МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

Специализация 1–40 01 01 Программное обеспечение информационных технологий (программирование интернет – изданий)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:

База данных с технологией «шифрование и маскирования в БД» для программного средства «Таксокомпания»

Выполнил студент Яскович Марк Эдуардович

(Ф.И.О.)

Руководитель проекта асс., Комарова Е. И.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант:  асс., Комарова Е. И.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: асс., Комарова Е. И.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2022

**Реферат**

Пояснительная записка курсового проекта содержит 30 страниц пояснительной записки, 8 источников литературы, 3 приложения.

C#, .NET FRAMEWORK 4.7.2, Windows FORMS, Microsoft SQL Server, ADO.NET FOR C#, XML IMPORT, XML EXPORT.

Цель курсового проекта: разработка базы данных для таксокомпании с применением технологии шифрования и маскирования данных.

Первая глава описывает приложения, которыми был вдохновлен курсовой проект, и пункты, которые следует имплементировать.

Вторая глава описывает процесс создания и настройки базы данных, с которой будет происходить взаимодействие.

Третья глава описывает разработку объектов для базы данных.

Четвёртая глава описывает используемые процедуры

В пятой главе описывается используемая технология;

В шестой главе происходит тестирование производительности

В седьмой главе приведено руководство пользователя по взаимодействию с приложением и описаны базовые тестовые случаи.

В заключении приведены результаты проделанной работы.

Содержание

[Введение 4](#_Toc122380944)

[1. Постановка задачи 5](#_Toc122380945)

[1.1 Анализ прототипов 5](#_Toc122380946)

[2. Разработка модели базы данных 7](#_Toc122380947)

[3. Разработка необходимых объектов 9](#_Toc122380948)

[3.1. Проектирование базы данных 9](#_Toc122380949)

[3.2. Процедуры для решения поставленных задач 10](#_Toc122380950)

[3.2.1. Выборка данных из таблиц 10](#_Toc122380951)

[3.2.2. Добавление данных в таблицу 10](#_Toc122380952)

[3.2.3. Изменение данных в таблице 11](#_Toc122380953)

[3.2.4. Удаление данных из таблицы 12](#_Toc122380954)

[3.2.5. Выборка данных по поисковому запросу 12](#_Toc122380955)

[3.3. Индексы 12](#_Toc122380956)

[4. Описание процедур экспорта и импорта 15](#_Toc122380957)

[5. Описание и реализация технологии 17](#_Toc122380958)

[6. Тестирование производительности 19](#_Toc122380959)

[7. Руководство пользователя 20](#_Toc122380960)

[7.1 Установка программного средства 20](#_Toc122380961)

[7.2 Регистрация и авторизация 21](#_Toc122380962)

[7.2 Руководство пользователя по работе с программным средством 22](#_Toc122380963)

[Заключение 25](#_Toc122380964)

[Список используемых источников 26](#_Toc122380965)

[Приложение А 27](#_Toc122380966)

[Приложение Б 29](#_Toc122380967)

[Приложение В 30](#_Toc122380968)

## **Введение**

В наше время всё больше информации хранится в цифровом виде и для хранения такой информации отлично подходят различные базы данных. Использование базы данных упрощает управление данными и их хранение. Одним из примеров успешного внедрения баз данных является использование приложения для заказа такси.

Сейчас популярностью пользуется приложение Яндекс Go, позволяющее без особых усилий оформить такси из любой точки города и сохранять данные о поездках.

В ходе выполнения данной курсовой работы было разработано приложение «Таксокомпания».

Цель данного курсового проекта – создание базы данных для таксокомпании для взаимодействия водителей и пользователей ознакомление с технологией шифрования и маскирования в БД и её применение в базе данных, а также разработка удобного программного средства для обработки этой информации. Готовое программное средство должно предоставлять возможность пользователям оформить заказ, а водителям принять его.

Основными задачами курсовой работы являются:

* + - провести аналитический обзор прототипов;
    - спроектировать базу данных;
    - реализовать функциональность базы данных;
    - провести тестирование используемой технологии в базе данных;
    - разработать приложение для работы с базой данных;
    - написать руководство пользователя.

Для проектирования базы данных используется СУБД «Microsoft SQL Server 2019». В базе данных применяется технология мультимедийных типов данных. Приложение для работы с базой данных было создано на платформе Windows Forms. Для работы с Windows Forms использовался объектно-ориентированный язык программирования С#. Для осуществления связи между базой данных и приложением, используется технология ADO.

## **Постановка задачи**

Основной задачей курсового проекта является разработка программного средства, позволяющего пользователю оформить заказ и принять заказ водителю.

# **Анализ прототипов**

Яндекс GO – самое популярное в СНГ приложение для заказа такси и аренды грузовых автомобилей.

Достоинства:

1. приятный и удобный интерфейс;
2. возможность выбрать свое местоположение на карте.

Недостатки:

1. неизвестно примут ли заказ или нет;
2. наличие только на мобильных устройствах.

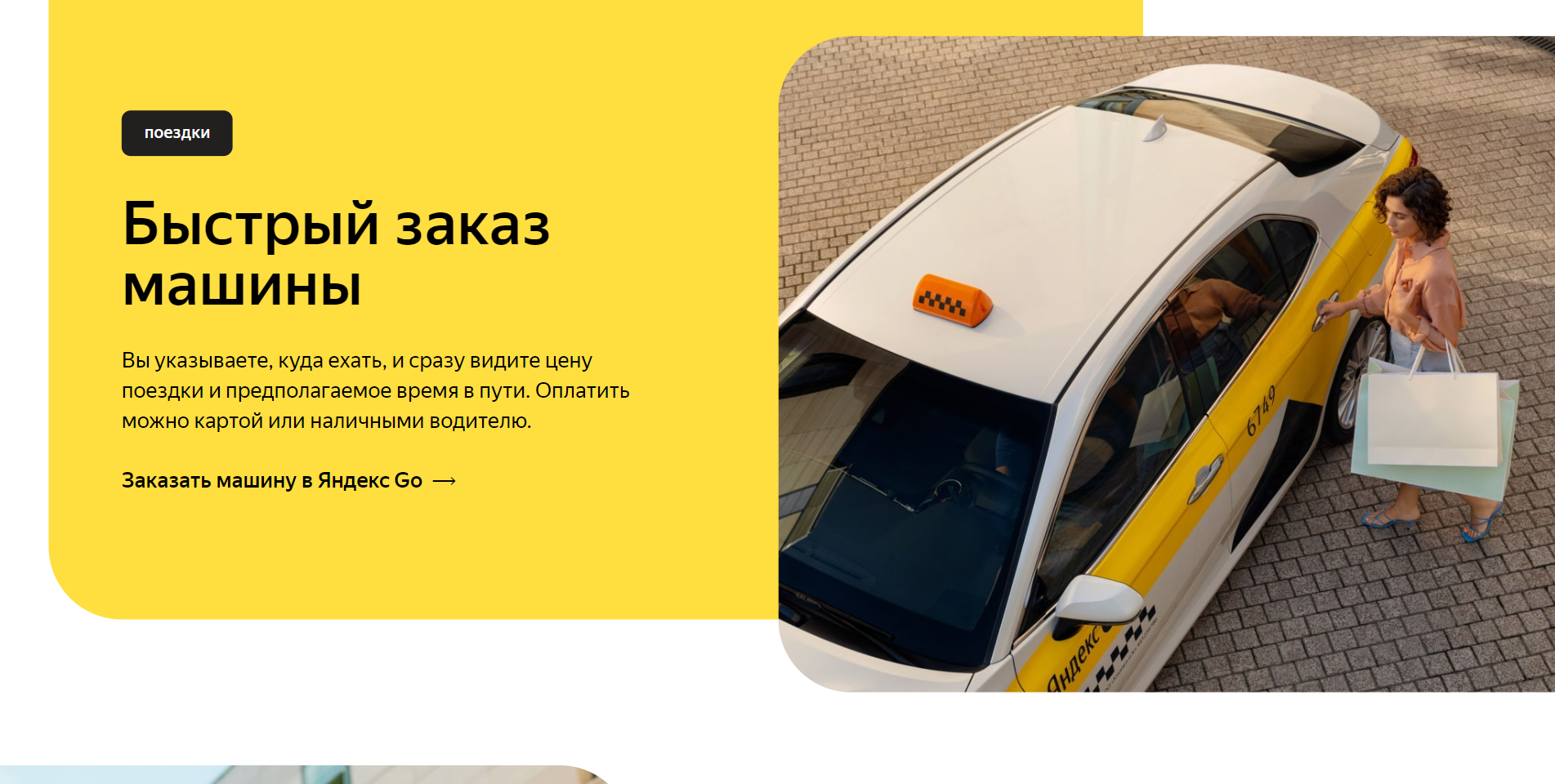


Рисунок 1.1 – Интерфейс веб-приложения «Яндекс-Go»

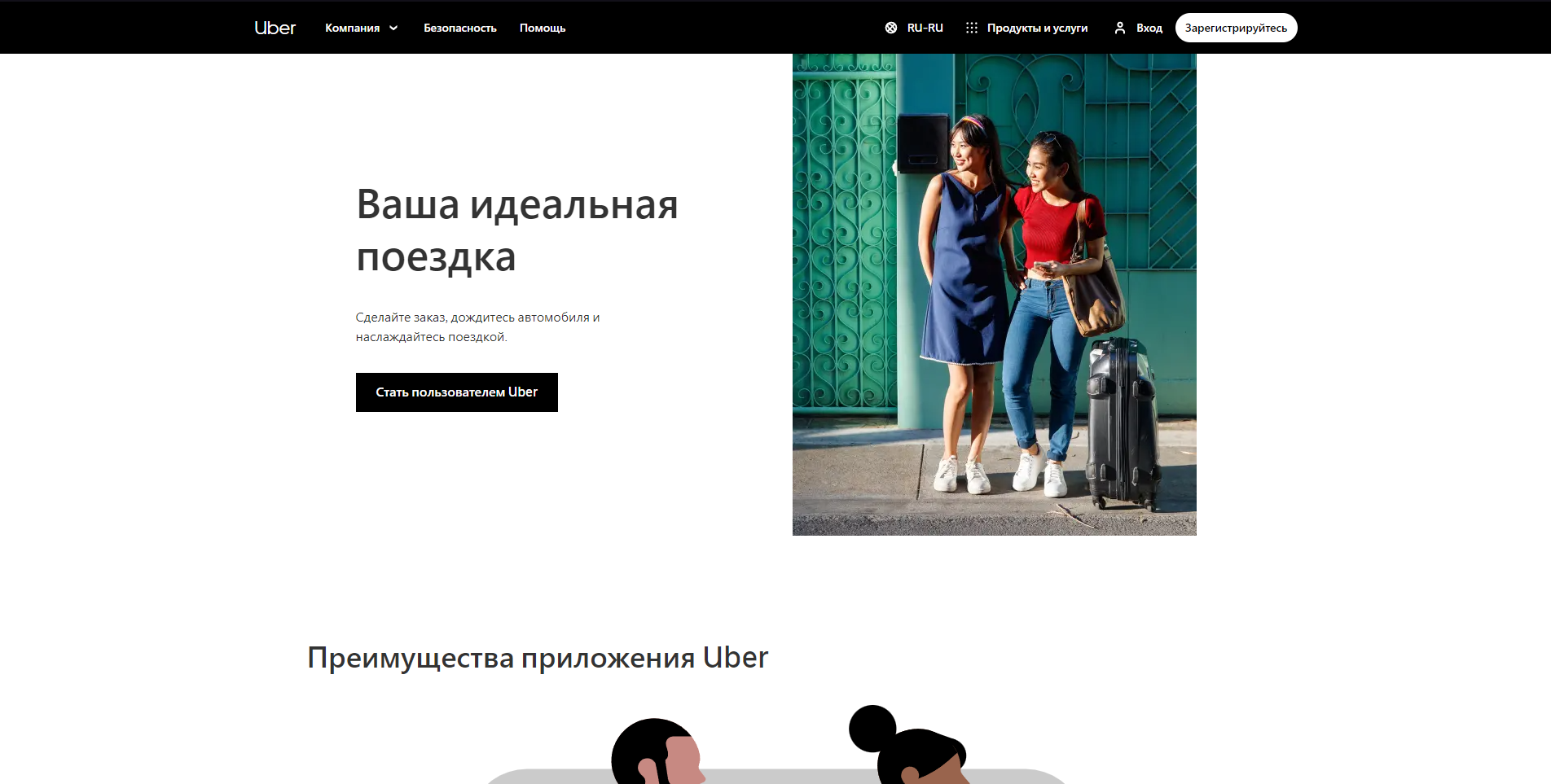
Uber - бывшие популярное приложение, в последствии поглощено Яндексом

Достоинства:

1. приятный и удобный интерфейс;
2. возможность выбрать свое местоположение на карте;
3. цены ниже чем у «Яндекс-Go»

Недостатки:

1. неудобное веб-приложение
2. наличие только на мобильных устройствах.

Рисунок 1.2 – Интерфейс веб-приложения «Uber»

135.by – менее популярное, но удобный белорусский сервис

Достоинства:

1. приятный и удобный интерфейс;
2. наличие удобного веб-приложения, с картой;

Недостатки:

1. наличие только на мобильных устройствах и веб-приложении;

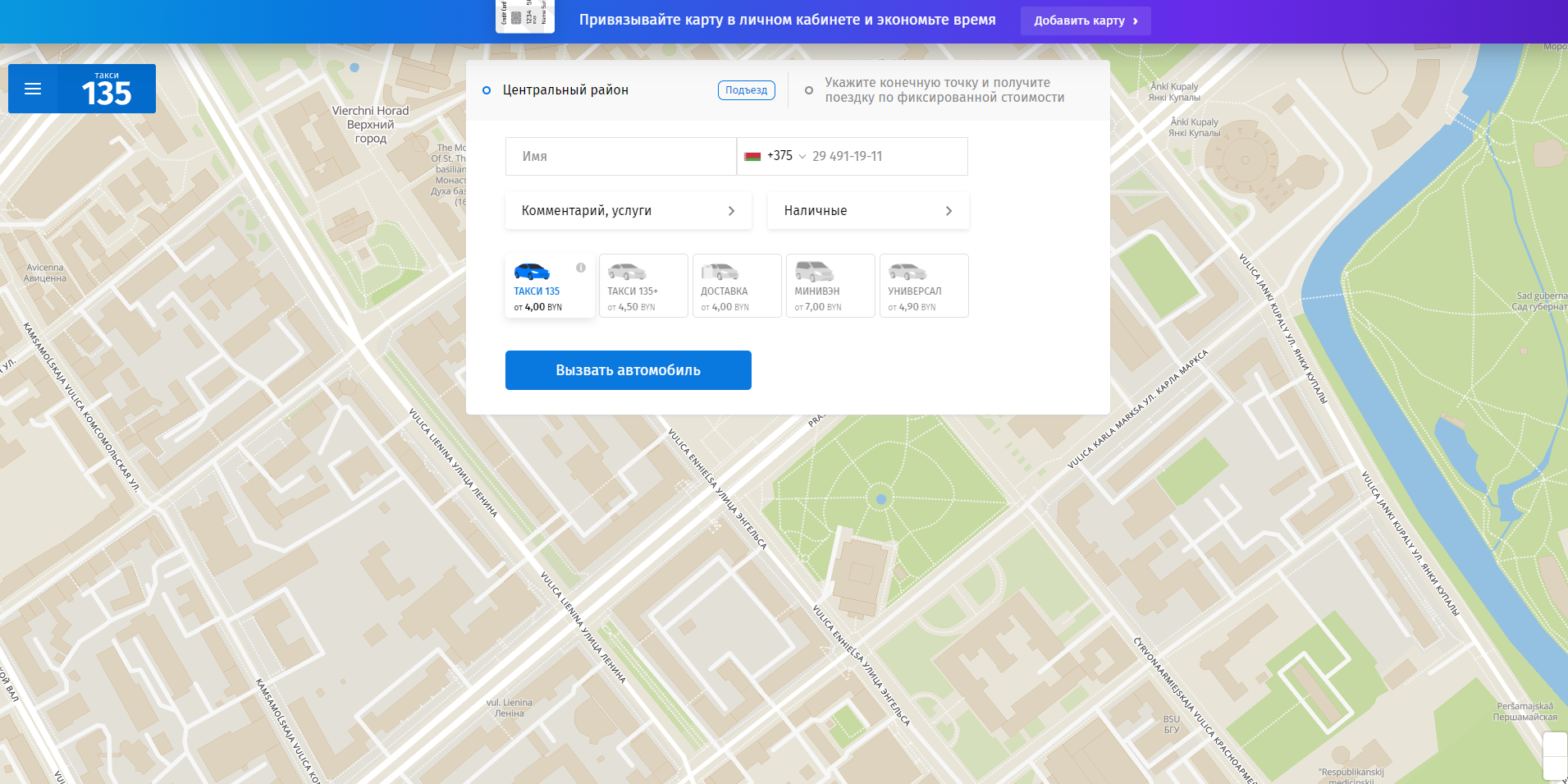


Рисунок 1.3 – Интерфейс веб-приложения «135.by»

## **Разработка модели базы данных**

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Анализируя ранее приведённые примеры, я составил основные требования. Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

* простота и лёгкость интерфейса;
* использование принципов объектно-ориентированного программирования;
* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы.

Неавторизованный пользователь данного программного решения должен иметь возможность:

* регистрировать себя в системе;
* авторизоваться;
* выходить из приложения.

Авторизованный пользователь может:

* просматривать и изменять личные данные;
* просматривать историю поездок;
* оформить заказ;
* отследить активный заказ;

Администратор имеет возможность:

* просматривать уже существующие заказы;
* удалять заказы в которых нет надобности;
* создавать новые заказы;
* изменять существующие заказы.

Более наглядно это видно на UML-схеме (Приложение А).

После проведения анализа были выявлены следующие функциональные требования:

* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

## **Разработка необходимых объектов**

## **Проектирование базы данных**

Для реализации поставленной задачи была создана база данных teest. Для её создания использовалась система управления реляционными базами данных Microsoft SQL Server 2019.

Прежде всего, необходимо было спроектировать корректную базу данных для работы. Моя база данных содержит следующие таблицы (рисунок 3.1):

* личные данные пользователей (Registers);
* информация о заказах (Orders);
* личные данные водителей (Drivers);
* данные администратора (Administrators);
* данные о совершенных поездках (History);

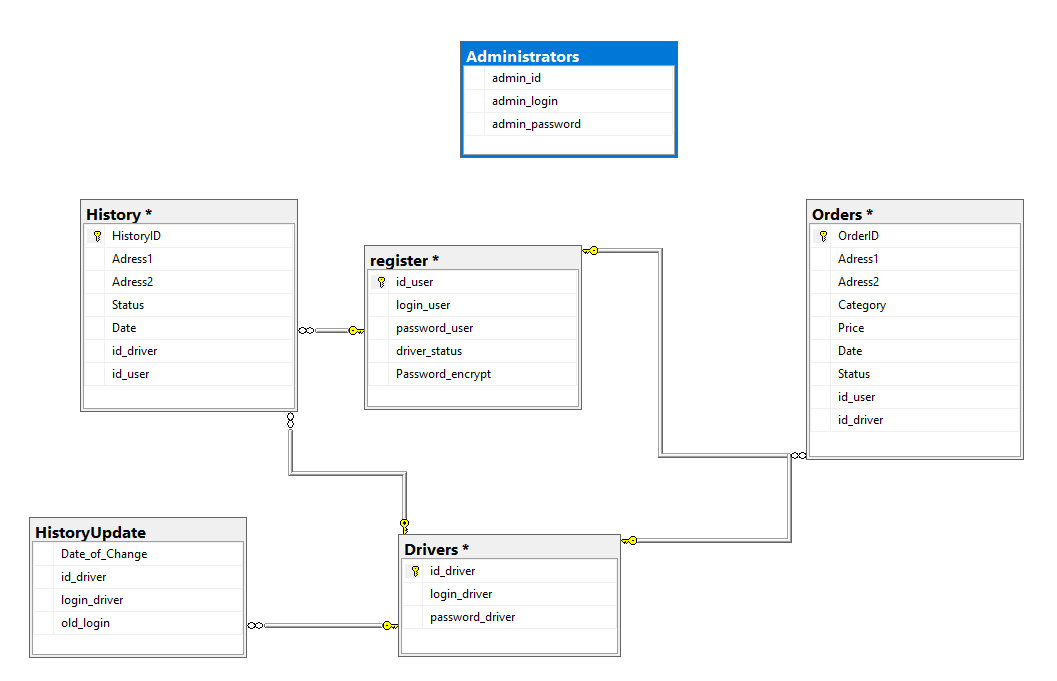


Рисунок 3.1 – Диаграмма базы данных

Листинг создания базы данных представлен в приложении Б.

## **Процедуры для решения поставленных задач**

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Использование хранимых процедур позволяет ограничить либо вообще исключить непосредственный доступ пользователей к таблицам базы данных, оставив пользователям только разрешения на выполнение хранимых процедур, обеспечивающих косвенный и строго регламентированный доступ к данным.

При разработке курсового проекта было создано 55 процедур для следующих целей:

1. Выборка данных из таблиц по различным полям;
2. Выборка данных по поисковому запросу;
3. Удаление данных из таблиц;
4. Добавление данных в таблицы;
5. Изменение данных в таблицах;
6. Экспорт и импорт таблиц в формат xml;
7. Заполнение таблицы на 100 000 строк.

Весь перечень созданных процедур будет представлен в Приложении В.

## **Выборка данных из таблиц**

Для выборки данных из таблиц были написаны следующие процедуры: ActiveOrder, CheckUser, HaveAdmin, HaveDrivers, HaveUsers, OutAllCurrentOrders, OutAllUsers, OutHistoryDrivers, OutHistoryUsers, OutOrders, Refresh, RefreshDrivers, RefreshOrders. Поскольку в приложении мы часто обращаемся с поиском к таблицам Order, History, которые содержат большое количество данных было принято решение создать индексы, тем самым сократив время поиска по этим таблицам.

## **Добавление данных в таблицу**

При создании пользователем нового заказа с помощью процедуры AddOrder ему необходимо заполнить все поля формы (Откуда, Куда, Класс авто, Дата) Процедура представлена ниже в листинге 3.1:

ALTER PROC [dbo].[AddOrder]

@Adress1 nvarchar(100),

@Adress2 nvarchar(100),

@Category nvarchar(20),

@Price int,

Продолжение листинга 3.1

@Status nvarchar(50),

@Date date,

@Id int

as

INSERT INTO Orders( Adress1, Adress2, Category, Price,Date, id\_user, Status)

values( @Adress1,@Adress2,@Category,@Price,@Date, @Id, @Status)

Листинг 3.1 – Описание процедуры AddOrder

При регистрации нового пользователя с помощью процедуры AddUser, ему необходимо заполнить необходимые поля: login, password. Для увеличения безопасности учетной записи пользователя его пароль шифруется. Также проверяется корректность ввода логина на повторение. Данная процедура представлена ниже в листинге 3.2:

ALTER PROC [dbo].[AddUser]

@login\_user varchar(50),

@password\_user varchar(50)

as

INSERT INTO register(login\_user, password\_user, driver\_status)  
values(@login\_user, @password\_user, 0)

Листинг 3.2 – Описание процедуры AddUser

Есди же все поля заполнены корректно, то процедура создаст нового пользователя с ролью User

## **Изменение данных в таблице**

Для управления базой данных были созданы процедуры обновления: CanceledOrder, DonedOrder, UpdateDriver, UpdateOrderInProcess, UpdateOrders, UpdateUsers . Создание одной из таких процедур представлена ниже в листинге 3.3:

ALTER proc [dbo].[UpdateDriver]

@id int,

@login\_driver nvarchar(50),

@password\_driver nvarchar(50)

as

update Drivers set login\_driver = @login\_driver, password\_driver = @password\_driver where id\_driver = @id;

Листинг 3.3 – Создание процедуры UpdateDriver

## **Удаление данных из таблицы**

Процедура, разработанная в рамках курсового проекта, выполняющая удаление данных: DeleteFromApproveProduct, она позволяет администратору удалять из таблицы заказы. Скрипт создания этой процедуры приведён ниже в листинге 3.4:

ALTER proc [dbo].[DeleteOrder]

@id int

as

delete from Orders where OrderID = @id ;

Листинг 3.4 – Описание процедуры DeleteOrder

### **3.2.5.** **Выборка данных по поисковому запросу**

Для вывода данных из таблиц по поисковому запросу была написана процедуры Search, SearchDriver, SearchOrders, SearchOrderForAdmin, SearchUser.

ALTER proc [dbo].[SearchOrderForAdmin]

@search nvarchar(50)

as

select \* from Orders where concat(Adress1, Adress2, Date, Status, Category, Price, id\_driver, id\_user) like '%' + @search + '%';

Листинг 3.5 – Описание процедуры SearchOrderForAdmin

## **Индексы**

Индекс – объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. В связи с необходимостью выборки таблиц, заполненных большим количеством строк, было принято решение разработать 1 индекс для таблиц: REGISTER.

* 1. **Триггеры** Для корректной работы с данными, необходимо убирать лишние пробелы.

Триггер – это особый вид хранимой процедуры, предназначенной для

обработки событий в БД, в нашем случае конкретно изменение поля веса.

Для этого в проекте был разработан триггер Delete\_space, который удаляет пробелы при добавлении или обновлении данных.

Пример создания триггера приводится в листинге 3.6:

go

CREATE TRIGGER Delete\_space

ON register

after update, insert

as

begin

declare @login nvarchar(50)

begin

insert into register (login\_user) values (trim(@login));

update register set login\_user = trim(@login);

end;

end;

Листинг 3.6 – Скрипт триггера Delete\_space

Так как в приложении будет находится много пользователей и водителей, необходимо следить за их действиями..

Для этого в проекте был разработан триггер Driver\_History, который фиксирует в отдельную таблицу все изменения. Таким образом, благодаря триггерам мы можем не переживать, что данные изменятся, но не зафиксируются в истории, что в дальнейшем позволит пользователю получить достоверную статистику.

Пример создания триггера приводится в листинге 3.7:

create or alter trigger Driver\_History

on Drivers

after update, insert

as

begin

DECLARE @oldLogin nvarchar(50)

SELECT @oldLogin = (SELECT login\_driver FROM deleted)

DECLARE @newLogin nvarchar(50)

SELECT @newLogin = (SELECT login\_driver FROM inserted)

DECLARE @id int

SELECT @id = (SELECT id\_driver FROM inserted)

IF @oldLogin != @newLogin

begin

INSERT INTO HistoryUpdate(Date\_of\_Change, id\_driver, login\_driver, old\_login ) VALUES( GETDATE(), @id, @newLogin, @oldLogin)

end

end

Листинг 3.7 – Скрипт создания процедуры Users\_History

## **Описание процедур экспорта и импорта**

Для экспорта таблицы Users в формате xml была разработана процедура ProductToXml. Для формирования xml в select запросе используется конструкция FOR XML. Для вывода в файл была использована расширенная хранимая процедура xp\_cmdshell. Процедура ExProdToXml представлена ниже в листинге 4.1.

ALTER procedure [dbo].[ExProdToXml]

as

begin

select login\_user, password\_user, driver\_status from register

for xml path('USER'), root('USERS');

EXEC master.dbo.sp\_configure 'show advanced options', 1

reconfigure with override

EXEC master .dbo.sp\_configure 'xp\_cmdshell', 1 reconfigure with override;

declare @fileName nvarchar(100)

declare @sqlStr varchar(1000)

declare @sqlCmd varchar(1000)

set @fileName = 'C:\Users\Lenovo\AppData\Local\Export.xml';

set @sqlStr = 'USE teeest; select login\_user, password\_user, driver\_status from register FOR XML PATH(''USER''), ROOT(''USERS'') '

set @sqlCmd = 'bcp.exe "' + @sqlStr + '" queryout ' + @fileName + ' -w -T'

EXEC xp\_cmdshell @sqlCmd;

end;

Листинг 4.1 – Скрипт создания процедуры ExProdToXml

Для импорта данных в таблицу Users, из файла формата xml, была разработана процедура XmlToProduct. Составляющая файла представлена в листинге 4.2:

<USERS>

<USER><id\_user>500</id\_user><login\_user>Markiks</login\_user><password\_user>1337</password\_user><driver\_status>0</driver\_status></USER>

<USER><id\_user>501</id\_user><login\_user>Igorek</login\_user><password\_user>1337</password\_user><driver\_status>0</driver\_status></USER>

<USER><id\_user>502</id\_user><login\_user>Maximus</login\_user><password\_user>1337</password\_user><driver\_status>0</driver\_status></USER>

Продолжение листинга 4.2

<USER><id\_user>503</id\_user><login\_user>Mishanya</login\_user><password\_user>1337</password\_user><driver\_status>0</driver\_status></USER>

</USERS>

Листинг 4.2 − Данные XML для импорта в базу данных

Для получения xml файла и последующего разбора со вставкой используется конструкция FROM OPENROWSET совместно с параметром BULK. Процедура XmlToProduct представлена ниже в листинге 4.3.

ALTER procedure [dbo].[XmlToProduct]

as begin

DECLARE @xml XML;

SELECT @xml = CONVERT(xml, BulkColumn, 2) FROM OPENROWSET(BULK 'C:\Users\Lenovo\AppData\Local\Import.xml', SINGLE\_BLOB) AS x

INSERT INTO register(id\_user, login\_user, password\_user, driver\_status)

SELECT

t.x.query('id\_user').value('.', 'INT'),

t.x.query('login\_user').value('.', 'varchar(50)'),

t.x.query('password\_user').value('.', 'varchar(50)') ,

t.x.query('driver\_status').value('.', 'INT')

FROM @xml.nodes('//USERS/USER') t(x)

End;

Листинг 4.3 – Скрипт создания процедуры XmlToProduct

Таким образом мы можем быстро и удобно импортировать данные в нашу таблицу.

## **Описание и реализация технологии**

* 1. **Шифрование данных**

В разработке своего проекта я использовал технологию шифрования и маскирования базы данных

Несомненно, между маскированием и шифрованием много сходства. А оптимальной защиты данных можно добиться, только если использовать их совместно. Часто даже специалисты в области защиты данных считают шифрование разновидностью маскирования. Что ещё хуже, есть те, кто считает, что это одно и то же. Но, по сути, это два разных с технической точки зрения процесса.

Для шифрования большую роль играет обратимость процесса. Для маскирования обратимость –недостаток. Ни при каких условиях пользователь не должен видеть первоначальную информацию, которая была замаскирована. Если маскируете данные, процессы редактирования (фрагментирование данных, скрывание или удаление) важных элементов дата-сета необратимы.

Технология предоставлена на листинге 5.1:

create master key encryption by password = '123123';

CREATE CERTIFICATE Certificate\_test WITH SUBJECT = 'test';

GO

SELECT name CertName,

certificate\_id CertID,

pvt\_key\_encryption\_type\_desc EncryptType, issuer\_name Issuer

FROM sys.certificates;

CREATE SYMMETRIC KEY SymKey\_test WITH ALGORITHM = AES\_256 ENCRYPTION BY CERTIFICATE Certificate\_test;

SELECT name KeyName,

symmetric\_key\_id KeyID,

key\_length KeyLength,

algorithm\_desc KeyAlgorithm

FROM sys.symmetric\_keys;

ALTER TABLE register ADD Password\_encrypt varbinary(MAX)

OPEN SYMMETRIC KEY SymKey\_test DECRYPTION BY CERTIFICATE Certificate\_test;

UPDATE register

SET Password\_encrypt = EncryptByKey (Key\_GUID('SymKey\_test'), password\_user)FROM register;

GO

CLOSE SYMMETRIC KEY SymKey\_test;

Листинг 5.1 Шифрование базы данных

* 1. **Маскирование данных**

Динамическое маскирование данных (DDM) ограничивает возможность раскрытия конфиденциальных данных за счет маскирования этих данных для непривилегированных пользователей. Оно позволяет значительно упростить проектирование и написание кода для системы безопасности в приложении.

Динамическое маскирование данных помогает предотвратить несанкционированный доступ к конфиденциальным данным, позволяя клиентам задать объем раскрываемых конфиденциальных данных с минимальным влиянием на уровень приложения. DDM можно настроить для отдельных полей базы данных, чтобы скрыть конфиденциальные данные в результирующих наборах запросов. При использовании DDM данные в базе данных не изменяются. DDM легко использовать с существующими приложениями, так как правила маскирования применяются к результатам запроса. Многие приложения могут маскировать конфиденциальные данные без изменения существующих запросов.

Пример маскирования представлен в листинге 5.2

GRANT SELECT ON Drivers TO [MaskingTestUser];

select \* from Drivers

Alter table Drivers

Alter column password\_driver nvarchar(50) MASKED WITH (FUNCTION='default()')

execute as user = 'MaskingTestUser';

select \* from Drivers;

revert;

GRANT UNMASK TO MaskingTestUser;

GO

EXECUTE AS USER = 'MaskingTestUser';

SELECT \* FROM Drivers;

REVERT;

GO

REVOKE UNMASK TO MaskingTestUser;

EXECUTE AS USER = 'MaskingTestUser';

SELECT \* FROM Drivers;

REVERT;

EXECUTE AS USER = 'MaskingTestUser';

SELECT \* FROM Drivers;

REVERT;

GO

REVERT;

SELECT \* FROM Drivers;

Листинг 5.2 Маскирование данных

## **Тестирование производительности**

К таблице Register было разработано несколько процедур для заполнения таблицы строками в количестве 100000.

Например, одной из таких процедур была процедура InsertThousands, представленная ниже в листинге 6.1:

GO

create or ALTER procedure InsertThousands as

begin

declare @i int;

set @i = 1;

while @i <= 100000

begin

insert into register(login\_user, password\_user, driver\_status )

values ('test', cast(FLOOR(1000000\*RAND()) as nvarchar(50)), 0);

set @i = @i + 1;

end;

end;

Листинг 6.1 – Описание процедуры InsertThousands

## **Руководство пользователя**

## **7.1** **Установка программного средства**

Для работы программного средства необходима первоначальная настройка. Для начала, необходимо наличие установленной на компьютере си-стемы управления базами данных Microsoft SQL Server, а также утилиты SQL Server Management Studio для управления и администрирования компонентов Microsoft SQL Server.

Необходимо войти в SQL Server Management Studio, где в обозревателе объектов раскрыть список текущего подключения и нажать правой кнопкой мыши на вкладке «Базы данных». В контекстном меню требуется выбрать пункт «Присоединить», как на рисунке 7.1.



Рисунок 7.1 – Выбор пункта присоединить

В открывшемся окне на панели «Базы данных для присоединения» необ-ходимо нажать кнопку «Добавить» и указать физический путь к базе данных проекта teeest. После этого нажать на кнопку «ОК».

После всех проделанных шагов в раскрывающемся списке «Базы дан-ных» обозревателя объектов должна появиться база данных teeest, как на рисунке 7.2.

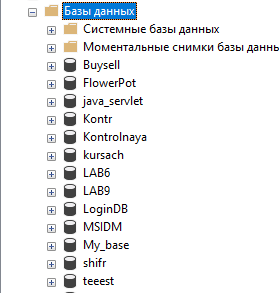


Рисунок 7.2 - Появление базы данных teeest

Далее необходимо запустить приложение через .exe файл. На рисунке 7.3 представлен установщик.



Рисунок 7.3 – Ярлык приложения Taxi

## **7.2 Регистрация и авторизация**

После запуска приложения пользователь видит главное окно приложения, на котором он может авторизоваться и получить возможность оформить заказ. Окно представлено на рисунке 7.4

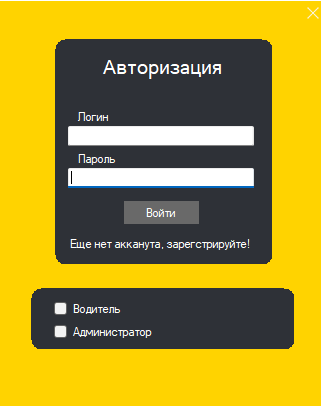


Рисунок 7.4 – Окно входа в аккаунт

Если пользователь ещё не имеет своего аккаунта, то по нажатию на соответствующую кнопку можно перейти к окну регистрации. Пользователь должен придумать логин и пароль, которые будет использовать для авторизации в приложении. Окно регистрации показано на рисунке 7.5.

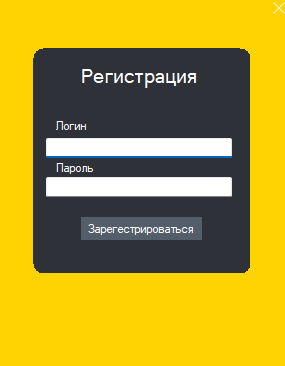


Рисунок 7.5 – Окно регистрации

## **Руководство пользователя по работе с программным средством**

После успешной авторизации пользователь попадает на главную страницу приложения где может посмотреть информацию о своем профиле и при надобности изменить ее, пример показан на рисунке 7.6.

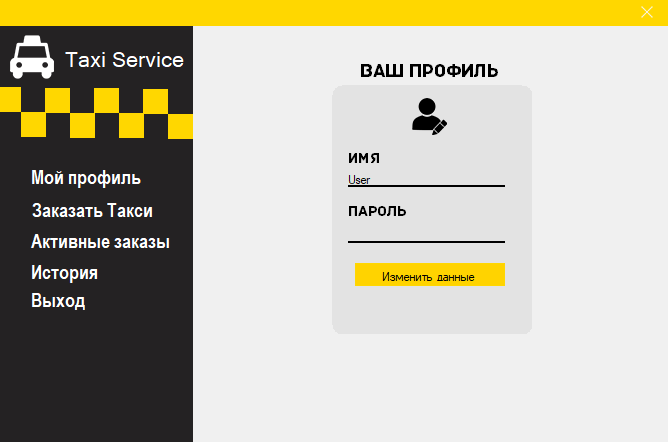


Рисунок 7.6 – Главная страница приложения

Далее пользователь может сделать заказ в поле «Заказать Такси», правильное заполнение формы заказа показано на рисунке 7.7.

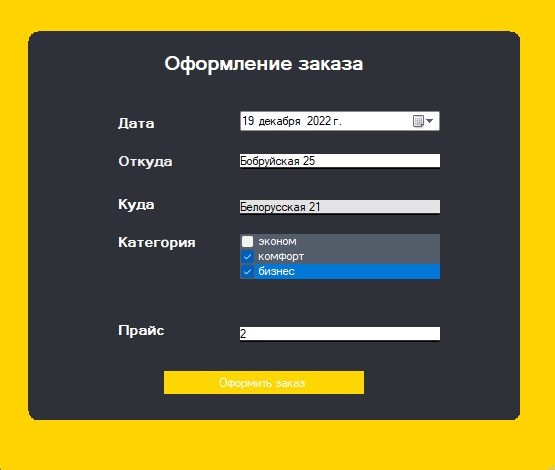


Рисунок 7.7 – Форма правильного заполнения заказа

В разделе Активные заказы, пользователь может посмотреть свои заказы которые находятся в статусе «Активные», пример показан на рисунке 7.8.

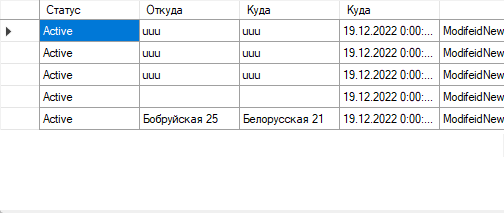


Рисунок 7.8 Активные заказы пользователя

Главный профиль водителя схож с пользовательским, отличием являются поля «Список заказов» где можно посмотреть активные заказы, пример показан на рисунке 7.9

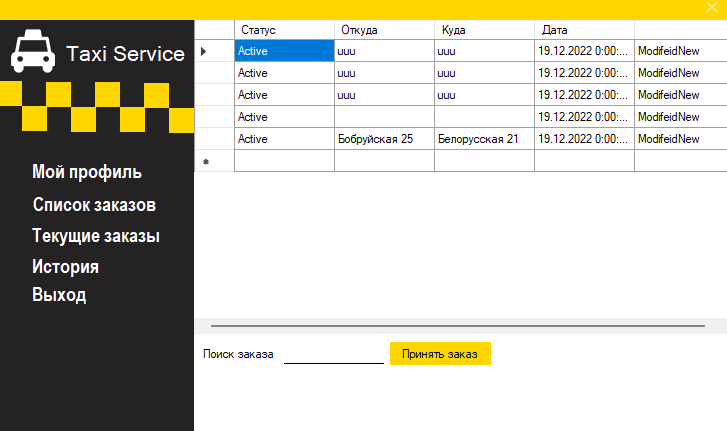


Рисунок 7.9 Список заказов водителя

Также есть еще новое поле «Текущие заказы», где водитель может либо завершить поездку, либо отменить ее, пример показан на рисунке 7.10

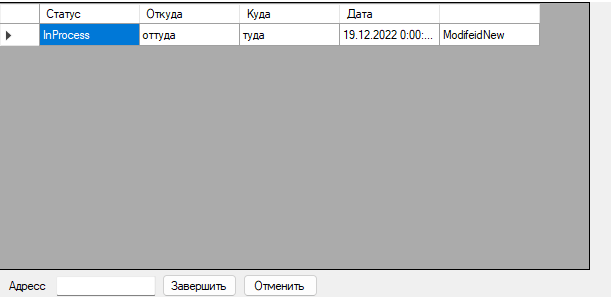


Рисунок 7.10 Текущие заказы водителя

## **Заключение**

В ходе выполнения курсовой работы был разработан проект, являющийся помощником инновационным решением на рынке по заказу такси, предоставляющий выбрать время и день заказа такси. Во время выполнения данной курсовой работы было изучено немало теоретического материала, а также просмотрено и разобрано большое количество уже готовых решений тех или иных задач. Были разработаны, описаны и применены на практике процедуры экспорта и импорта данных; было описано руководство пользователя для приложения.

Функционально были выполнены следующие задачи:

* + - администратор добавляет или удаляет пользователей;
    - администратор добавляет или удаляет водителей;
    - администратор добавляет или удаляет заказы;
    - администратор просматривает и изменяет личные данные пользователей;
    - администратор просматривает и изменяет личные данные водителей;
    - администратор просматривает и изменяет данные заказов;
    - клиент может оформить заказ;
    - клиент может посмотреть историю заказов;
    - водитель может принять заказ;
    - водитель может посмотреть историю заказов;

Приложение разработано на платформе .NET Framework с использованием технологий WinForms и ADO, реализацией технологии шифрование и маскирование БД, а также объектно-ориентированного языка программирования C#.

Данное программное средство имеет удобный и понятный пользователю интерфейс.

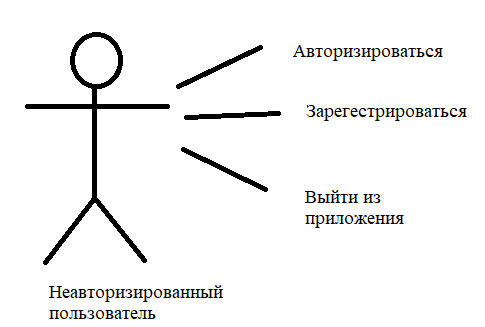
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

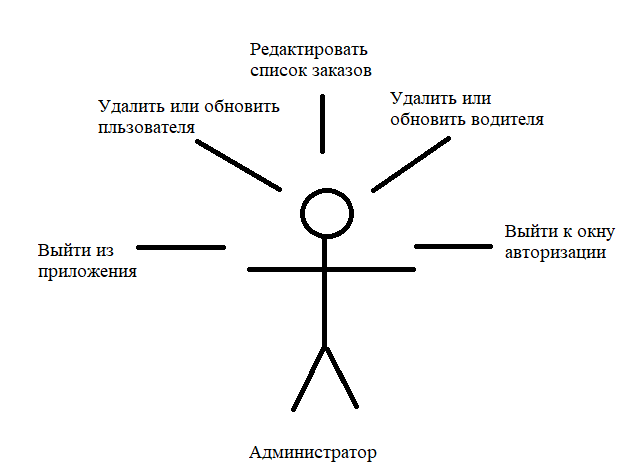
Естественно, существует ряд приложений, имеющих функционал и дизайн в разы лучше. Подводя итог всей курсовой работе, можно сделать выводы, что в разработке приложений мне предстоит еще долгий путь.

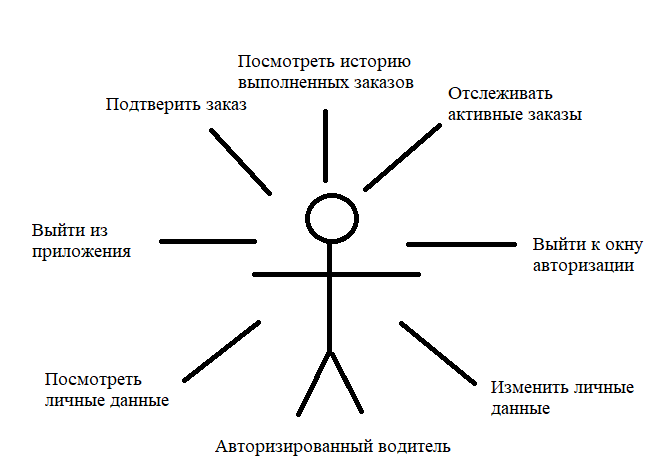
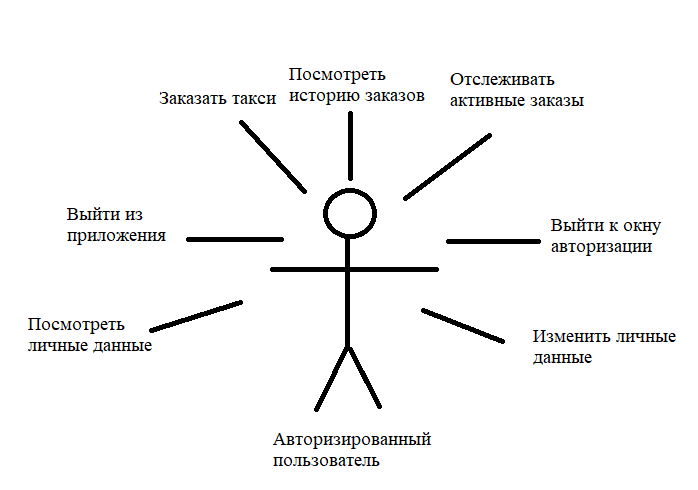
## **Список используемых источников**

1. Garofallo, R. Building Enterprise Applications with Windows Presentation Foundation and the Model View ViewModel Pattern / R. Garofallo, California: O’Reilly Media, 2011, 220 p.
2. Microsoft Справочник по Transact-SQL [Электронный ресурс]/ docs.microsoft.com– Режим доступа: https://docs.microsoft.com/ru-ru/tsql/language-reference?view=sql-server-ver15 – Дата доступа 02.12.2022
3. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
4. Интернет-портал [Электронный ресурс] / [https://docs.microsoft.com](https://vk.com/away.php?to=https%3A%2F%2Fdocs.microsoft.com&cc_key=) – SQL Server Documentation: [https://docs.microsoft.com/en-us/sql/sql-server/sql-s..](https://vk.com/away.php?to=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fsql%2Fsql-server%2Fsql-server-technical-documentation%3Fview%3Dsql-server-2017&cc_key=) – Дата доступа: 27.10.2022
5. Информационный сайт: базы данных [Электронный ресурс]. Режим доступа:inform.sch117nn.edusite.ru/DswMedia/klyaks\_netuchitelyutestyibazyidannyix.htm – Дата доступа: 24.05.2022.
6. Информационный сайт: Организация хранения файлов в базе данных Microsoft SQL Server [Электронный ресурс]. –Режим доступа: <https://streletzcoder.ru/organizatsiya-hraneniya-faylov-v-baze-dannyih-microsoft-sql-server-universalnyiy-sposob>. Дата доступа: 24.11.2022.
7. Пацей, Н.В. Курс лекций по языку программирования C# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
8. Сайт о программировании Metanit: Введение в ADO.NET [Электронный ресурс]– Режим доступа: https://metanit.com/sharp/adonet/1.1.php – Дата доступа: 24.05.2022.

# **Приложение А**







## **Приложение Б**

use master;

create database teeest;

use teeest;

create table register(

id\_user int identity(1, 1) not null primary key,

login\_user varchar(50) not null,

password\_user varchar(50) not null,

driver\_status int

create table Orders(

OrderID int identity(1,1) not null primary key,

Adress1 nvarchar(100),

Adress2 nvarchar(100),

Category nvarchar(20),

Price int,

Date date,

Status nvarchar(50),

id\_user int not null foreign key references register(id\_user),

id\_driver int foreign key references Drivers(id\_driver))

create table Administrators(

admin\_id int identity(1,1) not null,

admin\_login nvarchar(50),

admin\_password nvarchar(50)

);

create table Drivers(

id\_driver int identity(1, 1) primary key not null,

login\_driver varchar(50) not null,

password\_driver varchar(50) not null

);

Create table History (

HistoryID int identity (1,1) not null primary key,

Adress1 nvarchar(100),

Adress2 nvarchar(100),

Status nvarchar(50),

Date date,

id\_driver int,

id\_user int

);

create table HistoryUpdate(

Date\_of\_Change datetime,

id\_driver int foreign key references Drivers(id\_driver),

login\_driver nvarchar(50),

old\_login nvarchar(50)

);

## **Приложение В**

|  |  |  |
| --- | --- | --- |
| **Название процедуры** | **Описание** | **Принимаемые параметры** |
| ActiveOrder | Посмотреть активные заказы | id\_user |
| AddDriver | Добавить водителя в таблицу водителей | login\_driver, password\_driver |
| AddDriverId | Добавить Id водителя | id\_driver |
| AddHistory | Добавление заказа в таблицу история | Adress1, Adress2, Status, Date, id\_driver, id\_user |
| AddOrder | Добавить заказ в таблицу заказов | Adress1, Adress2, Category, Price,Date, id\_user, Status |
| AddUser | Создать пользователя и добавить в таблицу пользователей | login\_user, password\_user, driver\_status |
| CancelledOrder | Установить статус заказа «Отменен» | Status |
| CheckUser | Проверить, существует ли пользователь | login\_user, password\_user |
| CreateDriver | Создать водителя и добавить в таблицу водителей | login\_driver, password\_driver |
| CreateOrder | Создать заказ и добавить в таблицу заказов | Adress1, Adress2, Category, Price,Date, id\_user, id\_driver, Status |
| DeleteDriver | Удалить водителя из таблицы водителей | id\_driver |
| DeleteOrder | Удалить заказ из таблицы заказов | OrderID |
| DeleteUser | Удалить пользователя из таблицы пользователей | Id\_user |
| DonedOrder | Установить статус заказа «Выполнен» | Status |
| ExProdToXml | Процедура экспорта в XML |  |
| HaveAdmins | Проверить, существует ли администратор | admin\_login, admin\_password |
| HaveDrivers | Проверить, существует ли водитель | login\_driver, password\_driver |
| InsertThousands | Процедура заполнения таблицы на 100000 строк |  |
| OutAllCurrentOrders | Вывести заказ со статусом «В процессе» | Status |
| OutAllOrders | Вывести все заказы со статусом «Активен» | Status |
| OutHistoryDrivers | Вывести историю водителя | id\_driver |
| OutHistoryUsers | Вывести историю пользователя | id\_user |
| OutOrders | Вывести заказы со статусом «Активен» конкретного пользователя | Id\_user |
| Refresh | Обновить данные |  |
| RefreshDriver | Обновить данные водителя | id\_driver |
| RefreshOrders | Обновить данные заказов | OrderID |
| ReturnDriverId | Получения ID водителя по имени | driver\_login |
| ReturnId | Получения ID пользователя по имени | user\_login |
| SearchDriver | Получение данных водителя | id\_driver, login\_driver |
| SearchOrder | Поиск определенного заказа | Status, Adress1, Adress2, Date |
| SearchOrderForAdmin | Поиск определенного заказа в панели администратора | Adress1, Adress2, Date, Status, Category, Price, id\_driver, id\_user |
| SearchUser | Получение данных пользователей | id\_user, login\_user |
| UpdateDriver | Изменить данные водителя | login\_driver, password\_driver |
| UpdateOrderInProcess | Изменить статус заказа на «В Процессе» | Status |
| UpdateOrders | Изменить данные заказа | Adress1, Adress2, Date, Status, Category, Price, id\_driver, id\_user |
| UpdateUser | Изменить данные водителя | login\_user, password\_user |
| XmlToProudct | Импорт данных в XML |  |