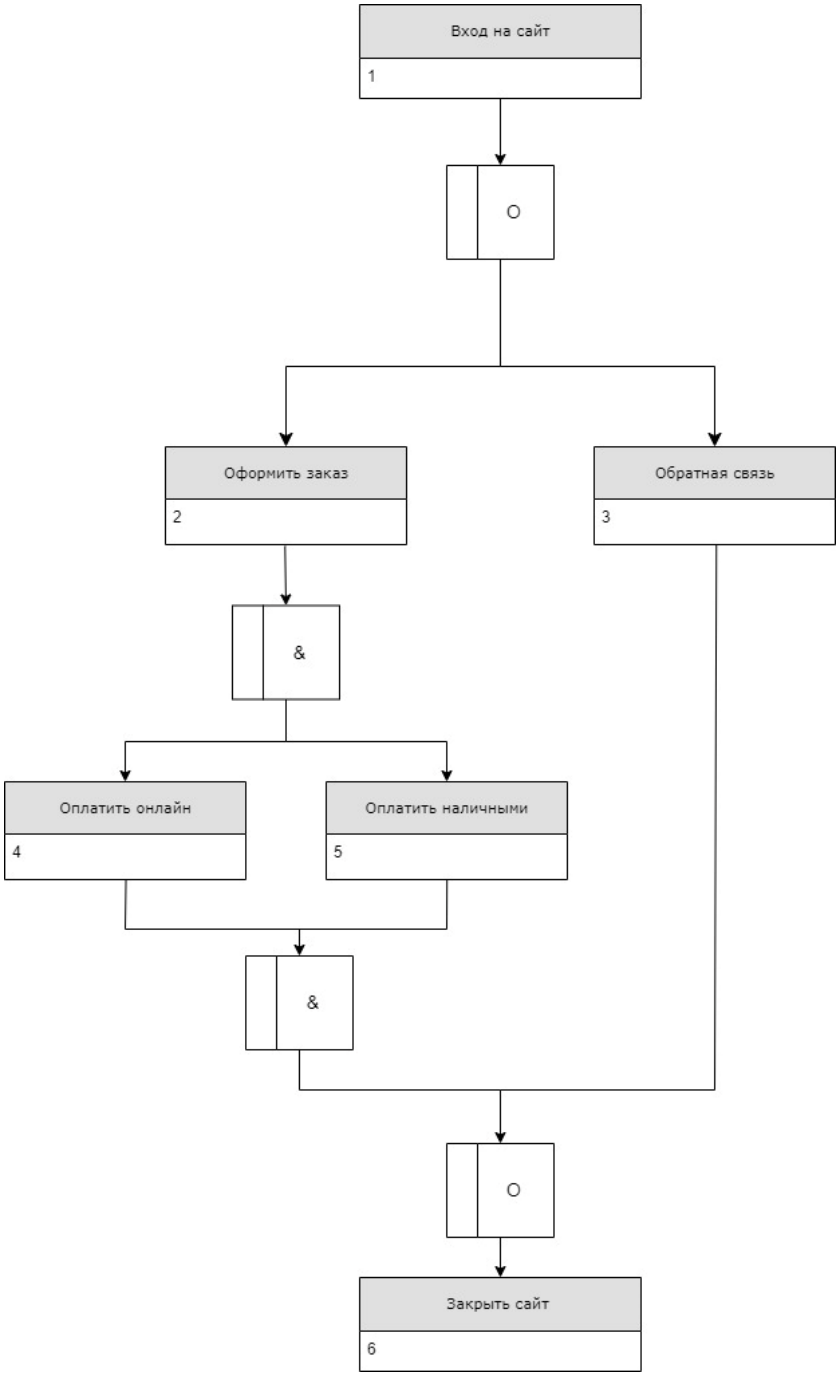

Рисунок В.1 – Диаграмма информационных потоков сайта

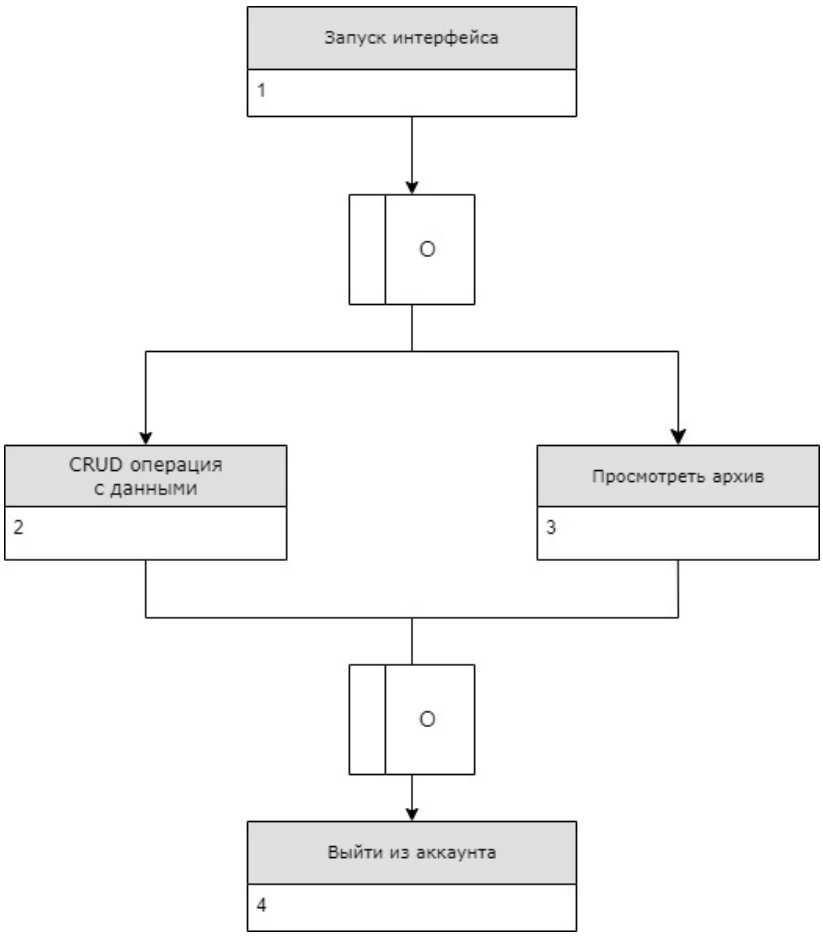

Рисунок В.2 - Диаграмма информационных потоков приложения администратора

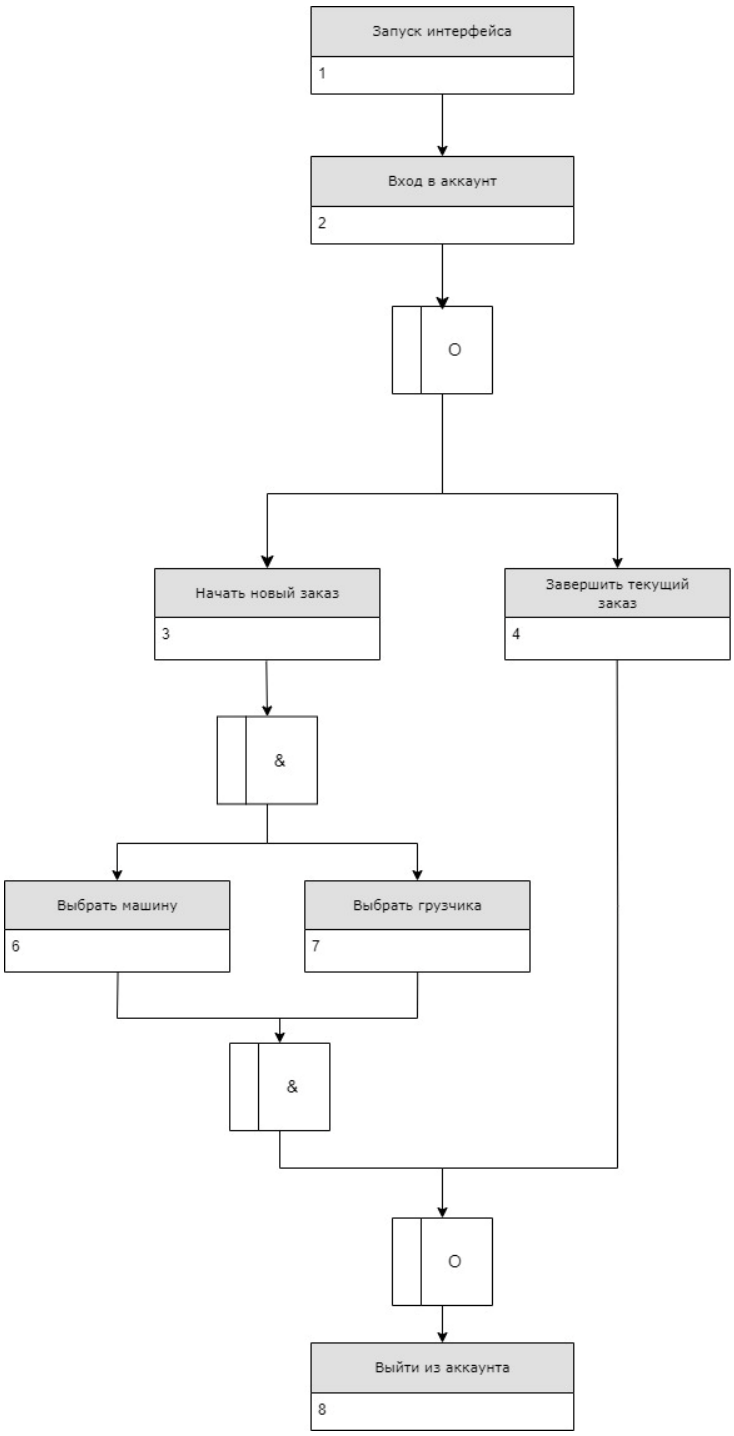

Рисунок В.3 - Диаграмма информационных потоков приложения водителя