Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический
университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по практике

Москва, 2020 г.

Оглавление

Введение

В рамках производстенной практики необходимо было реализовать базу данных на MS SQL со связкой с C Sharp.

Сфера деятельности компании

Компания «Цезарь Сателлит» - ведущий оператор систем безопасности для автомобилей и недвижимости.

Группа компаний «Цезарь Сателлит», созданная в 2000 году, является основателем и ключевым поставщиком рынка телематических услуг и комплексной безопасности на территории России. Более 200 000 частных клиентов и собственников бизнеса доверили компании самое дорогое: личную безопасность, защиту автомобиля, дома и офиса.

Использование новейших спутниковых (ГЛОНАСС/GPS) и мобильных (GSM) технологий, собственные патентованные разработки, развитая мониторинговая инфраструктура, уникальные технологии розыска и тесное взаимодействие с полицией – все это позволяет компании идти на опережение и противостоять современным методам угона.

Клиентами компании являются лидеры автомобильной промышленности, такие как BMW Russland Trading, Toyota Motor, Jaguar Land Rover, Mazda Motor Rus, Ford Sollers и крупнейшие автодилеры по всей стране. Под охраной «Цезарь Сателлит» находится значительная доля банковского сектора (Сбербанк, Райффайзенбанк, Ситибанк, Абсолют банк, Росбанк, БинБанк), ключевые сети розничной торговли (X5 Retail Group, «Азбука вкуса», «Магнит», «Дикси»).

Ежедневно «Цезарь Сателлит» предотвращает десятки случаев краж и автомобильных угонов по всей территории России, в странах Европы, Азии и СНГ, обеспечивая сохранность жизни и имущества своих клиентов[1].

Основная часть

Для реализации поставленной задачи была выбрана система управления реляционными базами данных, разработанная копорацие Microsoft, Microsoft SQL Server 2017. Выбор MS SQL 2017 был обусловлен стабильностью работы и совместимостью с языком программирования C Sharp. База данных, написанная на MS SQL, может подключится к C Sharp по прямому подключению, а также по подключению через фреймворк Entity.

Python

Язык программирования Python использовался для генерации случайных данных для базы данных. Так как заполнение базы данных и придумывание осмысленных данных для ее атрибутов слишком трудозатратная задача, поэтому была выбрана библиотека генерации случайных данных - Faker. С ее помощью было сгенерировано более десяти тысяч строк данных, сравнимых с данными.

Faker - это библиотека Python, которая генерирует поддельные данные. Независимо от того, нужно ли нам загрузить свою базу данных, создать красивые XML-документы, заполнить наш тестотовый сервис, чтобы протестировать его, или анонимизировать данные, взятые из производственной службы.

Листинг 1: Создание фейковых имен

```
from faker import Faker

fake = Faker()

for _ in range(3):
    print(fake.name())

# 'Elda Palumbo'
# 'Sig. Alighieri Monti'
# 'Costanzo Costa'
```

Листинг 2: Генерация фейковых адресов

```
from faker import Faker

fake = Faker()

for _ in range(3):
    print(fake.address())

# Michael Pike Dannyton, AR 20235
# Little Wall Apt. 359 East Loretta, NV 16913
# Steve Park East Austin, MI 06826
```

Листинг 3: Скрипт для генерации контрактов

```
from faker import Faker
import random

myFaker = Faker('en_US')

f = open('/Users/antontimonin/Desktop/Practice/data/Contract1.txt', 'w')

diaposon = 1001

card_ids = []
person_ids = []
service_ids = []

def formateDate(year):
  if year >= 0 and year < 10:
    return "0" + str(year)
  else:
    return str(year)

for i in range(diaposon):
  f.write(str(i+1) + ';') #contract id
  while (1):
    number = myFaker.pyint(min_value=1, max_value=350, step=1)
    if (number not in card_ids or len(card_ids) >= 350):
      card_ids.append(number)
      f.write(str(number) + ';') #car id
      break

  while (1):
    number = myFaker.pyint(min_value=1, max_value=250, step=1)
    if (number not in person_ids or len(person_ids) >= 250):
      person_ids.append(number)
      f.write(str(number) + ';') #person id
      break

  while (1):
    number = myFaker.pyint(min_value=1, max_value=15, step=1)
    if (number not in service_ids or len(service_ids) >= 15):
      service_ids.append(number)
      f.write(str(number) + ';') #service id
      break

  day1 = myFaker.pyint(min_value=1, max_value=28, step=1)
  month1 = myFaker.pyint(min_value=1, max_value=12, step=1)
  year1 = myFaker.pyint(min_value=0, max_value=20, step=1)

  day2 = myFaker.pyint(min_value=1, max_value=28, step=1)
  month2 = myFaker.pyint(min_value=1, max_value=12, step=1)
  year2 = myFaker.pyint(min_value=0, max_value=20, step=1)

  if year1 < year2:
    day1, day2 = day2, day1
```

```
53      month1, month2 = month2, month1
54      year1, year2 = year2, year1
55    elif year1 == year2 :
56      if month1 < month2:
57        day1, day2 = day2, day1
58        month1, month2 = month2, month1
59        year1, year2 = year2, year1
60      elif month1 == month2:
61        if day1 <= day2:
62          day1, day2 = day2, day1
63          month1, month2 = month2, month1
64          year1, year2 = year2, year1
65
66    date2 = str(day1) + "." + formateDate(month1) + "." + formateDate(year1)
67    date1 = str(day2) + "." + formateDate(month2) + "." + formateDate(year2)
68
69    f.write(date1 + ';') #date start id
70    f.write(date2) #date end id
71    f.write('\n')
72
73  f.close()
```

MS SQL

На языке SQL было создано две базы данных, предназначенных для оперативного взаимодействия всех компонентов компании. Главной таблицой в базе данных CS1 является таблица Contract. Эта таблица связывает все таблицы в двух базах данных. Через эту таблицу можно получить информацию о машинах оперделенного пользователя, какие сообщение приходили на оборудование, устоновленные в этой машине. Также можно определить в каком сервисном центре клиент устанавливал оборудование и когда это оборудование последний раз проверялось.

База данных, представленная на рисунке 1 спроектирована и приведена третьей нормальной форме.

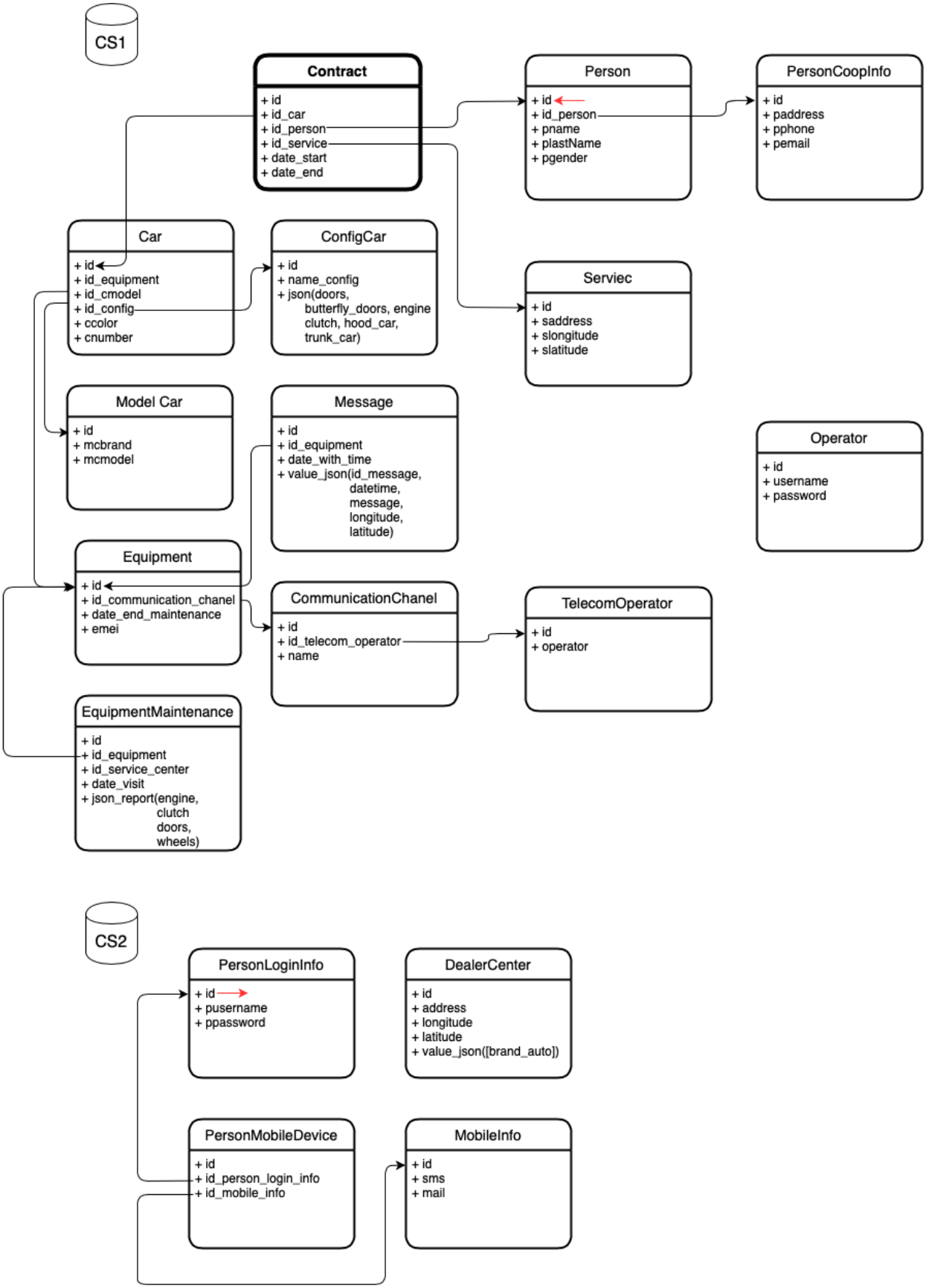

Рис. 1: Спроектированная база данных

База данных CS1 состоит из следующих таблиц:

- **Contract**- таблица с контрактами пользователей;
- **Person** - таблица с именем и фамилией клиента;
- **PersonCoopInfo** - таблица с контактами, по которым можно связать с клиентами;
- **Serviec** - таблица с адресами сервисов, устанавливающих оборудование компании клиентам;
- **Operator** - таблица с логинами и паролями операторов, мониторящих события, происходящие с имуществом клиентов;
- **Car** - таблица с номером и цветом машины клиента;
- **ConfigCar** - таблица с конфигурацией оборудования, поставленного на конкретную машину;
- **ModelCar** - таблица с брендом и моделью машины клиента;
- **Equipment** - таблица с информаций об оборудование;
- **EquipmentMaintenance** - таблица с информацией о последнем техническом осмотре оборудования;
- **Message** - таблица с сообщениями, пришедшими на конкретное оборудование;
- **CommunicationChanel** - таблица с каналом коммуникации, установленным в оборудование;
- **TelecomOperator** - таблица с операторами связи.

База данных CS2 состоит из следующих таблиц:

- **PersonLoginInfo**- таблица с логинами и паролями пользователей, для входа в систему для мониторинга состояния их имущества;
- **PersonMobileDevice**- таблица информацией о включенных функциях на мобильных устройствах;
- **MobileInfo**- таблица с информацией о включенных методах оповещения;
- **DealerCenter**- таблица с информацией о местоположении дилерских центров.

На листингах 4-9 приведены запросы к базам данных CS1 и CS2.

Листинг 4: Функция поиска информации по автомобилям по пользовательскому id

```
CREATE FUNCTION GetConcreteCarByID (@id_person int)
RETURNS TABLE
AS
RETURN
(
  SELECT c.id, p.pname, p.plastname, pinf.paddress, pinf.pemail, pinf.pphone, mc.mcbrand, mc.
      mcmodel, cr.cnumber
  FROM Contracts as c
  JOIN Person AS p ON p.id = c.id_person
  JOIN PersonInfo AS pinf ON pinf.id = p.id_personinfo
  JOIN Car AS cr ON cr.id = c.id_car
  JOIN ModelCar as mc ON mc.id = cr.id_cmodel
  WHERE c.id = @id_person
);

SELECT * FROM GetConcreteCarByID(15);

DROP FUNCTION GetConcreteCarByID;
```

Листинг 5: Хранимая процедура выводящая полную информацию по id контракта

```
CREATE PROCEDURE AddAllInfo(@id_contract int) AS
BEGIN
  DROP TABLE IF EXISTS dbo.TmpFullInfo

create TABLE CS.dbo.TmpFullInfo(
  id int NOT NULL,
  person_name VARCHAR(100) NOT NULL,
  person_lastname VARCHAR(100) NOT NULL,
  person_address VARCHAR(100) NOT NULL,
  person_email VARCHAR(100) NOT NULL,
  person_phone VARCHAR(100) NOT NULL,
  service_address VARCHAR(100) NOT NULL,
  car_brand VARCHAR(100) NOT NULL,
  car_model VARCHAR(100) NOT NULL,
  car_number VARCHAR(100) NOT NULL
)

  insert into dbo.TmpFullInfo
  SELECT c.id, p.pname, p.plastname, pinf.paddress, pinf.pemail,
        pinf.pphone, s.saddress, mc.mcbrand, mc.mcmodel, cr.cnumber FROM Contracts as c
  JOIN Serviec AS s ON s.id = c.id_service
  JOIN Person AS p ON p.id = c.id_person
  JOIN PersonInfo as pinf ON pinf.id = p.id_personinfo
  JOIN Car AS cr ON cr.id = c.id_car
  JOIN ModelCar as mc ON mc.id = cr.id_cmodel
  WHERE c.id = @id_contract
END;

exec AddAllInfo 3;

```

```
31 select * from TmpFullInfo;
32
33 drop procedure AddAllInfo;
34
35 select * from TmpFullInfo;
```

Листинг 6: Функция для вывода полной информации по пользователям

```
1  DROP FUNCTION GetFullInfoAboutPerson;
2
3  CREATE FUNCTION GetFullInfoAboutPerson()
4  RETURNS TABLE
5  AS
6  RETURN
7  (
8    SELECT p.id, p.pname, p.plastname, pinf.pemail, plog.pusername, plog.ppassword FROM CS.dbo.
       Person as p
9    JOIN CS.dbo.PersonInfo as pinf ON pinf.id = p.id_personinfo
10   JOIN CS2.dbo.PersonLoginInfo AS plog ON plog.id = p.id
11 );
12
13 SELECT * FROM GetFullInfoAboutPerson();
```

Листинг 7: Процедура для формирования нового контракта с проставлением всех зависимостей

```
1  CREATE PROCEDURE FormContract(@person_name VARCHAR(100),
2                                @person_lastname VARCHAR(100),
3                               @person_gender VARCHAR(100),
4                                @person_phone VARCHAR(100),
5                                @car_brand VARCHAR(100),
6                                @car_model VARCHAR(100),
7                                @car_color VARCHAR(100),
8                                @car_number VARCHAR(100),
9                                @equipment_emei VARCHAR(100),
10                               @operator VARCHAR(100)) AS
11 DECLARE
12   -- Contract
13   @ID_CONTRACT INT,
14   @DATE_START DATE,
15   @DATE_END DATE,
16
17   -- Person
18   @ID_PERSON INT,
19   @ID_PERSON_COOP_INFO INT,
20
21   -- Car
22   @ID_CAR INT,
23   @ID_EQUIPMENT INT,
24   @ID_MODELCAR INT,
25   @ID_CONFIG INT,
26
27   -- Communication Chanel
```

```
28    @ID_COMMUNICATION_CHANEL INT
29
30  BEGIN
31
32    SELECT @DATE_START = CONVERT(DATE, GETDATE());
33    SELECT @DATE_END = CONVERT(DATE, DATEADD(year, 8, GETDATE()));
34
35    SELECT @ID_COMMUNICATION_CHANEL = 1;
36    SELECT @ID_CONFIG = 1;
37
38    SELECT @ID_MODELCAR = mc.id FROM CS.dbo.ModelCar as mc
39    WHERE mc.mcbrand = @car_brand and mc.mcmodel = @car_model
40
41    IF NOT EXISTS (SELECT mc.id FROM CS.dbo.ModelCar as mc WHERE mc.id = @ID_MODELCAR)
42    BEGIN
43      INSERT INTO CS.dbo.ModelCar(mcbrand, mcmodel)
44      VALUES (@car_brand, @car_model);
45      SET @ID_MODELCAR = (SELECT SCOPE_IDENTITY());
46    END;
47
48    EXEC GetConfigID @car_brand;
49    SELECT @ID_CONFIG = tt.num_id_operator FROM CS.dbo.TmpTable as tt;
50
51    -- Add in Equipment table
52    INSERT INTO CS.dbo.Equipment(id_communication_chanel, date_end_maintenance, emei)
53    VALUES (@ID_COMMUNICATION_CHANEL, @DATE_END, @equipment_emei);
54    SET @ID_EQUIPMENT = (SELECT SCOPE_IDENTITY());
55
56    -- Add in Car table
57    INSERT INTO CS.dbo.Car(id_equipment, id_cmodel, id_config, ccolor, cnumber)
58    VALUES (@ID_EQUIPMENT, @ID_MODELCAR, @ID_CONFIG, @car_color, @car_number);
59    SET @ID_CAR = (SELECT SCOPE_IDENTITY());
60
61    -- Add in PersonInfo table
62    INSERT INTO CS.dbo.PersonInfo(paddress, pphone, pemail)
63    VALUES ('', @person_phone, '');
64    SET @ID_PERSON_COOP_INFO = (SELECT SCOPE_IDENTITY());
65
66    -- Add in Person table
67    INSERT INTO CS.dbo.Person(id_personinfo, pname, plastname, pgender)
68    VALUES (@ID_PERSON_COOP_INFO, @person_name, @person_lastname, @person_gender);
69    SET @ID_PERSON = (SELECT SCOPE_IDENTITY());
70
71    -- Add in Contracts table
72    INSERT INTO CS.dbo.Contracts(id_car, id_person, id_service, date_start, date_end)
73    VALUES (@ID_CAR, @ID_PERSON, 0, @DATE_START, @DATE_END);
74    SET @ID_CONTRACT = (SELECT SCOPE_IDENTITY());
75    END;
76
77  exec FormContract 'Bar', 'Baz', 'Man', 'email@mail.ru', 'BMW', 'X1', 'Bluebi', '
        TT777T777', 'JKWNNGKWEJNGKWE', 'Yota';
78
79  drop procedure FormContract;
```

Листинг 8: Процедура для удаления контрака с удалением всех необходимых зависимостей для вывода полной информации по пользователям

```
CREATE PROCEDURE DeleteContractWithID(@ID INT) AS
DECLARE
  @ID_PERSON INT,
  @ID_PERSON_COOP_INFO INT,
  @ID_CAR INT,
  @ID_EQUIPMENT INT;
BEGIN

  SELECT @ID_PERSON = c.id_person FROM CS.dbo.Contracts as c
  WHERE c.id = @ID;

  SELECT @ID_PERSON_COOP_INFO = p.id_personinfo FROM CS.dbo.Person as p
  WHERE p.id = @ID_PERSON;

  SELECT @ID_CAR = c.id_car FROM CS.dbo.Contracts as c
  WHERE c.id = @ID;

  SELECT @ID_EQUIPMENT = cr.id_equipment FROM CS.dbo.Car as cr
  WHERE cr.id = @ID_CAR;

  -- DELETE CONTRACT
  DELETE FROM CS.dbo.Contracts
  WHERE CS.dbo.Contracts.id = @ID;

  -- DELETE PERSON
  IF EXISTS (SELECT @ID_PERSON) DELETE FROM CS.dbo.Person
  WHERE CS.dbo.Person.id = @ID_PERSON;

  -- DELETE PERSON COOP INFO
  IF EXISTS (SELECT @ID_PERSON_COOP_INFO) DELETE FROM CS.dbo.PersonInfo
  WHERE CS.dbo.PersonInfo.id = @ID_PERSON_COOP_INFO;

  -- DELETE CAR
  IF EXISTS (SELECT @ID_CAR) DELETE FROM CS.dbo.Car
  WHERE CS.dbo.Car.id = @ID_CAR;

  -- DELETE EQUIPMENT
  IF EXISTS (SELECT @ID_EQUIPMENT) DELETE FROM CS.dbo.Equipment
  WHERE CS.dbo.Equipment.id = @ID_EQUIPMENT;

  -- DELETE MESSAGES
  IF EXISTS (SELECT @ID_EQUIPMENT) DELETE FROM CS.dbo.Message
  WHERE CS.dbo.Message.id_equipment = @ID_EQUIPMENT;

  SELECT @ID, @ID_PERSON, @ID_PERSON_COOP_INFO, @ID_CAR, @ID_EQUIPMENT;
-delete message with id_equipment id
END;

```

```
drop procedure DeleteContractWithID;

EXEC DeleteContractWithID 247;
```

Листинг 9: Процедура для добавления нового канала коммуникации

```
CREATE PROCEDURE AddNewCommunicationChanel(@name_communication VARCHAR(200)) AS
DECLARE
  @ID_COMMUNICATION_CHANEL INT,
  @ID_OPERATOR INT,
  @NAME_OPERATOR VARCHAR(200);
BEGIN

  SELECT @NAME_OPERATOR =
  CASE
    WHEN @name_communication = 'MTC' THEN 'mtc_tech'
    WHEN @name_communication = 'Megafon' THEN 'megafon_tech'
    WHEN @name_communication = 'Tele2' THEN 'tele2_tech'
    WHEN @name_communication = 'Bilain' THEN 'bilain_tech'
    WHEN @name_communication = 'Yota' THEN 'yota_tech'
    WHEN @name_communication = 'Tinkoff' THEN 'tinkoff_tech'
    WHEN @name_communication = 'Vineah' THEN 'vineah_tech'
    ELSE 'unknown_tech'
  END;

  SELECT @ID_OPERATOR = toper.id FROM CS.dbo.TelecomOperator as toper
  WHERE toper.operator = @NAME_OPERATOR;

  INSERT INTO CS.dbo.CommunicationChanel(id_telecom_operator, name)
  VALUES (@ID_OPERATOR, @name_communication);
  SET @ID_COMMUNICATION_CHANEL = (SELECT SCOPE_IDENTITY());

  IF (object_id('CS.dbo.TmpTable','U')) IS NOT NULL
  BEGIN
    DROP TABLE CS.dbo.TmpTable;
  END;

  CREATE TABLE CS.dbo.TmpTable(
    num_id_operator int NOT NULL
  );

  INSERT INTO CS.dbo.TmpTable (num_id_operator)
  VALUES (@ID_COMMUNICATION_CHANEL);
END;

EXEC AddNewCommunicationChanel 'Yota';

drop procedure AddNewCommunicationChanel;
```

C Sharp

C Sharp (произносится как "си шарп") — современный объектно-ориентированный и типобезопасный язык программирования. C Sharp относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

C Sharp является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Главная особенность таких компонентов в том, что они представляют собой модель программирования со свойствами, методами и событиями. У них есть атрибуты, предоставляющие декларативные сведения о компоненте. Они включают в себя собственную документацию. C Sharp предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C Sharp подходит для создания и применения программных компонентов[2].

На листинге 10 представлен класс, отвечающий за подключение к базе данных MS SQL Server.

Листинг 10: Подключение к базе данных

```
class CS_DB
{
  private SqlConnection connection = new SqlConnection("Data Source=4D97\\MSSSQLSERVER;
      Initial Catalog=CS;Integrated Security=True");

  public void openConnection()
  {
    if (connection.State == ConnectionState.Closed)
    {
      connection.Open();
    }
  }

  public void closeConnection()
  {
    if (connection.State == ConnectionState.Open)
    {
      connection.Close();
    }
  }

  public SqlConnection getConnection()
  {
    return connection;
  }
}
```

Windows Forms

Windows Forms позволяет разрабатывать интеллектуальные клиенты. Интеллектуальный клиент — это приложение с полнофункциональным графическим интерфейсом, простое в развертывании и обновлении, способное работать при наличии или отсутствии подключения к Интернету и использующее более безопасный доступ к ресурсам на локальном компьютере по сравнению с традиционными приложениями Windows [3].

Пример создание формы через Windows Forms API представлен на листинге 11. На рисунке 2, 3, 4 представлены форма авторизации оператора, главная форма с информацией о пользователе, форма с информацией о машинах конкретного пользователя соответственно.

Листинг 11: Пример создания формы в программе оператора

```
public partial class FormAuth : Form
{
  ControllerFormAuth controller;
  public FormAuth()
  {
    InitializeComponent();

    this.controller = new ControllerFormAuth();
  }

  private void loginButton_Click(object sender, EventArgs e)
  {

    if (controller.IsLogged(usernameTextBox.Text, passwordTextBox.Text))
    {
      this.Hide();
      Form1 nextWindow = new Form1(usernameTextBox.Text, passwordTextBox.Text);
      nextWindow.ShowDialog();
    }
    else
    {
      MessageBox.Show("This operator does not exist");
    }
  }
}
```

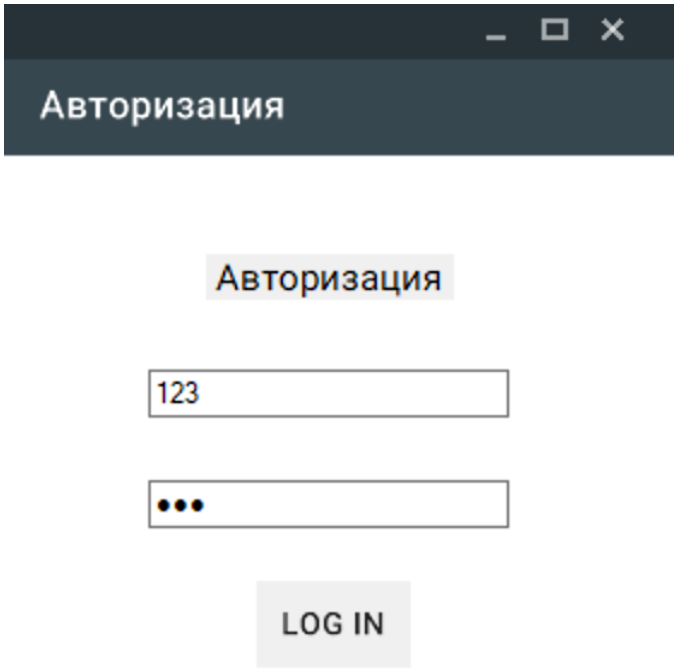

Рис. 2: Форма авторизации

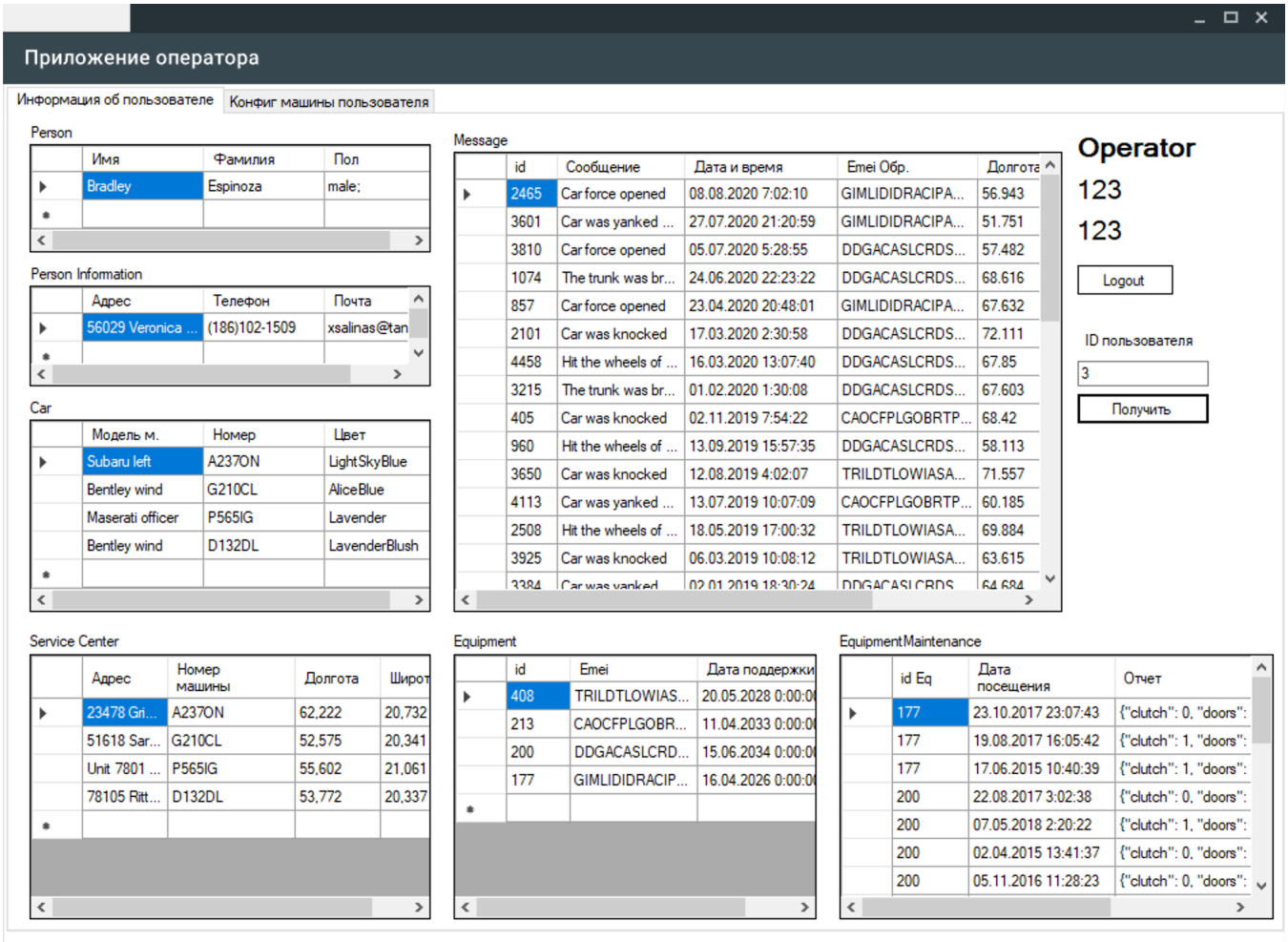

Рис. 3: Форма со всей информацие о пользователе

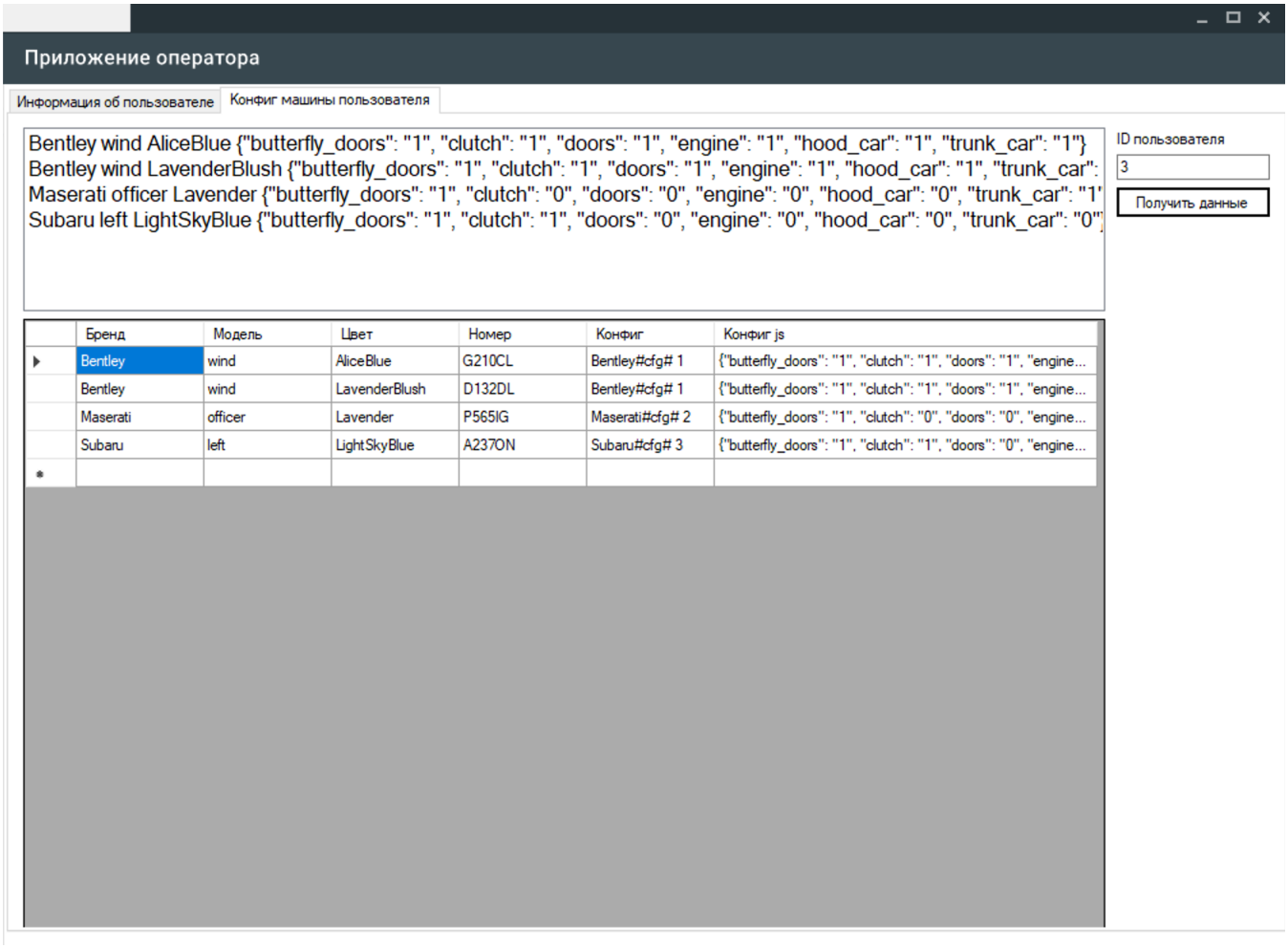

Рис. 4: Форма с информацией об машине пользователя

Entity Framework

Entity Framework — это набор технологий в ADO.NET, которые поддерживают разработку программных приложений, ориентированных на данные. Архитекторам и разработчикам приложений, ориентированных на обработку данных, приходится учитывать необходимость достижения двух совершенно различных целей. Они должны моделировать сущности, связи и логику решаемых бизнес-задач, а также работать с ядрами СУБД, используемыми для сохранения и получения данных. Данные могут распределяться по нескольким системам хранения данных, в каждой из которых применяются свои протоколы, но даже в приложениях, работающих с одной системой хранения данных, необходимо поддерживать баланс между требованиями системы хранения данных и требованиями написания эффективного и удобного для обслуживания кода приложения.

Платформа Entity Framework позволяет работать с данными в форме специфических для домена объектов и свойств (например, с клиентами и их адресами) без необходимости учитывать формат базовых таблиц и столбцов базы данных, где хранятся эти данные. Entity Framework дает разработчикам возможность работать с данными на более высоком уровне абстракции, создавать и сопровождать

приложения, ориентированные на работу с данными, одновременно с этим сокращая объем кода по сравнению с традиционными приложениями. Поскольку Entity Framework является компонентом .NET Framework, Entity Framework приложения могут работать на любом компьютере, на котором установлена .NET Framework с пакетом обновления 1 (SP1) версии 3,5. [4]

На практике по фреймворку Entity была дана теоретическа информация с последующим самостоятельным изучением.

Заключительная часть

Разработана база данных в связке с десктопным приложением, позволяющим авторизироваться операторам мониторинга и посмотреть всю информацию о клиенте. За время практики база данных приложение для операторов связи было переведено с обычного подключения базы данных к серверу MS SQL на подключение через строку подключения и фреймворк Entity.

Литература

[1] Цезарь Сателлит [Электронный ресурс]. - Режим доступа: https://www.csat.ru/about/ Дата обращения: 23.07.2020

[2] Краткий обзор языка C Шарп [Электронный ресурс]. - Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ Дата обращения: 23.07.2020

[3] Обзор Windows Forms [Электронный ресурс]. - Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/framework/winforms/windows-forms-overview Дата обращения: 24.07.2020

[4] Обзор Windows Forms [Электронный ресурс]. - Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/ef/overview Дата обращения: 21.08.2020