

Информационная безопасность АС

Павлова Виктория Сергеевна

г. Тула, ООО «Диди Плэнет Интеграция»

с 07 июля 2023 г. по 20 июля 2023 г.

Производственная (технологическая)

второй

Кафедра Информационная безопасность

**23**

# **Содержание**

[**Дневник прохождения производственной практики** 3](#_Toc140594669)

[**Характеристикa** 5](#_Toc140594670)

[**Введение** 6](#_Toc140594671)

[**I Общая характеристика порядка проведения практики** 7](#_Toc140594672)

[***1.1*** ***Порядок организации практики*** 7](#_Toc140594673)

[***1.2*** ***Инструктаж по режиму работы и технике безопасности*** 8](#_Toc140594674)

[**II Сведения о предприятии по месту прохождения практики** 10](#_Toc140594675)

[***1.1*** ***Общие сведения об организации*** 10](#_Toc140594676)

[***1.2*** ***Функционал отделов и структура организации*** 11](#_Toc140594677)

[***1.3*** ***Стек технологий*** 12](#_Toc140594678)

[**III Анализ автоматизации на предприятии** 14](#_Toc140594679)

[***1.1*** ***Классификация имеющихся на предприятии информационных систем*** 14](#_Toc140594680)

[***1.2*** ***Предложения по усовершенствованию автоматизации на предприятии*** 15](#_Toc140594681)

[**IV Описание этапов выполнения индивидуального задания** 18](#_Toc140594682)

[***1.1*** ***Цели и задачи разработки в архитектурном стиле REST API*** 18](#_Toc140594683)

[***1.2*** ***Перечень автоматизированных функций*** 20](#_Toc140594684)

[***1.3*** ***Обеспечение безопасного хранения пользовательских данных*** 21](#_Toc140594685)

[**Заключение** 23](#_Toc140594686)

[**Список использованных источников** 24](#_Toc140594687)

[**Приложение** 25](#_Toc140594688)

# **Дневник** **прохождения производственной практики**

Студента Павловой Виктории Сергеевны

На базе ООО «Диди Плэнет Интеграция»

Период прохождения: с 07.07.2023 по 20.07.2023 в качестве практиканта

|  |  |  |
| --- | --- | --- |
| **Дата** | **Наименование подразделения организации** | **Краткое содержание работы** |
| 7.07.2023 | ООО «Диди Плэнет Интеграция» | Прибытие на производственную практику, изучение структуры и основной деятельности компании, а также ознакомление с инструктажем по работе с информацией в контексте разработки приложений. |
| 10.07.2023 | ООО «Диди Плэнет Интеграция» | Получение индивидуального задания по разработке веб-приложения REST API для работы с реляционной базой данных MSSQL. |
| 11.07.2023 | ООО «Диди Плэнет Интеграция» | Описание таблиц и создание базы данных каталога услуг управляющей компании на основе списка необходимых для реализации сущностей. |
| 12.07.2023 | ООО «Диди Плэнет Интеграция» | Настройка базы данных, установка связей между таблицами с помощью внешних ключей. |
| 13.07.2023 | ООО «Диди Плэнет Интеграция» | Настройка валидации адресов для отсечения незаполненных промежуточных уровней адресации и отсеивания пересечений локаций в базе данных согласно родительским адресам ФИАС. |
| 14.07.2023 | ООО «Диди Плэнет Интеграция» | Создание сервисов для обработки данных, моделей сущностей и настройка контроллеров. |
| 17.07.2023 | ООО «Диди Плэнет Интеграция» | Настройка безопасности приложения, реализация хэширования данных пользователя. |
| 18.07.2023 | ООО «Диди Плэнет Интеграция» | Написание функций для управления каталогом услуг и справочником локаций управляющей компании с использованием фреймворка Swagger. |
| 19.07.2023 | ООО «Диди Плэнет Интеграция» | Тестирование созданных функций для работы с приложением с помощью Swagger, инструментария, использующего спецификацию OpenAPI. |
| 20.07.2023 | ООО «Диди Плэнет Интеграция» | Завершение тестирования приложения и окончание практики. Подведение итогов практики и составление отчёта. |

|  |  |
| --- | --- |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ФИО, подпись) |
| Руководитель практики от организации | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ФИО, подпись) |

# **Характеристикa**

Студент 2 курса ФГБОУ ВО «Тульский Государственный Университет» Павлова Виктория Сергеевна в период с 7 июля по 20 июля 2023 года проходила производственную (технологическую) практику в качестве практиканта в ООО «Диди Плэнет Интеграция».

Во время прохождения практики студент была ознакомлена со структурой и основной деятельностью IT-компании, в частности, с циклом создания digital-сервисов от аудита и проектирования до техподдержки и продвижения, а также была активно вовлечена в процесс разработки web-приложения.

Студенту было выдано индивидуальное задание по разработке каталога услуг и справочника локаций (зон действия) управляющих компаний. При прохождении практики студентом было разработано приложение с архитектурой REST API, а в целях повышения квалификации с помощью алгоритмов хэширования студентом был дополнительно реализован метод для обеспечения безопасности пользовательских данных.

За период прохождения практики студент проявила активность, внимательность, трудолюбие и ответственность, показала себя как коммуникабельный и организованный работник, который нацелен на результат и способен быстро обучаться новому. Индивидуальное задание практикант выполнила успешно в соответствии с профилем студента – разработка и обеспечение безопасности автоматизированных информационных систем.

Результаты практики могут быть оценены «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Руководитель практики от организации \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО, роспись, печать)

# **Введение**

Данный отчёт представляет собой документацию по производственной (технологической) практике, выполненной студентом специальности «Информационная безопасность автоматизированных систем» на втором курсе обучения. Практика была пройдена в ООО «Диди Плэнет Интеграция».

Целью прохождения производственной практики является приобретение ценных практических навыков и опыта профессиональной деятельности в области разработки и эксплуатации веб-приложений, а также обеспечения их информационной безопасности.

В течение прохождения практики мною было получено и успешно выполнено индивидуальное задание по разработке приложения для реализации каталога услуг и справочника локаций.

Данный отчёт содержит описание результатов работы, в частности, скрипты создания описанных таблиц, а также код сервиса для работы с базой данных, реализующий функции добавления, удаления и изменения сущностей базы данных. Он также содержит выводы о проделанной работе и предложения по улучшению процесса обеспечения информационной безопасности на предприятии.

Прохождение практики в IT-компании «Диди Плэнет Интеграция» представляет собой существенный этап в профессиональном становлении и позволяет овладеть ценным практическом опытом, который призван внести вклад в дальнейшее развитие в сфере обеспечения информационной безопасности автоматизированных систем.

# **I Общая характеристика порядка проведения практики**

## ***Порядок организации практики***

Производственная (технологическая) практика играет существенную роль в профессиональном развитии студента. Она представляет собой неотъемлемый этап, который помогает студенту перейти от теоретических знаний, полученных в учебном процессе, к их практическому применению в реальной профессиональной среде.

Согласно положению о практике ТулГУ, практика – вид учебных занятий, направленный на получение первичных профессиональных умений и навыков, а также интеграцию теоретической и профессионально-практической, учебной и научно-исследовательской деятельности студентов. Первым этапом для успешного её прохождения становится заключение договора между учебным заведением и организацией, на базе которой планируется проведение практики [1].

Прохождение студентами практик осуществляется на основании:

– договоров, заключенных между университетом и предприятиями,

учреждениями и организациями;

– гарантийных писем от предприятий, учреждений и организаций о приеме студентов на практику.

Выпускающая кафедра не позднее, чем за 25 дней до начала практики готовит приказ по университету о направлении студентов на практику и назначении руководителей практики от кафедры. Для руководства практикой студентов назначаются руководители практики от выпускающих кафедр университета и от предприятий, учреждений, организаций. В дальнейшем порядок организации практики определяется индивидуальным заданием на предприятии. В предшествии выдачи индивидуального задания в рамках прохождения практики в IT-компании со мной было проведено краткое собеседование об имеющихся компетенциях, реализованных проектах, а также об уже освоенном стеке технологий.

В течение срока прохождения практики мною был написан дневник прохождения практики, содержащий в себе описание ежедневных задач для достижения целей практики, в частности, для выполнения индивидуального задания. В процессе выполнения задания практики между мной и руководителем практики от предприятия шёл непрерывный процесс коммуникации по всем возникающим технических и организационным вопросам, что активно вовлекало меня в рабочий процесс, максимально приближенный к реальным условиям.

## ***Инструктаж по режиму работы и технике безопасности***

Прохождению практики в IT-компании «Диди Плэнет Интеграция» предшествовало проведение инструктажа и ознакомление с техникой безопасности работы в организации, содержание которых приведено ниже:

1. Знакомство с рабочим местом: перед началом работы необходимо ознакомиться со своим рабочим местом, его оборудованием и расположением экстренных выходов;
2. Основные правила безопасности:
   * Соблюдать все инструкции по технике безопасности, предоставленные компанией;
   * Используйте инструменты и оборудование (компьютеры, принтеры, ПО) только по их назначению и после обучения;
   * Соблюдать порядок и чистоту на рабочем месте;
3. Работа с электрооборудованием:
   * Перед началом работы проверять состояние электрооборудования на наличие повреждений;
   * Включать и выключать любое электрооборудование только при необходимости;
4. Пожарная безопасность:
   * Ознакомиться с местами расположения огнетушителей и огнезащитных средств;
   * Не хранить горючие материалы рядом с источниками тепла или электрооборудованием;
   * В случае возникновения пожара, немедленно известить о нём и следовать указаниям по эвакуации;

# **II Сведения о предприятии по месту прохождения практики**

## ***Общие сведения об организации***

Digital-интегратор DD Planet основан в 2004 году, специализируется на создании digital-сервисов и информационно-аналитических систем [2].

Миссия IT-компании заключается в разработке и предоставлении инновационных решений в области информационных технологий, создании программного обеспечения, IT-консалтинге и обеспечении безопасности данных. Компания занимается реализацией высоконагруженных веб-сервисов, создаёт корпоративные порталы и мобильные приложения, а также занимаемся их продвижением в digital. Среди собственных проектов — отраслевые порталы Выберу.ру и МирКвартир.ру, что стоят на одном уровне с сервисами Яндекса, госуслуг и другими популярными онлайн-ресурсам. За 19 лет создано более 500 проектов для крупнейших брендов.

Организация расположена по адресу г. Тула, ул. Жуковского, д. 58. Бурденко Григорий Юрьевич является генеральным директором организации и учредителем компании. В команде имеется свыше 120 специалистов, а также 3 офиса: в Москве, Туле и Калуге [2].

Для сотрудников компании предусмотрена возможность как внешнего, так и внутреннего обучения. При необходимости возможно отправиться на курсы дополнительного образования, профессиональную сертификацию и конференции, которые частично или полностью оплачиваются за счет компании. Сотрудники компании DD Planet с 2018 года принимают участие в хакатонах и IT-олимпиадах в качестве наставников и экспертов.

Новички обучаются по индивидуальному плану под руководством кураторов. Регулярно проводятся внутренние встречи, на которых коллеги делятся друг с другом успешными практиками и наработками. Мероприятия ориентированы на обмен профессиональным опытом в области backend, frontend, тестирования, дизайна и т.д., а также развития soft skills. Для студентов в Туле, только начинающих свой путь в разработке, действует круглогодичная оплачиваемая стажировка в нашем офисе.

Основным видом деятельности является веб-разработка, разработка мобильных приложений, техподдержка и SLA, автоматизация бизнеса, digital-продвижение, UX/UI-дизайн. Кроме того, компания также предоставляет консультативные услуги, работает в области компьютерных технологий и управлению компьютерным оборудованием.

## ***Функционал отделов и структура организации***

IT-компании DDPlanet занимается разработкой компьютерного программного обеспечения. Её организационная структура включает в себя следующие основные элементы:

1. Высшее руководство:
   * Генеральный директор (Бурденко Григорий Юрьевич) осуществляет общее управление компанией, определяет стратегические цели и направления развития.
   * Исполнительный директор: отвечает за операционную деятельность компании, координирует работу подразделений и решает текущие вопросы.
2. Отделы и подразделения:
   * Отдел разработки: занимается созданием программного обеспечения, включая разработку и тестирование новых продуктов и обновлений.
   * Отдел проектного управления: отвечает за планирование, координацию и контроль выполнения проектов, обеспечивает их своевременную и успешную реализацию.
   * Отдел качества: осуществляет контроль качества разрабатываемых продуктов, тестирование, обеспечивает соответствие стандартам и требованиям безопасности.
   * Отдел маркетинга: занимается анализом рынка, разработкой маркетинговых стратегий, продвижением продуктов компании и управлением брендом.
   * Отдел продаж: отвечает за поиск клиентов, проведение переговоров, заключение контрактов и обеспечение удовлетворения потребностей клиентов.
   * Отдел информационной безопасности: обеспечивает защиту информации, включая мониторинг угроз, внедрение систем безопасности и обучение сотрудников.
   * Отдел IT-инфраструктуры: отвечает за поддержку и обновление IT-инфраструктуры компании, управление сетевыми системами и обеспечение бесперебойной работы.
   * Отдел персонала: осуществляет найм, подбор, обучение и развитие сотрудников, управление кадровыми процессами и оценку производительности.
3. Команды проектов.

Каждый проект может иметь собственную команду, состоящую из разработчиков, тестировщиков, аналитиков, дизайнеров и других специалистов, необходимых для успешной реализации проекта. Команды работают под руководством проектного менеджера, который отвечает за планирование, организацию и управление проектом.

## ***Стек технологий***

Компания «DDPlanet» разрабатывает высоконагруженные цифровые системы с большими объемами данных и интеграциями. К стеку её технологий относится:

1. *В рамках веб-разработки:*
   * Разработка на .NET, PHP, Python.
2. *Мобильные приложения:*
   * Приложения на Xamarin;
   * Приложения на React Native;
   * Приложения под Android;
   * Приложения под IOS;
   * Приложения на Flutter.
3. *В рамках автоматизации бизнес-процессов:*
   * Автоматизация бизнес-процессов
   * Облачная версия Битрикс24
   * CRM интеграция и настройка
   * Корпоративные порталы
   * В2В-портал
   * Внедрение машинного обучения в управление бизнесом
   * Автоматизация отчетности и внедрение Bl-систем
   * Интеграция Битрикс24
4. *SEO Driven Development.*

К числу используемых сред разработки относится Visual Studio Code, а для контроля версий используется Git. Работа с .Net происходит в среде Visual Studio и Rider, с базами данных – в SQL Management Studio, MongoDB Compass, Robo3t и DBeaver для Linux.

# **III Анализ автоматизации на предприятии**

## ***Классификация имеющихся на предприятии информационных систем***

Для компании, занимающейся информационными технологиями, можно привести следующую классификацию информационных систем:

1. *Управление отношениями с клиентами (CRM – Customer Relationship Management):*

Данные системы предназначены для управления взаимодействием с клиентами, учета клиентской базы, отслеживания продаж, управления контактами и прогнозирования спроса.

1. *Управление предприятием (ERP – Enterprise Resource Planning):*

ERP-системы интегрируют различные бизнес-процессы и подразделения компании, такие как управление запасами, финансы, производство, управление проектами и ресурсами, что позволяет эффективно управлять ресурсами компании в целом.

1. *Управление проектами (Project Management):*

Эти системы предназначены для планирования, организации и управления проектами, включая управление задачами, ресурсами, сроками, бюджетом и коммуникациями внутри команды.

1. *Управление контентом (CMS – Content Management System):*

CMS-системы используются для создания, управления и публикации контента на веб-сайтах, блогах и других онлайн-платформах.

1. *Управление отношениями с поставщиками (SRM - Supplier Relationship Management):*

SRM-системы позволяют управлять отношениями с поставщиками, включая учет и оценку поставщиков, управление заказами и контрактами, а также анализ эффективности сотрудничества.

1. *Управление персоналом (HRM – Human Resource Management):*

HRM-системы предоставляют инструменты для управления рекрутингом, наймом, обучением, оценкой производительности, учетом рабочего времени и другими аспектами управления персоналом.

1. *Управление информационной безопасностью (ISM – Information Security Management):*

ISM-системы обеспечивают защиту информации и управление рисками, связанными с безопасностью, включая мониторинг угроз, управление доступом, шифрование данных и аудит безопасности.

1. *Управление облачными сервисами (Cloud Management):*

Эти системы предназначены для управления облачными ресурсами, включая развертывание, масштабирование, мониторинг и оптимизацию облачных сервисов.

## ***Предложения по усовершенствованию автоматизации на предприятии***

Проводя анализ текущего уровня автоматизации систем организации, на базе которой осуществлялось прохождение практики, можно сделать вывод о высоком уровне автоматизации благодаря использованию передовых технологий и инструментов, в частности, технологий machine learning и использованию искусственного интеллекта. На мой взгляд, в целях усовершенствования имеющегося прогресса можно рассмотреть следующие области развития:

* Автоматическое тестирование и использование инструментов для автоматизации тестирования позволяет значительно сократить ручное тестирование и ускорить процесс проверки функциональности и качества программного обеспечения.
* Непрерывная интеграция и доставка как автоматизация процессов сборки, тестирования и развертывания позволяет быстро и эффективно интегрировать изменения в код, тестировать и доставлять их в продукцию.
* Оркестрация и автоматизация: использование инструментов для автоматизации управления инфраструктурой и облачными сервисами позволяет быстро развертывать и масштабировать ресурсы, управлять сетями и обеспечивать высокую доступность систем.
* Контейнеризация: использование контейнерных технологий, таких как Docker, позволяет автоматизировать процессы развертывания и управления приложениями, обеспечивая их изолированное и независимое выполнение.

В современных IT-компаниях имеется тенденция к активному использованию методов машинного обучения и искусственного интеллекта для автоматизации и оптимизации различных процессов. Данное направление для развития и автоматизации также является очень перспективным, в частности потому, что машинное обучение позволяет создавать модели, которые автоматически анализируют большие объемы данных, выявляют скрытые закономерности и предоставляют ценные инсайты для бизнеса. Это позволяет компаниям принимать обоснованные решения на основе данных и оптимизировать процессы.

Методы искусственного интеллекта, такие как обработка естественного языка (Natural Language Processing, NLP), используются для автоматического анализа, понимания и обработки текстовой информации. Это может быть полезно для автоматической классификации текстов, создания чат-ботов, анализа настроений и многих других задач. Методы машинного обучения используются для создания рекомендательных систем, которые предлагают пользователям персонализированные рекомендации на основе анализа их предпочтений и поведения. Такие системы активно применяются в электронной коммерции, стриминговых сервисах, социальных сетях и других платформах.

Машинное обучение и искусственный интеллект применяются для прогнозирования и оптимизации различных параметров и результатов, таких как спрос на продукцию, ценообразование, управление запасами, рекламные кампании и многие другие. Внедрение методов машинного обучения и искусственного интеллекта позволяет IT-компаниям сократить ручной труд, повысить эффективность, снизить риски и обеспечить автоматизацию сложных задач. Эти технологии оказывают существенное влияние на различные отрасли и являются ключевыми элементами цифровой трансформации.

# **IV Описание выполнения индивидуального задания**

## ***Цели и задачи разработки в архитектурном стиле REST API***

REST API – это архитектурный стиль, способ организации API, то есть взаимодействия клиента с сервером [3]. Его можно охарактеризовать следующими тезисами:

* Модель — это объект, который представляет данные в нашем приложении (простой класс C#).
* Контроллер — это объект, который обрабатывает HTTP-запрос и создает HTTP-ответ.
* Клиентом является ПО, которое использует веб-API (браузер, мобильное приложение и прочее).
* Репозиторий — это объект, который инкапсулирует уровень данных и содержит логику для извлечения данных и направлениях их к модели. Интерфейс определяет основные операции CRUD (create, read, update, delete). Сперва создаётся интерфейс IRepository (напр. IService), а затем класс, который его реализует [4].
* Единый интерфейс (все запросы осуществляются по одному URL)
  + POST – добавить информацию
  + PUT – обновить информацию
  + DELETE – удалить
  + GET – получить

Для удобства в процессе разработки использовался Swagger —машиночитаемое представление RESTful API, который представляет удобный и простой пользовательский интерфейс.

Целью индивидуального задания является разработка бэкенд части веб-приложения. Задачами являются описание сущностей, согласно техническому заданию, а также реализация методов работы с базой данных. Индивидуальное задание в рамках проведения практики содержит следующий перечень сущностей для реализации:

Таблица 1 – Описание сущностей для каталога услуг

|  |  |
| --- | --- |
| **Название** | **Поля** |
| Управляющая компания | – id  – название |
| Локация (зона действия) | – id  – название  – id управляющей компании (foreign key на таблицу управляющих компаний) |
| Адрес в локации | – код дома ФИАС  – id локации (foreign key на таблицу локаций) |
| Позиция каталога (услуга) | – id  – название  – минимальное количество элементов в заказе  – максимальное количество элементов в заказе  – единицы измерения (просто строка) – id управляющей компании |
| Цена по локации | – id  – цена (целое число, в копейках)  – признак "цена по запросу" (если true, то цена равна 0)  – id локации (foreign key на таблицу локаций)  – id позиции каталога (foreign key на таблицу услуг) |

В таблице 2 приведено описание сущностей для справочника локаций управляющей компании.

Таблица 2 – Описание сущностей для справочника локаций

|  |  |
| --- | --- |
| **Название** | **Поля** |
| Управляющая компания | – id  – название |
| Локация (зона действия) | – id  – название  – id управляющей компании (foreign key на таблицу управляющих компаний) |
| Адрес в локации | – код дома по ФИАС  – код улицы по ФИАС  – код города по ФИАС  – код региона по ФИАС  – id локации (foreign key на таблицу локаций) |

Дополнительным условием является тот факт, что адреса в локациях могут быть на любом уровне ФИАС. Для адресов в локации заполняются уровни ФИАС выше текущего. Для дома – улица, город, регион. Для улицы – город, регион. Для города – регион.

## ***Перечень автоматизированных функций***

Для осуществления работы с базой данных, а именно добавления, удаления и редактирования данных, согласно требованиям индивидуального задания, было реализовано несколько различных методов в рамках архитектуры REST API. Разработка производилась в среде Visual Studio на языке программирования C# с использованием фреймворков Swagger и EntityFrameworkCore. Скрипты создания описанных таблиц приведены в приложении в листинге 1, а код сервиса для работы с базой данных – в листинге 2. Перечень автоматизированных функций приведён в таблице 3 на примере класса-сервиса для адресов.

Таблица 3 – Список реализованных функций для обработки запросов

|  |  |
| --- | --- |
| **Название и поля** | **Описание** |
| public AdressService(SqlDriver sqlDriver = null) | Конструктор сервиса, осуществляющий подключение к базе данных |
| public string AddNewAdress(AdressModel model) | Добавление новой записи об адресе согласно модели адреса (post-запрос) |
| public bool TryAddNewAdress(int idMC, AdressModel model) | Метод валидации адреса по ФИАС |
| public AdressModel GetAdressByID(int id) | Получение сведений об адресе из базы (get-запрос) |
| public AdressModel ChangeAdressByID(int id, AdressModel newModel) | Обновление сведений об адресе (put-запрос) |
| public string DeleteAdressByID(int id) | Удаление записи из базы (delete-запрос) |

Заполнение сведений об адресе должно быть последовательным: регион → город → улица → дом. Учтя, что каждый адрес принадлежит локации, а все адреса в рамках одной локации (области действия УК) разные, нужно проверять, чтобы в соседних локациях не было пересечения адресов. Валидация происходит следующим образом: проверка на совпадение начинается с региона, и если у одной из моделей регион заполнен, а у другой нет, то далее проверка идёт по городу и т.д. При совпадении хотя бы одного из полей метод валидации возвращает BadRequest().

## ***Обеспечение безопасного хранения пользовательских данных***

В рамках прохождения практики я узнала, что хранение пользовательских данных, в частности, паролей в «чистом» виде является крайне небезопасной практикой. В качестве метода обеспечения безопасности в своем веб-приложении я выбрала хэширование пользовательских данных по алгоритму MD5, удобная работа с которым осуществляется с помощью методов пространства имён System.Security.Cryptography. [4]

На вход метода подается строка данных произвольной длины, а на выходе получается хэш-сумма длиной 128 бит. MD5 (Message Digest Algorithm 5) – это криптографический алгоритм хэширования, который принимает на вход произвольное сообщение переменной длины и вычисляет 128-битный хэш (дайджест) этого сообщения. MD5 был разработан Рональдом Ривестом в 1991 году и широко применялся для проверки целостности данных и хранения хэшей паролей [4].

Таким образом, в базе данных хранится не сам пароль пользователя, а хэш-сумма, что, даже если база данных с хэшами паролей станет доступной злоумышленнику, это исключает возможность их использования, ведь хэширование делает процесс восстановления паролей чрезвычайно сложным и времязатратным – для строк разной длины итоговая хэш-сумма всё равно имеет фиксированную длину. Алгоритмы хэширования обладают свойством необратимости, что означает, что хэш нельзя просто обратно преобразовать в исходный пароль. Это предотвращает раскрытие исходного пароля даже в случае компрометации базы данных.

# **Заключение**

Производственная практика дает студентам ценный практический опыт, позволяющий им столкнуться с реальными вызовами и проблемами, с которыми они столкнутся в будущей профессиональной деятельности. Они учатся адаптироваться к рабочей среде, работать в команде, применять свои знания и навыки для решения конкретных задач. Этот опыт помогает студентам лучше понять, как они могут применять свои учебные знания на практике и какие навыки им следует развивать для успешной карьеры в выбранной области.

Резюмируя, можно сказать, что технологическая практика является связующим звеном между миром учебных аудиторий и реальной индустрией. В учебной среде студенты осваивают основные концепции, принципы и теоретические модели, которые служат фундаментом для понимания своей будущей профессии. Однако именно через практику появляется возможность реализовать эти знания и увидеть, как они проявляю себя в реальных рабочих ситуациях.

Проводя анализ полученных компетенций, прежде всего можно выделить получение навыка обработки пользовательских запросов на стороне сервиса, то есть получение опыта бэкенд-разработки на .Net. Второй немаловажный приобретённый навык – работа с реляционными базами данных, в частности, создание таблиц и запросов в MSSQL.

Разработанное веб-приложение с точки зрения бэкенд-разработки является полностью функциональным и протестированным, описанные методы работают правильно и корректно отображают необходимые данные согласно техническому индивидуальному заданию. Прохождение практики на базе IT-компании я нахожу интересным, продуктивным и значимым в области моего профессионального развития.

# **Список использованных источников**

1. Положение по практикам // Тульский государственный университет URL: https://clck.ru/3533nV (дата обращения: 18.07.2023).
2. IT-компания DDPlanet URL: https://www.ddplanet.ru (дата обращения: 18.07.2023).
3. REST API: введение в технологию // Сообщество IT-специалистов / Хабр URL: https://habr.com/ru/articles/590679/ (дата обращения: 18.07.2023).
4. Вострецова, Е.В. В78 Основы информационной безопасности : учебное пособие для студентов вузов / Е.В. Вострецова.— Екатеринбург : Изд-во Урал. ун-та, 2019.— 204 с

# **Приложение**

Листинг 1 – Скрипты создания описанных таблиц

CREATE TABLE [dbo].[Adresses] (

[id] INT IDENTITY (1, 1) NOT NULL,

[idLocation] INT NOT NULL,

[streetFIASCode] NVARCHAR (50) NULL,

[cityFIASCode] NVARCHAR (50) NULL,

[regionFIASCode] NVARCHAR (MAX) NOT NULL,

[houseFIASCode] INT NULL,

PRIMARY KEY CLUSTERED ([id] ASC),

FOREIGN KEY ([idLocation]) REFERENCES [dbo].[Locations] ([id])

);

CREATE TABLE [dbo].[Locations] (

[id] INT IDENTITY (1, 1) NOT NULL,

[name] NVARCHAR (50) NOT NULL,

[idMC] INT NOT NULL,

PRIMARY KEY CLUSTERED ([id] ASC),

FOREIGN KEY ([idMC]) REFERENCES [dbo].[ManagementCompanies] ([id])

);

CREATE TABLE [dbo].[ManagementCompanies] (

[id] INT IDENTITY (1, 1) NOT NULL,

[name] NVARCHAR (50) NOT NULL,

PRIMARY KEY CLUSTERED ([id] ASC)

);

CREATE TABLE [dbo].[Prices] (

[id] INT IDENTITY (1, 1) NOT NULL,

[price] INT NOT NULL,

[ifPriceOnRequest] BIT NOT NULL,

[idLocation] INT NOT NULL,

[idService] INT NOT NULL,

[name] NVARCHAR (50) DEFAULT (N'unnamed') NOT NULL,

PRIMARY KEY CLUSTERED ([id] ASC),

FOREIGN KEY ([idLocation]) REFERENCES [dbo].[Locations] ([id]),

FOREIGN KEY ([idService]) REFERENCES [dbo].[Services] ([id])

);

CREATE TABLE [dbo].[Services] (

[id] INT IDENTITY (1, 1) NOT NULL,

[name] NVARCHAR (50) NOT NULL,

[minCount] INT NOT NULL,

[maxCount] INT NOT NULL,

[measure] NVARCHAR (50) NOT NULL,

[idMC] INT NOT NULL,

PRIMARY KEY CLUSTERED ([id] ASC),

FOREIGN KEY ([idMC]) REFERENCES [dbo].[ManagementCompanies] ([id])

);

CREATE TABLE [dbo].[Users] (

[id] INT IDENTITY (1, 1) NOT NULL,

[email] NVARCHAR (50) NOT NULL,

[login] NVARCHAR (50) NOT NULL,

[passwordHash] NVARCHAR (50) NOT NULL,

[roleCode] INT NULL,

PRIMARY KEY CLUSTERED ([id] ASC)

);

Листинг 2 – Код сервиса для работы с базой данных

namespace MCService.Database

{

public class SqlDriver

{

private SqlConnection sqlConnection = null;

public void Open()

{

sqlConnection = new SqlConnection("Data Source = (LocalDB)\\MSSQLLocalDB; AttachDbFilename = D:\\GIT\\MCApp\\MCServiceCatalog.mdf; Integrated Security = True");

sqlConnection.Open();

}

public string ExecNonQuery(string query)

{

SqlCommand command = new SqlCommand(query, sqlConnection);

return command.ExecuteNonQuery().ToString();

}

public string ExecScalar(string query)

{

SqlCommand command = new SqlCommand(query, sqlConnection);

return command.ExecuteScalar().ToString();

}

public SqlDataReader ExecReader(string query)

{

SqlCommand command = new SqlCommand(query, sqlConnection);

return command.ExecuteReader();

}

public void Close()

{

sqlConnection.Close();

}

}

namespace MCService.Models

{

public class BaseModel

{

public string Name { get; set; }

}

public class LocationModel : BaseModel

{

public int CompanyID { get; set; }

}

public class PriceModel : BaseModel

{

public int Price { get; set; }

public bool isPriceOnRequest { get; set; }

public int LocationID { get; set; }

public int ServiceID { get; set; }

}

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

public class ServiceModel : BaseModel

{

public int MinCount { get; set; }

public int MaxCount { get; set; }

public string Measurement { get; set; }

public int CompanyID { get; set; }

}

public class UserModel

{

public string Email { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public UserRole Role { get; set; }

public enum UserRole

{

Admin = 0,

Customer = 1

}

}

public class AdressModel

{

public int LocationID { get; set; }

public FIASCode fullFIASCode { get; set; }

}

public struct FIASCode

{

public int HouseFIASCode { get; set; }

public string StreetFIASCode { get; set; }

public string CityFIASCode { get; set; }

public string RegionFIASCode { get; set; }

public FIASCode(int houseCode, string streetCode, string cityCode, string regionCode)

{

HouseFIASCode = houseCode;

StreetFIASCode = streetCode;

CityFIASCode = cityCode;

RegionFIASCode = regionCode;

}

public FIASCode(string streetCode, string cityCode, string regionCode)

{

HouseFIASCode = -1;

StreetFIASCode = streetCode;

CityFIASCode = cityCode;

RegionFIASCode = regionCode;

}

public FIASCode(string cityCode, string regionCode)

{

HouseFIASCode = -1;

StreetFIASCode = string.Empty;

CityFIASCode = cityCode;

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

RegionFIASCode = regionCode;

}

public FIASCode(string regionCode)

{

HouseFIASCode = -1;

StreetFIASCode = string.Empty;

CityFIASCode = string.Empty;

RegionFIASCode = regionCode;

}

}

}

namespace MCService.Services

{

public class AdressService

{

private readonly SqlDriver sqlDriver;

public AdressService(SqlDriver sqlDriver = null)

{

this.sqlDriver = sqlDriver;

this.sqlDriver.Open();

}

~AdressService()

{

sqlDriver.Close();

}

public string AddNewAdress(AdressModel model)

{

return sqlDriver.ExecNonQuery("INSERT INTO Adresses "

+ "(idLocation, streetFIASCode, cityFIASCode, regionFIASCode, houseFIASCode) "

+ $"VALUES ({model.LocationID}, N'{model.fullFIASCode.StreetFIASCode}', N'{model.fullFIASCode.CityFIASCode}', "

+ $"N'{model.fullFIASCode.RegionFIASCode}', {model.fullFIASCode.HouseFIASCode})")

+ " adresses was added successfully!";

}

public bool TryAddNewAdress(int idMC, AdressModel model)

{

var locationsIDList = new List<int>();

using (var reader = sqlDriver.ExecReader

($"SELECT id FROM Locations WHERE idMC = {idMC}"))

{

while (reader.Read())

{

locationsIDList.Add((int)reader[0]);

}

}

var adressList = new List<AdressModel>();

foreach (var id in locationsIDList)

{

adressList.Clear();

using (var reader = sqlDriver.ExecReader($"SELECT \* FROM Adresses "

+ $"WHERE idLocation = {id}"))

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

{

while (reader.Read())

{

int house = -1;

if (reader["houseFIASCode"] != DBNull.Value)

house = (int)reader["houseFIASCode"];

adressList.Add(new AdressModel()

{

LocationID = (int)reader["idLocation"],

fullFIASCode = new FIASCode

{

RegionFIASCode = reader["regionFIASCode"].ToString(),

CityFIASCode = reader["cityFIASCode"].ToString(),

StreetFIASCode = reader["streetFIASCode"].ToString(),

HouseFIASCode = house

}

});

}

}

foreach (var adress in adressList)

{

if (model.fullFIASCode.RegionFIASCode

== adress.fullFIASCode.RegionFIASCode)

return false;

if (model.fullFIASCode.CityFIASCode != string.Empty

&& model.fullFIASCode.CityFIASCode

== adress.fullFIASCode.CityFIASCode)

return false;

if (model.fullFIASCode.StreetFIASCode != string.Empty

&& model.fullFIASCode.StreetFIASCode

== adress.fullFIASCode.StreetFIASCode)

return false;

if (model.fullFIASCode.HouseFIASCode != -1

&& model.fullFIASCode.HouseFIASCode

== adress.fullFIASCode.HouseFIASCode)

return false;

}

}

return true;

}

public string DeleteAdressByID(int id)

{

return sqlDriver.ExecNonQuery($"DELETE FROM Adresses WHERE id = {id}")

+ " adresses was deleted successfully!";

}

public AdressModel GetAdressByID(int id)

{

var adressInfo = sqlDriver.ExecReader($"SELECT \* FROM Adresses WHERE id = {id}");

adressInfo.Read();

return new AdressModel()

{

LocationID = (int)adressInfo["idLocation"],

fullFIASCode = new FIASCode

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

{

RegionFIASCode = adressInfo["regionFIASCode"].ToString(),

CityFIASCode = adressInfo["cityFIASCode"].ToString(),

StreetFIASCode = adressInfo["streetFIASCode"].ToString(),

HouseFIASCode = (int)adressInfo["houseFIASCode"]

}

};

}

public AdressModel ChangeAdressByID(int id, AdressModel newModel)

{

sqlDriver.ExecNonQuery("UPDATE Adresses\n"

+ $"SET idLocation = {newModel.LocationID},\n"

+ $"streetFIASCode = N'{newModel.fullFIASCode.StreetFIASCode}',\n"

+ $"cityFIASCode = N'{newModel.fullFIASCode.CityFIASCode}',\n"

+ $"regionFIASCode = N'{newModel.fullFIASCode.RegionFIASCode}',\n"

+ $"houseFIASCode = {newModel.fullFIASCode.HouseFIASCode}\n"

+ $"WHERE id = {id}");

return newModel;

}

}

public class LocationService

{

private readonly SqlDriver sqlDriver;

public LocationService(SqlDriver sqlDriver = null)

{

this.sqlDriver = sqlDriver;

this.sqlDriver.Open();

}

~LocationService()

{

sqlDriver.Close();

}

public LocationModel AddNewLocation(LocationModel model)

{

sqlDriver.ExecNonQuery("INSERT INTO Locations (name, idMC)"+

$"VALUES (N'{model.Name}', {model.CompanyID})");

return model;

}

public LocationModel GetLocationByID(int id)

{

var locationInfo = sqlDriver.ExecReader($"SELECT name, idMC FROM Locations WHERE id = {id}");

locationInfo.Read();

return(new LocationModel

{

Name = locationInfo[0].ToString(),

CompanyID = (int)locationInfo[1]

});

}

public string GetMCNameByLocationID(int id)

{

return "SELECT name FROM ManagementCompanies " +

$"WHERE id = (SELECT idMC FROM Locations WHERE id = {id})";

}

public string DeleteLocationByID(int id)

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

{

return sqlDriver.ExecNonQuery($"DELETE FROM Locations WHERE id = {id}")

+ " adresses was deleted successfully!";

}

public LocationModel ChangeLocationByID(int id, LocationModel newModel)

{

sqlDriver.ExecNonQuery("UPDATE Locations\n"

+ $"SET name = N'{newModel.Name}',\n"

+ $"idMC = {newModel.CompanyID}\n"

+ $"WHERE id={id}");

return newModel;

}

}

public class PriceService

{

private readonly SqlDriver sqlDriver;

public PriceService(SqlDriver sqlDriver = null)

{

this.sqlDriver = sqlDriver;

this.sqlDriver.Open();

}

~PriceService()

{

sqlDriver.Close();

}

public string AddNewPrice(PriceModel newModel)

{

int isRequest = Convert.ToInt32(newModel.isPriceOnRequest);

return sqlDriver.ExecNonQuery("INSERT INTO Prices (price, ifPriceOnRequest, idLocation, idService, name) "

+ $"VALUES ({newModel.Price}, {isRequest}, {newModel.LocationID}, "

+ $"{newModel.ServiceID}, N'{newModel.Name}')")

+ " adresses was added successfully!";

}

public string DeletePriceByID(int id)

{

return sqlDriver.ExecNonQuery($"DELETE FROM Prices WHERE id = {id}")

+ " adresses was deleted successfully!";

}

public PriceModel ChangePriceByID(int id, PriceModel newModel)

{

sqlDriver.ExecNonQuery("UPDATE Prices\n"

+ $"SET price = {newModel.Price},\nifPriceOnRequest = {newModel.isPriceOnRequest},"

+ $"\nidLocation = {newModel.LocationID},\n"

+ $"idService = {newModel.ServiceID},\n"

+ $"name = N'{newModel.Name}'\n"

+ $"WHERE id={id}");

return newModel;

}

public PriceModel GetPriceByID(int id)

{

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

var priceInfo = sqlDriver.ExecReader("SELECT price, ifPriceOnRequest, idLocation, idService "

+ $"FROM Prices WHERE id = {id}");

priceInfo.Read();

int serviceId = (int)priceInfo[3];

var model = new PriceModel

{

Name = string.Empty,

Price = (int)priceInfo[0],

isPriceOnRequest = (bool)priceInfo[1],

LocationID = (int)priceInfo[2],

ServiceID = serviceId

};

sqlDriver.Close();

sqlDriver.Open();

var serviceInfo = sqlDriver.ExecReader($"SELECT name, minCount, maxCount, measure, idMC "

+ $"FROM Services WHERE id = {serviceId}");

serviceInfo.Read();

model.Name = serviceInfo["name"].ToString();

return model;

}

public PriceModel GetPriceByCode(int codeFias, int idService)

{

var priceInfo = sqlDriver.ExecReader("SELECT price FROM Prices\n"

+ $"WHERE idLocation = {codeFias} "

+ $"AND idService = {idService}");

priceInfo.Read();

var serviceInfo = sqlDriver.ExecReader($"SELECT \*"

+ $"FROM Services WHERE id = {idService}");

serviceInfo.Read();

return new PriceModel

{

Name = serviceInfo["name"].ToString(),

Price = (int)priceInfo[0],

isPriceOnRequest = (bool)priceInfo[1],

LocationID = (int)priceInfo[2],

ServiceID = idService

};

}

}

public class UserService

{

private readonly SqlDriver sqlDriver;

public UserService(SqlDriver sqlDriver = null)

{

this.sqlDriver = sqlDriver;

this.sqlDriver.Open();

}

~UserService()

{

sqlDriver.Close();

}

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

public string AddNewUser(UserModel newModel)

{

var passwordHash = GetHashedKey(newModel.Password);

return sqlDriver.ExecNonQuery("INSERT INTO Users"

+ "(email, login, passwordHash, roleCode)"

+ $"VALUES(N'{newModel.Email}', N'{newModel.Login}', N'{passwordHash}', 1)")

+ " users was added successfully!";

}

public string DeleteUserByID(int id)

{

return sqlDriver.ExecNonQuery($"DELETE FROM Users WHERE id = {id}")

+ " users was deleted successfully!";

}

public UserModel ChangeUserByID(int id, UserModel newModel)

{

//хэширование нового пароля для хранения хэша в БД

var passwordHash = GetHashedKey(newModel.Password);

sqlDriver.ExecNonQuery($"UPDATE Users\n"

+ $"SET email = N'{newModel.Email}',\n"

+ $"login = {newModel.Login},\n"

+ $"passwordHash = {passwordHash},\n"

+ $"roleCode = N'{newModel.Role}'\n"

+ $"WHERE id = {id}");

return newModel;

}

public UserModel GetUserByID(int id)

{

var userInfo = sqlDriver.ExecReader($"SELECT \* "

+ $"FROM Users WHERE id = {id}");

userInfo.Read();

return new UserModel

{

Email = userInfo["email"].ToString(),

Login = userInfo["login"].ToString(),

Password = userInfo["passwordHash"].ToString(),

Role = (UserModel.UserRole)userInfo["roleCode"],

};

}

static string GetHashedKey(string input)

{

using (MD5 md5Hash = MD5.Create())

{

byte[] data = md5Hash.ComputeHash(Encoding.UTF8.GetBytes(input));

StringBuilder result = new StringBuilder();

foreach (byte b in data)

result.Append(b.ToString("X2"));

return result.ToString();

}

}

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

}

public class WorkService

{

private readonly SqlDriver sqlDriver;

public WorkService(SqlDriver sqlDriver = null)

{

this.sqlDriver = sqlDriver;

this.sqlDriver.Open();

}

~WorkService()

{

sqlDriver.Close();

}

public string AddNewService(ServiceModel model)

{

return sqlDriver.ExecNonQuery("INSERT INTO Services (name, minCount, maxCount, measure, idMC) "

+ $"VALUES (N'{model.Name}', {model.MinCount}, {model.MaxCount}, N'{model.Measurement}', "

+ $"'{model.CompanyID}')")

+ " services was added successfully!";

}

public string DeleteServiceByID(int id)

{

return sqlDriver.ExecNonQuery($"DELETE FROM Services WHERE id = {id}")

+ " services was deleted successfully!";

}

public ServiceModel ChangeServiceByID(int id, ServiceModel newModel)

{

sqlDriver.ExecNonQuery($"UPDATE Services\n"

+ $"SET name = N'{newModel.Name}',\n"

+ $"minCount = {newModel.MinCount},\n"

+ $"maxCount = {newModel.MaxCount},\n"

+ $"measure = N'{newModel.Measurement}',\n"

+ $"idMC = {newModel.CompanyID}\n"

+ $"WHERE id = {id}");

return newModel;

}

public ServiceModel GetServiceByID(int id)

{

var serviceInfo = sqlDriver.ExecReader($"SELECT name, minCount, maxCount, measure, idMC "

+ $"FROM Services WHERE id = {id}");

serviceInfo.Read();

return new ServiceModel

{

Name = serviceInfo["name"].ToString(),

CompanyID = (int)serviceInfo["idMC"],

MaxCount = (int)serviceInfo["maxCount"],

MinCount = (int)serviceInfo["minCount"],

Measurement = serviceInfo["measure"].ToString()

};

}

}

}

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

namespace MCService.Web

{

[Route("api/[controller]")]

[ApiController]

public class AdressController : ControllerBase

{

private readonly AdressService adressService;

public AdressController(AdressService adressService = null)

{

this.adressService = adressService;

}

[HttpGet("{id:int}")]

public ActionResult GetAdressByID(int id) => Ok(adressService.GetAdressByID(id));

[HttpPost]

public ActionResult AddNewFullAdress(int idMC, AdressModel model)

{

if (adressService.TryAddNewAdress(idMC, model))

return Ok(adressService.AddNewAdress(model));

else

return BadRequest();

}

[HttpPut("{id:int}", Name = "UpdateAdressByID")]

public ActionResult UpdateLocationByID(int id, AdressModel newModel)

{

return Ok(adressService.ChangeAdressByID(id, newModel));

}

[HttpDelete("{id:int}", Name = "DeleteAdressByID")]

public ActionResult DeleteLocationByID(int id)

{

return Ok(adressService.DeleteAdressByID(id));

}

}

[Route("api/[controller]")]

[ApiController]

public class LocationController : ControllerBase

{

private readonly LocationService locationService;

public LocationController(LocationService locationService = null)

{

this.locationService = locationService;

}

[HttpGet("{id:int}", Name = "GetLocationByID")]

public ActionResult GetLocationByID(int id)

{

return Ok(locationService.GetLocationByID(id));

}

[HttpPost(Name = "AddNewLocation")]

public ActionResult AddNewLocation(LocationModel model)

{

return Ok(locationService.AddNewLocation(model));

}

[HttpPut(Name = "UpdateLocationByID")]

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

public ActionResult UpdateLocationByID(int id, LocationModel newModel)

{

return Ok(locationService.ChangeLocationByID(id, newModel));

}

[HttpDelete("{id:int}", Name = "DeleteLocationByID")]

public ActionResult DeleteLocationByID(int id)

{

return Ok(locationService.DeleteLocationByID(id));

}

}

[Route("api/[controller]")]

[ApiController]

public class PriceController : ControllerBase

{

private readonly PriceService priceService;

public PriceController(PriceService priceService = null,

WorkService workService = null)

{

this.priceService = priceService;

}

[HttpGet("{id:int}", Name = "GetPriceByID")]

public ActionResult GetPriceByID(int id)

{

return Ok(priceService.GetPriceByID(id));

}

[HttpGet(Name = "GetPriceByCode")]

public ActionResult GetPriceByCode(int houseFIASCode, int idService)

{

return Ok(priceService.GetPriceByCode(houseFIASCode, idService));

}

[HttpPost(Name = "AddNewPrice")]

public ActionResult AddNewPrice(PriceModel newModel)

{

return Ok(priceService.AddNewPrice(newModel));

}

[HttpPut(Name = "UpdatePriiceByID")]

public ActionResult UpdatePriiceByID(int id, PriceModel newModel)

{

return Ok(priceService.ChangePriceByID(id, newModel));

}

[HttpDelete("{id:int}", Name = "DeletePriceByID")]

public ActionResult DeletePriceByID(int id)

{

return Ok(priceService.DeletePriceByID(id));

}

[Route("api/[controller]")]

[ApiController]

public class UserController : ControllerBase

{

private readonly UserService userService;

public UserController(UserService userService = null)

{

this.userService = userService;

}

[HttpGet("{id:int}", Name = "GetUserByID")]

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

public ActionResult GetUserByID(int id)

{

return Ok(userService.GetUserByID(id));

}

[HttpPost(Name = "AddNewUser")]

public ActionResult AddNewService(UserModel model)

{

return Ok(userService.AddNewUser(model));

}

[HttpPut(Name = "UpdateUserByID")]

public ActionResult UpdateServiceByID(int id, UserModel newModel)

{

return Ok(userService.ChangeUserByID(id, newModel));

}

[HttpDelete("{id:int}", Name = "DeleteUserByID")]

public ActionResult DeleteServiceByID(int id)

{

return Ok(userService.DeleteUserByID(id));

}

}

[Route("api/[controller]")]

[ApiController]

public class WorkController : ControllerBase

{

private readonly WorkService workService;

public WorkController(WorkService workService = null)

{

this.workService = workService;

}

[HttpGet("{id:int}", Name = "GetServiceByID")]

public ActionResult GetServiceByID(int id) => Ok(workService.GetServiceByID(id));

[HttpPost(Name = "AddNewService")]

public ActionResult AddNewService(ServiceModel model) =>  
Ok(workService.AddNewService(model));

[HttpPut(Name = "UpdateServiceByID")]

public ActionResult UpdateServiceByID(int id, ServiceModel model) => Ok(workService.ChangeServiceByID(id, model));

[HttpDelete("{id:int}", Name = "DeleteServiceByID")]

public ActionResult DeleteServiceByID(int id) =>

Ok(workService.DeleteServiceByID(id));

}

}

using MCService.Services;

using MCService.Database;

using MCService.Web.Controllers;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

string connection = builder.Configuration.GetConnectionString("MSSQL");

builder.Services.AddControllers();

builder.Services.AddScoped<AdressService>();

Листинг 2 – Код сервиса для работы с базой данных (продолжение)

builder.Services.AddScoped<LocationService>();

builder.Services.AddScoped<WorkService>();

builder.Services.AddScoped<PriceService>();

builder.Services.AddScoped<UserService>();

builder.Services.AddScoped<SqlDriver>();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();