# Содержание

**Введение** . . . . . 4

**1 Характеристика организации и подразделения** . . . . . 5

1.1 Характеристика организации . . . . . 5

1.2 Характеристика подразделения . . . . . 5

**2 Аналитическая часть** . . . . . 6

2.1 Формализация задачи . . . . . 6

2.2 Формализация данных . . . . . 6

2.3 Формализация категорий пользователя . . . . . 7

2.4 Модели баз данных . . . . . 7

2.4.1 Дореляционные модели . . . . . 8

2.4.2 Реляционные модели . . . . . 8

2.4.3 Постреляционные модели . . . . . 8

2.5 Вывод . . . . . 9

**3 Технологическая часть** . . . . . 10

3.1 Средства реализации ПО . . . . . 10

3.2 Архитектура приложения . . . . . 11

3.3 Выбор СУБД . . . . . 11

3.4 Листинги кода . . . . . 13

3.5 Демонстрация работы веб-приложения . . . . . 18

3.6 Вывод . . . . . 24

**Заключение** . . . . . 25

**Список литературы** . . . . . 26

# Введение

В настоящие время огрмное количество людей тратит много времени и денег при покупке автомобиля.

Траты жителей нашей страны на покупку подержанных легковых автомобилей в последние 6 лет составляли 2 — 3 трлн рублей ежегодно, подсчитали эксперты аналитического агентства «АВТОСТАТ». Так, по данным агентства «АВТОСТАТ», с 2015-го по 2020 годы объем вторичного рынка вырос на 12% (с 4,9 млн до 5,5 млн единиц). При этом средневзвешенная цена автомобиля с пробегом за этот период увеличилась на 32%. Если в 2015 году ее показатель составлял почти 415 тыс. рублей, то в 2020-м — уже 548,7 тыс. рублей. Затраты на покупку подержанных машин все эти годы тоже росла – за 5 лет она увеличилась почти в 1,5 раза (с 1,99 трлн до 2,94 трлн рублей) [1].

Учитывая все вышеуказанное, можно сделать вывод о том, что интерес к автомобилям огромный. Создание интернет-магазина автомобилей поможет пользователям организовать и структурировать процесс поиска транспортного средства.

**Целью** практики является разработка веб-приложения для интернет-магазина автомобилей.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- формализовать задачу и определить необходимый функционал;
- рассмотреть модели баз данных и выбрать подходящую;
- проанализировать существующие СУБД и выбрать нужную;
- спроектировать и разработать БД;
- спроектировать и разработать WEB-приложение.

# 1 Характеристика организации и подразделения

## 1.1 Характеристика организации

Направления деятельности АО «НИИ ТП» включает в себя разработку и изготовление следующих основных видов изделий, а также проведения их полномасштабных испытаний на воздействие различных факторов окружающей среды [2].

- Комплексы автоматизированного управления космическими аппаратами.
- Радиотехнические системы взаимных измерений (СВИ) для поиска, сближения и стыковки космических аппаратов.
- Комплексы приема, обработки, распределения и доведения до потребителей информации дистанционного зондирования Земли.
- Радиолокационные системы наблюдения Земли самолетного и космического базирования.
- Радиотехнические комплексы для низкоорбитальных спутниковых систем связи, обеспечивающие глобальность передачи данных.
- Системы сбора, хранения и высокоскоростной передачи информации.
- Серийное изготовление геопространственной продукции.

## 1.2 Характеристика подразделения

Отдел АО «НИИ ТП», в котором я проходил практику, занимается проектированием систем защиты на изделия, созданием моделей угроз, а также установкой и настройкой систем защиты.

# 2 Аналитическая часть

## 2.1 Формализация задачи

В ходе выполнения практики необходимо спроектировать и разработать веб-приложение про автомобили, которое будет предоставлять интерфейс для взаимодействия с базой данных с возможностью добавлять каждому отдельному пользователю понравившиеся автомобили в корзину. Также необходимо разработать базу данных для хранения информации об автомобилях, их категориях, а также иинформации о пользователях.

Требуется реализовать функциональность для разных категорий пользователей, каждый из которых получит свой определённый набор прав.

## 2.2 Формализация данных

Разрабатываемая база данных должна содержать информацию о об автомобилях, их категориях, а также иинформацию о пользователях.

Таблица 2.1 – Классификация сведений о данных

| **Категория** | **Информация** |
|---|---|
| Автомобиль | Название бренда, название модели, цена, принадлежность к виду автомобилей |
| Категория | Название, описание |
| Заказ | Имя, фамилия, адрес доставки, номер телефона, email |
| Пользователь | Логин, пароль, категория пользователя |

## 2.3 Формализация категорий пользователя

Для взаимодействия с веб-приложением было выделено две категории пользователя:

- гость;
- авторизованный пользователь.

Гость (неавторизованный пользователь) сможет просматривать базовую информацию об автомобилях.

Для получения дополнительного функционала пользователю необходимо авторизоваться. Зарегистрированный пользователь в качестве дополнительного функционала получит возможность добавлять понравившиеся авто в корзину, а также оформить их заказ.

## 2.4 Модели баз данных

База данных — это любая совокупность информации, которая была систематизирована для быстрого поиска с помощью компьютера. Базы данных используются для хранения, изменения, удаления и доступа к любым данным. Система управления базами данных (СУБД) использует запросы для извлечения информации из базы данных.

Модель базы данных определяет логическую структуру базы данных и то, каким образом данные могут храниться, организовываться и обрабатываться.

Существует три основных типа модели баз данных:

- дореляционные;
- реляционные;
- постреляционные.

### 2.4.1 Дореляционные модели

К дореляционным моделям баз данных относятся иерархическая и сетевая модели.

В иерархической модели записи связаны по принципу «родитель-потомок», при этом каждый родитель может быть связан с более чем одним дочерним элементом, но каждый дочерний элемент связан только с одним родителем. У такой модели есть проблемы с отношениями «многие ко многим». Кроме того, в иерархической модели добавление новых связей может привести к массовым изменениям существующей структуры.

Сетевая модель базы данных позволяет каждой записи иметь несколько родительских и несколько дочерних записей. Эта модель была создана на замену иерархической модели, так как она может обрабатывать отношения «многие ко многим».

Самая большая проблема с дореляционными моделями баз данных заключалась в отсутствие структурной независимости — внести структурные изменения в базу данных было очень сложно.

### 2.4.2 Реляционные модели

В реляционных моделях [3] данные организованы в набор двумерных взаимосвязанных таблиц. Каждая из которых представляет собой набор столбцов и строк, где столбец представляет атрибуты сущности, а строки представляют записи. Использование таблиц для хранения данных обеспечило простой и эффективный способ хранения структурированной информации, доступа к ней, а также легкую сортировку.

### 2.4.3 Постреляционные модели

Постреляционные базы данных [4] реализуют схему с использованием нереляционной модели данных. Такие БД поддерживают логическую модель данных, ориентированную на обработку транзакций. Постреляционные базы данных включают хранилища данных с ключевыми значениями, сетевые и гра-

фовые базы данных, а также базы данных, ориентированные на документы. Недостатком такой модели является сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных.

## 2.5 Вывод

В данном разделе были формализованы поставленная задача, данные, а также категории пользователя. Были рассмотрены модели баз данных. Для решения поставленной задачи была выбрана реляционная модель, так как для разрабатываемой базы данных важна целостность хранимых данных, а также простота хранения структурированной информации и ее сортировка.

# 3 Технологическая часть

## 3.1 Средства реализации ПО

Для написания Web-приложения был выбран язык программирования C# [5]. Такой решение обусловлено следующими причинами.

- Главной причиной выбора C# стала его Web-поддержка.
- Данный язык поддерживает объектно-ориентированную парадигму программирования, что было обязательным требованием для выбора языка разработки.
- C# многим похож на ранее знакомые мне языки программирования, такие как C++ и C, что также помогло склонить чашу весов в пользу C#.

В качестве среды разработки был выбран Visual Studio [6]. Такой решение обусловлено следующими причинами.

- Главным требованием при выборе среды разработки была Web-поддержка, которая имеется у Visual Studio.
- VS является бесплатным программным обеспечением, что также было важным требованием при выборе среды.
- Visual Studio доступна на macOS.

При выборе платформы Web-разработки рассматривались два фреймворка на платформе .NET. Это ASP.NET [7] и ASP.NET Core [8]. В пользу выбора второго из них повлияли следующие причины.

- Данная платформа имеет большую производительность по сравнению с ASP.NET.
- ASP.NET Core предназначена для Windows, macOS и Linux, в свою очередь как ASP.NET только для Windows.

Другие платформы не рассматривались, так как ранее в качестве языка программирования был выбран C#, а средой разработки Visual Studio.

## 3.2 Архитектура приложения

За основу архитектуры разрабатываемого Web-приложения была взята модель MVC (MODEL, VIEW, CONTROLLER). Эта модель помогает обеспечить разделение между бизнес-логикой программного обеспечения и отображением.

Фреймворк MVC включает в себя следующие три компонента.

- Модель – управляет данными и бизнес-логикой.
- Представление – используется для всей логики пользовательского интерфейса приложения.
- Контроллер – обеспечивает взаимосвязь между представлениями и моделью, поэтому он действует как посредник.

ASP.NET Core поддерживает модель разработки MVC. ASP.NET Core MVC framework [9] – это легкий, легко тестируемый фреймворк представления, который интегрирован с существующими ASP.NET Core функциями.

## 3.3 Выбор СУБД

В аналитическом разделе при выборе типа модели баз данных была выбрана реляционная модель, следовательно выбор СУБД будет производиться для реляционной модели баз данных. Рассмотрим только самые популярные из них: MySQL, Oracle, PostgreSQL [10], Microsoft SQL Server.

Выделим слудующие критерии для сравнения выбранных СУБД:

1. бесплатное распространение СУБД, что будет главным требованием при выборе;
2. наличие подробной документации, которая поможет разобраться во всех возникших проблемах;
3. наличие высокого уровня оптимизации СУБД, что будет отличным плюсом при выборе;

4. опыт работы с СУБД, не является основным требованием, но очень желательным.

Результаты сравнения выбранных СУБД по заданным критериям представлены в таблице 3.1.

| **Критерий** | **MySQL** | **Oracle** | **PostgreSQL** | **Microsoft SQL Server** |
|---|---|---|---|---|
| 1 | + | - | + | - |
| 2 | + | + | + | + |
| 3 | + | - | + | - |
| 4 | - | - | + | - |

Таблица 3.1 – Сравнение выбранных СУБД

По результатам сравнения в качестве СУБД для реляционной модели баз данных был выбран PostgreSQL, так как он еденственный удовлетворил всем четырем критериям сравнения.

## 3.4 Листинги кода

Листинг 3.1 – Функция вывода всех автомобилей

```
public IActionResult GetAllCars(CarSortState sortOrder = CarSortState.IdAsc,
                                string brand = null, string model = null,
                                uint minPrice = 0, uint maxPrice = 0,
                                string categoryName = null)
{
    ViewBag.Title = "Cars";

    ViewData["IdSort"] = sortOrder == CarSortState.IdAsc ?
        CarSortState.IdDesc : CarSortState.IdAsc;
    ViewData["BrandSort"] = sortOrder == CarSortState.BrandAsc ?
        CarSortState.BrandDesc : CarSortState.BrandAsc;
    ViewData["ModelSort"] = sortOrder == CarSortState.ModelAsc ?
        CarSortState.ModelDesc : CarSortState.ModelAsc;
    ViewData["CategoryNameSort"] = sortOrder ==
        CarSortState.CategoryNameAsc ? CarSortState.CategoryNameDesc :
        CarSortState.CategoryNameAsc;
    ViewData["PriceSort"] = sortOrder == CarSortState.PriceDesc ?
        CarSortState.PriceAsc : CarSortState.PriceDesc;

    IEnumerable<Car> cars = carService.GetByParameters(brand, model,
        minPrice, maxPrice, categoryName);

    CarViewModel carViewModel = new CarViewModel
    {
        cars = carService.GetSortCarsByOrder(cars, sortOrder),
        categories = categoryService.GetAll(),

        filterCarViewModel = new FilterCarViewModel
        {
            brand = brand,
            model = model,
            minPrice = minPrice,
            maxPrice = maxPrice,
            categoryName = categoryName
        }
    };

    return View(carViewModel);
}
```

Листинг 3.2 – Функция вывода всех добавленных в корзину автомобилей

```
public IActionResult GetMyCars(CarSortState sortOrder = CarSortState.IdAsc,
                               IsUpdata isUpdate = IsUpdata.IsNotUpdate,
                               int carId = 0)
{
    ViewBag.Title = "MyCars";

    ViewData["IdSort"] = sortOrder == CarSortState.IdAsc ?
        CarSortState.IdDesc : CarSortState.IdAsc;
    ViewData["BrandSort"] = sortOrder == CarSortState.BrandAsc ?
        CarSortState.BrandDesc : CarSortState.BrandAsc;
    ViewData["ModelSort"] = sortOrder == CarSortState.ModelAsc ?
        CarSortState.ModelDesc : CarSortState.ModelAsc;
    ViewData["CategoryNameSort"] = sortOrder ==
        CarSortState.CategoryNameAsc ? CarSortState.CategoryNameDesc :
        CarSortState.CategoryNameAsc;
    ViewData["PriceSort"] = sortOrder == CarSortState.PriceDesc ?
        CarSortState.PriceAsc : CarSortState.PriceDesc;

    User user = userService.GetByLogin(User.Identity.Name);
    IEnumerable<Car> myCars = carService.UpdateMyCars(isUpdate, user.Id,
        carId);

    MyCarsViewModel myCarsViewModel = new MyCarsViewModel
    {
        myCars = carService.GetSortCarsByOrder(myCars, sortOrder),
        categories = categoryService.GetAll(),

        car = carService.GetByID(carId),

        _isUpdate = isUpdate
    };

    return View(myCarsViewModel);
}
```

Листинг 3.3 – Функция поиска автомобилей по заданным параметрам

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Index(HomeViewModel model)
{
    ViewBag.Title = "Find";

    if (ModelState.IsValid)
    {
        uint minPrice = checkForNull(model.filterCarViewModel.minPrice);
        uint maxPrice = checkForNull(model.filterCarViewModel.maxPrice);

        return RedirectToAction("GetAllCars", "Car",
            new
            {
                brand = model.filterCarViewModel.brand,
                model = model.filterCarViewModel.model,
                minPrice = minPrice,
                maxPrice = maxPrice,
                categoryName = model.filterCarViewModel.categoryName
            });
    }
    else
        ModelState.AddModelError("", "Некорректные данные");

    return View(model);
}
```

Листинг 3.4 – Функция оформления заказа

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Checkout(OrderViewModels model)
{
    ViewBag.Title = "Order";

    if (ModelState.IsValid)
    {
        carService.DeleteMyCarsByUserLogin(User.Identity.Name);

        return RedirectToAction("Success", "Order");
    }
    else
        ModelState.AddModelError("", "Некорректные данные");

    return View(model);
}
```

Листинг 3.5 – Функция регистрации

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel model)
{
    ViewBag.Title = "Register";

    if (ModelState.IsValid)
    {
        try
        {
            User user = new User
            {
                Login = model.Login,
                Password = model.Password,
                Permission = "user"
            };

            userService.Add(user);

            await Authenticate(user);

            return RedirectToAction("Index", "Home");
        }
        catch (Exception ex)
        {
            Console.Write(ex.Message);
            ModelState.AddModelError("", ex.Message);
        }
    }
    else
        ModelState.AddModelError("", "Некорректные данные");

    return View(model);
}
```

Листинг 3.6 – Функция авторизации

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Login(LoginViewModel model)
{
    ViewBag.Title = "Login";

    if (ModelState.IsValid)
    {
        User user = userService.GetByLogin(model.Login);

        if (user != null && user.Password == model.Password)
        {
            await Authenticate(user);

            return RedirectToAction("Index", "Home");
        }
        else
            ModelState.AddModelError("", "Некорректные логин и(или) пароль
                ");
    }
    else
        ModelState.AddModelError("", "Некорректные логин и(или) пароль");

    return View(model);
}
```

## 3.5 Демонстрация работы веб-приложения

На рисунке 3.1 изображена начальная страница веб-приложения. Здесь можно реализовать поиск автомобиля по параметрам. На рисунке 3.2 показан выбор категории автомобиля.

Рисунок 3.1 – Начальная страница

Рисунок 3.2 – Выбор категории автомобиля для поиска по параметрам

На рисунках 3.3 - 3.4 изображены страницы с разными видами автомобилей.

Рисунок 3.3 – Страница с автомобилями с ДВС

Рисунок 3.4 – Страница с электромобилями

На страницах с автомобилями реализована сортировка по четырём параметрам (как по возрастанию, так и по убыванию). На рисунках 3.5 - 3.6 изображены страницы с отсортированными автомобилями.

Рисунок 3.5 – Автомобили, отсортированные по убыванию их стоимости

Рисунок 3.6 – Автомобили, отсортированные по их моделям в алфавитном порядке

На рисунках 3.7 - 3.8 показано добавление одного автомобиля в корзину, а также удаление другого из неё.

Рисунок 3.7 – Добавление автомобиля в корзину

Рисунок 3.8 – Удаление автомобиля из корзины

Добавленные в корзину автомобили можно заказать, оформив заказ. На рисунках 3.9 - 3.10 показано оформление заказа.

Рисунок 3.9 – Страница оформления заказа

Рисунок 3.10 – Заказ успешно оформлен

На рисунках 3.11 - 3.12 изображена регистрация и авторизация пользователя.

Рисунок 3.11 – Страница регистрации

Рисунок 3.12 – Страница авторизации

Если пользователь не авторизировался, то он не сможет добавить понравившиеся автомобили в корзину, а также оформить заказ.

## 3.6 Вывод

В данном разделе были описаны средства реализации ПО, архитектура приложения, был произведен выбор СУБД для реляционной модели БД. После сравнения самых популярных СУБД по выделенным критериям был выбран PostgreSQL, так как он единственный подошел по всем требованиям. Также были приведены листинги кода некоторых функций программного обеспечения. Была продемонстрирована работа веб-приложения.

# Заключение

Во время выполнения практики была разработана база данных, а также создано Web-приложение для интернет магазина автомобилей. Были рассмотрены модели баз данных и произведен выбор СУБД для реляционной модели БД. После сравнения самых популярных СУБД по выделенным критериям для поставленной задачи был выбран PostgreSQL.

Проделанная работа позволила лучше изучить как язык программирования C#, так и среду разработки Visual Studia.

Также в ходе выполнения практики были выполнены следующие задачи:

- формализовать задачу и определить необходимый функционал;
- рассмотреть модели баз данных и выбрать подходящую;
- проанализировать существующие СУБД и выбрать нужную;
- спроектировать и разработать БД;
- спроектировать и разработать WEB-приложение.

Поставленная цель была достигнута.

# Литература


[1] Аналитическое агентство АВТОСТАТ [Электронный ресурс]. – URL: https://www.autostat.ru (дата обращения: 18.07.2022).

[2] Научно-исследовательский институт точных приборов [Электронный ресурс]. – URL: https://niitp.ru (дата обращения: 18.07.2022).

[3] What is a Relational Database (RDBMS)? [Электронный ресурс]. – URL: https://www.oracle.com/database/what-is-a-relational-database/ (дата обращения: 18.07.2022).

[4] Нереляционные данные и базы данных NoSQL [Электронный ресурс]. – URL: https://www.oracle.com/database/what-is-a-relational-database/ (дата обращения: 18.07.2022).

[5] Документация по C# [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 18.07.2022).

[6] Visual Studio [Электронный ресурс]. – URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 18.07.2022).

[7] ASP.NET [Электронный ресурс]. – URL: https://dotnet.microsoft.com/en-us/apps/aspnet (дата обращения: 18.07.2022).

[8] Общие сведения об ASP.NET Core [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0 (дата обращения: 18.07.2022).

[9] Общие сведения ASP.NET Core MVC [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-6.0 (дата обращения: 18.07.2022).

[10] PostgreSQL [Электронный ресурс]. – URL: https://www.postgresql.org/docs/ (дата обращения: 18.07.2022).