Министерство образования и науки Нижегородской области

Государственное бюджетное профессиональное образовательное учреждение «Нижегородский Губернский колледж»

Допущен к защите:

преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Романова

\_\_\_\_\_\_\_\_\_\_\_\_ Ю.С. Мамшева

«03» июля 2024 г.

**ОТЧЕТ**

**ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

ПМ.02 ОСУЩЕСТВЛЕНИЕ ИНЕТЕГРАЦИИ ПРОГРАММНЫХ МОДУЛЕЙ

Руководитель

от предприятия: \_\_\_\_\_\_\_\_\_\_Шмелёва Е.С. 03.07.2024г.

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Чернов С.В. 03.07.2024г.

Специальность, группа: 09.02.07, 33П

Нижний Новгород

2024г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc170861317)

[1. ТЕОРИТИЧЕСКАЯ ЧАСТЬ 6](#_Toc170861318)

[1.1. Организационная структура предприятия 6](#_Toc170861319)

[1.2. Анализ программного обеспечения и технического оснащения 8](#_Toc170861322)

[1.3. Описание инструментальных средств разработки и сопровождения программных средств. 9](#_Toc170861326)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 11](#_Toc170861327)

[2.1. Анализ предметной области 11](#_Toc170861328)

[2.2. Проектирование программного продукта 14](#_Toc170861331)

[2.3. Разработка программного модуля 23](#_Toc170861341)

[2.4. Тестирование программного продукта 26](#_Toc170861344)

[ЗАКЛЮЧЕНИЕ 27](#_Toc170861345)

[СПИСОК ИСПОЛЬЗОВАНОЙ ЛИТЕРАТУРЫ 28](#_Toc170861346)

[ПРИЛОЖЕНИЯ 30](#_Toc170861347)

[Приложение А 30](#_Toc170861348)

[Приложение Б 31](#_Toc170861349)

[Приложение В 32](#_Toc170861350)

[Приложение Г 35](#_Toc170861351)

[Приложение Д 36](#_Toc170861352)

[Приложение Ж 42](#_Toc170861357)

[Приложение З 44](#_Toc170861358)

[Приложение И 47](#_Toc170861363)

[Приложение К 47](#_Toc170861364)

**ВВЕДЕНИЕ**

Производственная практика по ПМ.02 «Осуществление интеграции программных модулей» проходящая в организации АО «НЗ 70-летия Победы», предоставляет возможность применить знания, полученные в ходе обучения и учебной практики в реальном производственном процессе, а именно выполнения задач, поставленных руководителем практики от предприятия.

Целью производственной практики является, вышеупомянутое, применение знаний на практике посредством создания модуля продукта необходимого предприятию, который будет полу автоматизировать работу с заявками на исправление каких-либо нарушений работ систем или на улучшения в предприятии от его работников и необходимой документацией для составляемых заявок.

Установленная цель, при выполнении, решает проблему, которой, в данном случае, является создание модуля программного интерфейса приложения (API), которое будет встроено в общую систему в соответствии с требованиями заказчика.

Практическим результатом выполнения задания производственной практики является API для взаимодействия с базой данных и документами, созданный для дальнейшей разработки пользовательского интерфейса сторонними разработчиками.

API (англ. «Application Programming Interface» — программный интерфейс приложения) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными. Он встречается практически во всех проектах, связанных с работой с пользовательскими данными как общедоступными, так и закрытыми, поскольку позволяет разработчикам связывать разные программные модули и программы между собой, делая связь проще и безопаснее.

Для достижения цели были поставлены следующие задачи:

* повторить материал, пройденный на учебной практике;
* получить требования к разрабатываемому модулю;
* описать структуру системы в соответствии с требованиями к разрабатываемому модулю;
* описать структуру взаимодействия системы с пользователем;
* разработать модуль информационной системы;
* создать тестовую документацию и протестировать разработанный модуль информационной системы;
* провести защиту презентации продукта.

Основными методами решения задач будут являться: логический, практический, комбинаторный.

1. **ТЕОРИТИЧЕСКАЯ ЧАСТЬ**
   1. **Организационная структура предприятия**
      1. Концепция предприятия

Нижегородский завод 70-летия Победы, предприятие оборонной промышленности в Нижнем Новгороде.

Создан в 2014 г. на базе существовавшего с 1932 г. Нижегородского машиностроительного завода (специализация – ракетные комплексы, артиллерийские системы, атомные корабельные установки), входит в структуру Концерна воздушно-космической обороны «Алмаз-Антей» со штаб-квартирой в Москве.

Значимое машиностроительное предприятие Нижнего Новгорода (свыше 5 тыс. занятых в 2021), является одним из крупнейших производителей перспективного вооружения и военной техники, обеспечивающих выполнение государственного оборонного заказа.

Дополнительные специализации гражданского назначения – карьерная и сельскохозяйственная техника, крановое оборудование и системы хранения.

Основным заказчиком являются вооружённые силы России и другие государственные военно-исследовательские и сельскохозяйственные предприятия.

Лозунг предприятия: «Мирное небо – наша работа!».

* + 1. Описание структуры предприятия

Поскольку предприятие закрытое, то описать можно только структуру отдела, где проходила практика.

* + - 1. Деятельность отдела прохождения практики

Деятельностью отдела прохождения практики являются:

* разработка, внедрение, настройка и сопровождение информационных систем на основе экономико-математических методов, современных средств вычислительной техники, коммуникации и связи, элементов теории экономической кибернетики;
* исследование подсистем систем с целью определения возможности их формализации и целесообразности перевода соответствующих процессов на автоматический режим;
* подготовка планов проектирования и внедрения подсистем на предприятие и контроль их выполнения;
* организация работы по совершенствованию электронного документооборота по процессам соответствующих подсистем;
* осуществление организационного и технического взаимодействия подсистем предприятия;
* проектирование и разработка технологических схем обработки информации по всем задачам предприятия и технологических процессов обработки информации при помощи средств вычислительной техники;
* обеспечение бесперебойного функционирования информационных систем предприятия на основе анализа и изучения проблем обслуживания и принятие оперативных мер по устранению, возникающих в процессе работы нарушений.
* разработка инструкций, методических и нормативных материалов, связанных с информационным обеспечением подсистем предприятия.
  + - 1. Структура отдела и должности сотрудников

Отдел информационных систем (ОИС) является структурным подразделением управления информационных технологий (УИТ) и подчиняется начальнику управления информационных технологий.

ОИС возглавляет начальник отдела информационных систем, подчиняющийся начальнику УИТ.

Начальнику ОИС в установленном порядке подчиняется заместитель начальника отдела и руководители групп и бюро.

В соответствии с направлениями деятельности и выполняемыми задачами структура отдела следующая (наглядно представлена на рисунке А1):

* группа управления производством и развития;
* группа управления ресурсами предприятия;
* группа нормативно-справочной информации;
* группа сопровождения информационных систем;
* бюро программного обеспечения собственной разработки.

На время отсутствия начальника отдела, руководство отделом осуществляется работником, назначенным в установленном порядке.

ОИС взаимодействует со всеми структурными подразделениями предприятия, осуществляя сопроводительную и управленческую деятельности над информационными системами.

* 1. **Анализ программного обеспечения и технического оснащения**
     1. Системное обеспечение

Для решения задач, поставленных на предприятии, использовались компьютеры со следующими характеристиками:

* процессор: Intel(R) Core (TM) i7-6700 CPU @ 3.40GHz;
* оперативная память: 32.0 Гб, 2133 MHz (неизвестный производитель);
* твердотельный накопитель: SSD SPCC M.2 PCIe 477 Гб;
* видеокарта: NVIDIA Quadro M400 8 Гб;
* сетевая карта: Intel(R) Ethernet Connection (2) I219-V;
* мониторы: 24-27” 1980\*1080p 1-3 шт на рабочее место.

Данные характеристики являются средними для всех персональных компьютеров предприятия.

Также, на предприятии присутствуют принтеры и МФУ (многофункциональное устройство) (модели неизвестны), источники бесперебойного питания (APC) и IP-телефоны (модели неизвестны).

* + 1. Программное обеспечение

Основные виды программного обеспечения, которое используют сотрудники предприятия:

* Windows 10 Pro x 64;
* Kaspersky endpoint security;
* 7-zip;
* Adobe Acrobat;
* IPS;
* PDF24;
* SAP;
* Tessa;
* Yandex browser;
* 1С (различные конфигурации);
* Visual Studio;
* Microsoft SQL Server 2016;
* Microsoft SQL Server Manager 2016;
* Microsoft Office 2013;
* Git Bash.
  + 1. Топология сети предприятия

Топология всей сети предприятия состоит из следующих элементов:

* Звезда;
* Кольцо;
* Последовательная.

Но поскольку предприятие является военно-направленным, схему реальной топологии получить возможности нет, была составлена примерная топология сети информационного отдела предприятия (рисунок 1Б).

Совокупность программного и технического обеспечения удовлетворяет необходимым требованиям для решения поставленной задачи, поскольку техническая составляющая соответствует требованиям программной части.

* 1. **Описание инструментальных средств разработки и сопровождения программных средств.**

Инструментальными средствами разработки модуля программы являются следующие инструменты:

* Visual Studio 2022 – интегрированная среда разработки;
* Microsoft SQL Server 2016 – локальный сервер базы данных;
* Microsoft SQL Server Manager 2016 – система управления базами данных;
* Microsoft Office 2013 – набор программ для документирования и оформления отчётности;
* Git – система контроля версий;
* GitHub – удалённое хранилище репозиториев.

Инструментальные средства предприятия удовлетворяют всем необходимым потребностям для разработки модуля программного продукта и его сопровождения.

1. **ПРАКТИЧЕСКАЯ ЧАСТЬ**
   1. **Анализ предметной области**
      1. Анализ требований и информационных потребностей

Требования к разрабатываемому модулю представлены в виде требований к базе данных и реализации API.

* + - 1. Анализ требований пользователей к базе данных

Требования пользователей к базе данных предоставлена заказчиком в виде предметной области, описанной далее.

База данных должна содержать данные пользователей, необходимые для хода и регистрации, список заявок, а также ссылки (пути) на документы, необходимые для составления заявки.

Пользователь может входить в систему и выполнять действия в соответствии со своей ролью.

Обычный пользователь может создать заявку на решение какой-либо проблемы и прикрепить к данной заявке документы, оформленные с его стороны.

Администратор может зарегистрировать пользователя и выдать ему роль, а также обработать заявки от пользователя.

Управляющий должен изменить документы из заявки пользователя (изменить запись), заполнив документ с данными со своей стороны и прикрепить его обратно к заявке.

* + - 1. Анализ требований пользователей к API

Обращение со стороны клиентского приложения (сайта, программы, мобильного приложения) к базе данных происходит с помощью API, где реализованы следующие запросы к базе данных:

* регистрация пользователя администратором;
* авторизация пользователя в системе;
* изменение данных пользователя администратором;
* предоставление данных о пользователях;
* удаление пользователя администратором;
* создание заявки обычным пользователем;
* изменение заявки обычным пользователем;
* удаление заявки;
* изменение документа из заявки пользователем с ролью руководителя;
* изменение статуса заявки на выполненную исполнителем;
* предоставление документации для составления заявки;
* предоставление информации о заявках в соответствии с фильтрацией по выполнению, подтверждению, пользователям в разных ролях (создатель, подтверждающая сторона, исполнитель), дате и подобному.

Администратор владеет всеми правами и привилегиями относительно работы с данными API.

API должна быть легко масштабируема, а её функционал задокументирован с помощью «Swagger».

* + - 1. Составление функциональных требований
         1. Сущности базы данных

Судя по информации предметной области, можно сказать, что база данных для разрабатываемой информационной системы должна состоять из следующих таблиц:

* роли (описание ролей);
* пользователи (список данных пользователей для входа в систему);
* заявки (список заявок пользователей);
* статусы (описание статусов заявки);
* документы (описание документов в системе).

Вышеуказанные таблицы в дальнейшем будут связаны с клиентской стороной системы посредством какого-либо API.

* + - * 1. Задачи информационной системы

Основными задачами системы, которые она должна выполнять, являются:

* заполнение базы данных новыми данными (новые студенты, берущие книги, новые книги и т.п.);
* изменение информации в базе данных;
* удаление данных в базе данных;
* разграничение ролей пользователей системы.
  + - * 1. Выявление групп пользователей и их задач

Основными пользователями данной информационной системы будут:

* администратор информационной системы;
* простой пользователь;
* руководитель;
* исполнитель.

Администратор ИС – уровень доступа пользователя, который имеет доступ ко всем данным и привилегиям данной БД, поскольку его задачей будет её администрирование.

Простой пользователь – уровень доступа пользователя, который может использовать данные базы данных для составления заявки на какую-то услугу со стороны исполнителя, изменять данные заявки и удалить её.

Руководитель – уровень доступа пользователя, который может изменить документ, предоставленный для заполнения обычным пользователем, тем самым изменив статус заявки на «одобрено» и просмотреть данные пользователей, которые отправляют заявки.

Исполнитель – уровень доступа пользователя, который может изменить статус заявки на «в процессе» и «выполнена».

* + 1. Выявление информационных объектов и связей между ними

В разрабатываемой информационной системе основными объектами будут таблицы содержащие данные ролях, пользователях, заявках, статусах и документах структурированные посредством связей один к одному (1:1), один ко многим (1:М) и многие ко многим (М:М).

Структура базы данных информационной системы указана в таблицах с 1-ой по 7-мую в приложении «Приложение В», а схема самой базы данных на рисунке 1Г.

Так же, для создания указанного модуля информационной системы было создано техническое задание, описанное далее в приложении Д.Техническое задание.

* 1. **Проектирование программного продукта**

Полное отображение функциональной составляющей разрабатываемой информационной системы представлено виде UML-диаграмм, которые будут представлены далее.

Так же стоит отметить, что взаимодействие с разрабатываемым модулем информационной системы происходит с помощью интерфейса «клиентской стороны» (интерфейс, с которым взаимодействует пользователь), который будет реализован другими разработчиками.

* + 1. Диаграмма использования

Диаграмма использования (рисунок 1) описывает варианты использования системы обработки заявлений на решение проблем производства.

На данной диаграмме пользователь (уже авторизированный как обычный работник), может создавать заявки на решение каких-либо проблем, в тоже время пользователь с ролью «Начальник» может одобрить данную заявку или как-либо откорректировать, а «Мастер» «обработать заявление» устранив неполадку и отметить то, что, заявка «закрыта» (проблема, описанная в заявке решена).

Администратор в свою очередь может осуществлять полную управленческую деятельность в системе, манипулируя профилями, заявками и документами в системе.

Каждый из вышеперечисленных пользователей наследует функционал простого работника.

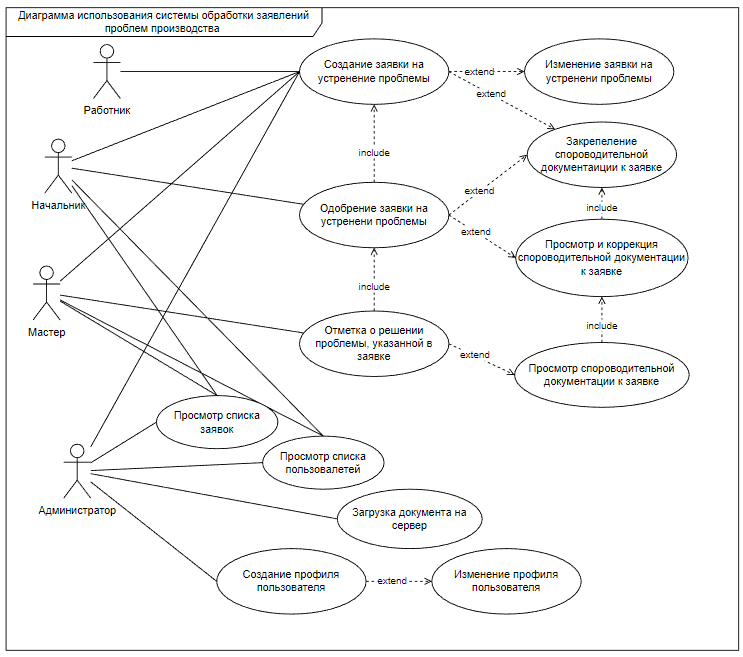


Рисунок 1. Диаграмма использования системы обработки заявлений на решение проблем производства

* + 1. Диаграмма последовательности

Диаграмма последовательности (рисунок 2) отражает порядок действий, которые могут совершать пользователи по отношению к системе.

Для пользования создания заявки, уже авторизованный пользователь должен заполнить необходимую форму, которая передаст данные в API, который проверит эти данные на корректность и создаст запрос для базы данных, которая запишет эти данные и вернёт подтверждение записи.

Если данные пользователь ввёл не корректно, то API возвращает сообщение о некорректности данных.

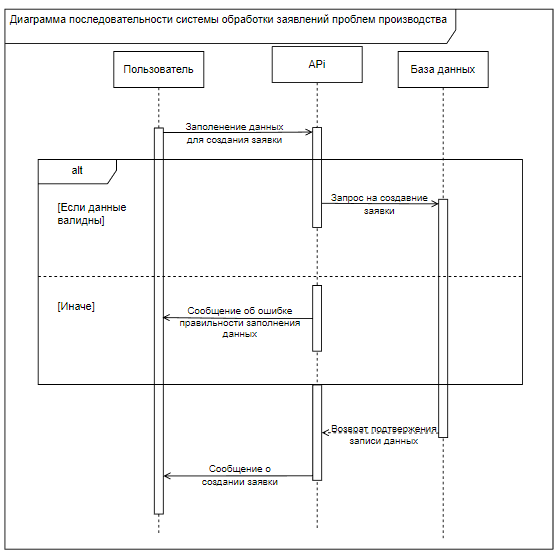


Рисунок 2. Диаграмма последовательности системы обработки заявлений на решение проблем производства

* + 1. Диаграмма классов

Диаграмма классов (рисунок 3) описывает классы и их взаимосвязи в системе, что отражает структуру системы.

На данной диаграмме изображены основные сущности базы данных, с которыми пользователь может взаимодействовать с помощью интерфейса.

Данными сущностями являются:

«User» (пользователь), которая реализует одноимённую таблицу в базе данных (с помощью свойств), а также методы авторизации и манипулирования данными пользователя, а также различные проверки на роль пользователя в системе;

«Role» (роль), которая реализует одноимённую таблицу в базе данных (с помощью свойств) и расширяет сущность «User»;

«Appointment» (заявка), которая реализует одноимённую таблицу в базе данных (с помощью свойств) и ассоциативно связана с пользователем («User»), который может манипулировать заявлением;

«Status» (статус), которая реализует одноимённую таблицу в базе данных (с помощью свойств) и расширяет сущность «Appointment»;

«Document» (документ), которая реализует одноимённую таблицу в базе данных (с помощью свойств) и имеет методы манипуляции файлами (документами) в системе.

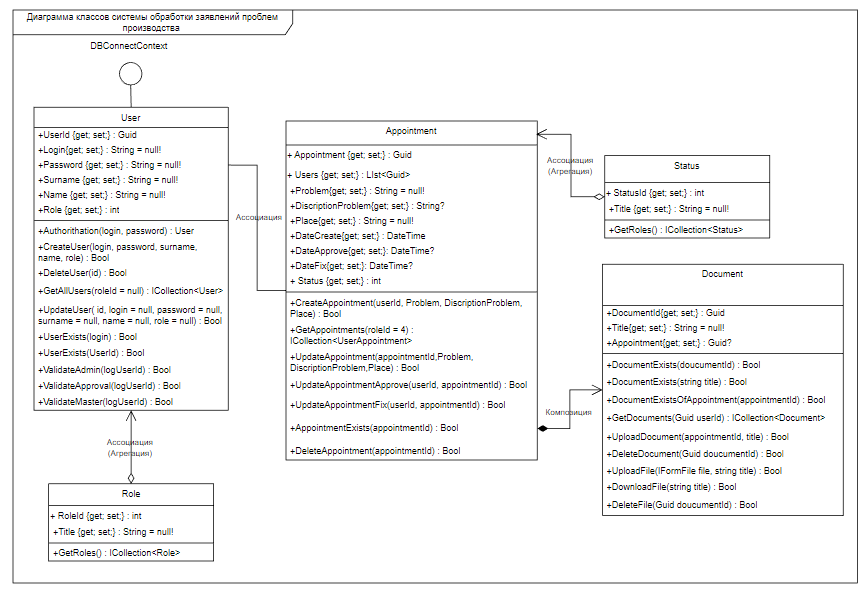


Рисунок 3. Диаграмма классов системы обработки заявлений на решение проблем производства

* + 1. Диаграмма коммуникации

Диаграмма коммуникации (рисунок 4) иллюстрирует работу системы со стороны «общения» её модулей.

На данной диаграмме пользователь через интерфейс клиентской части системы заполняет данные для авторизации в системе, после подтверждения которой может создать заявку на устранение какой-либо проблемы, посредством интерфейса того же приложения.

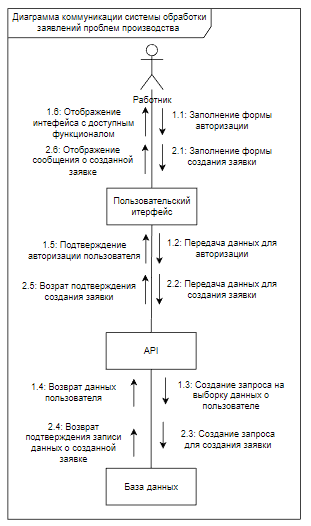


Рисунок 4. Диаграмма коммуникации системы обработки заявлений на решение проблем производства

* + 1. Диаграмма деятельности

Диаграмма деятельности (рисунок 5) описывает поведение системы при попытке пользователя создать заявку на устранение какой-либо проломы.

На данной диаграмме пользователь заполняет форму необходимыми данными, после чего система обрабатывает данный запрос и в зависимости от правильности заполнения данных предлагает пользователю такие действия как завершить операцию или проверить и откорректировать данные, после чего повторить запрос снова.

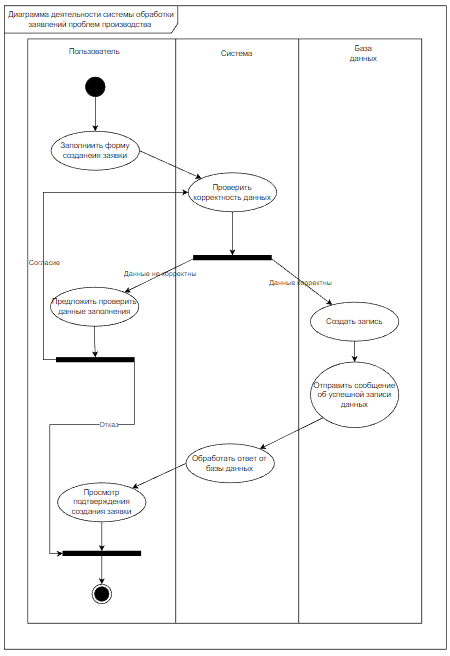


Рисунок 5. Диаграмма деятельности системы обработки заявлений на решение проблем производства

* + 1. Диаграмма состояний

Диаграмма состояний (рисунок 6) отражает реализацию запроса на создание заявки на устранение проблемы.

В начальном состоянии система ожидает данные для создания заявки от пользователя, а после их получения создаёт запрос к базе данных.

Если запрос может быть обработан, то он переходит к ожидающей его базе данных, где данные записываются в необходимые таблицы.

Если данные запроса не корректны, то система возвращается в исходное состояние, сообщив пользователю о некорректности введённых им данных.

Если запись данных в базу данных прошла успешно, то пользователь получает сообщение об создании записи, после чего происходит возврат системы в исходное состояние.

Если же запрос на не может быть обработан, то пользователю выводится сообщение об ошибке, администраторам отправляется то же сообщение, а система прекращает свою работа (либо, если это предусмотрено системными администраторами, уходит в режим самовосстановления).

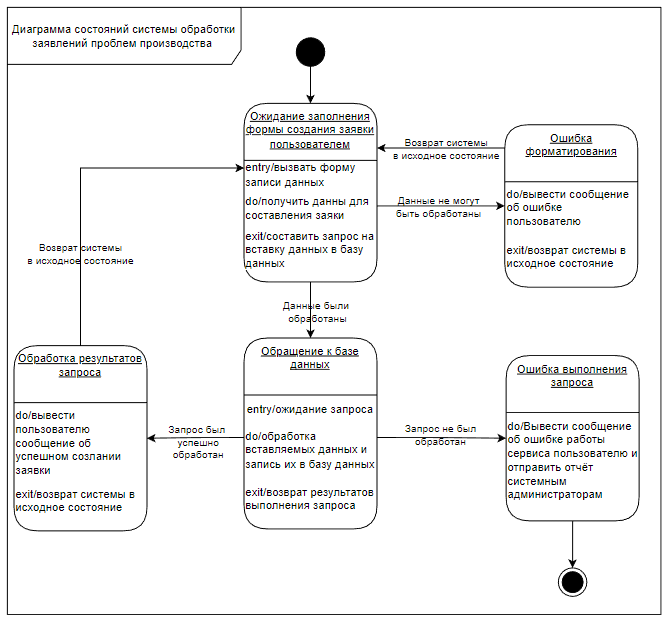


Рисунок 6. Диаграмма состояний системы обработки заявлений на решение проблем производства

* + 1. Диаграмма развёртывания

На диаграмме развёртывания (рисунок 7) показано, что пользователь, через интерфейс клиентского приложения на своём устройстве обращается посредством API может манипулировать данными в базе данных, которая общается с API посредством протокола TCP.

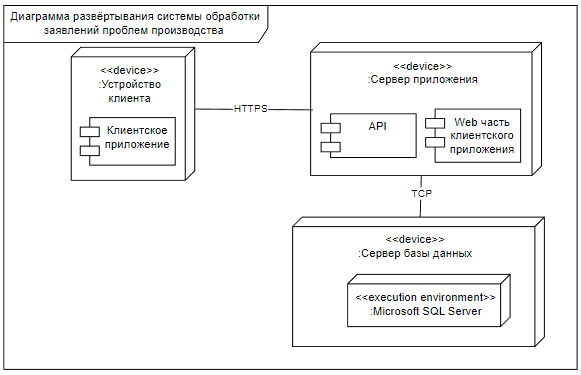


Рисунок 7. Диаграмма развёртывания системы обработки заявлений на решение проблем производства

* + 1. Диаграмма компонентов

Диаграмма компонентов (рисунок 8) описывает физическую структуру и зависимостей компонентов, в роли которых может выступать исходный, бинарный и исполняемый код, в системе.

Пользователь («Работник») через порт (делегирующий соединитель) обращается к компоненту-форме создания заявки, передавая в неё данные, которые с помощью различных интерфейсов (передавая данные по собираемому соединителю) обращается к компоненту «Appointment.cs» (контроллеру API для работы с заявками), который обрабатывает полученные данные и передаёт базе данных Microsoft SQL Server через делегирующий соединитель.

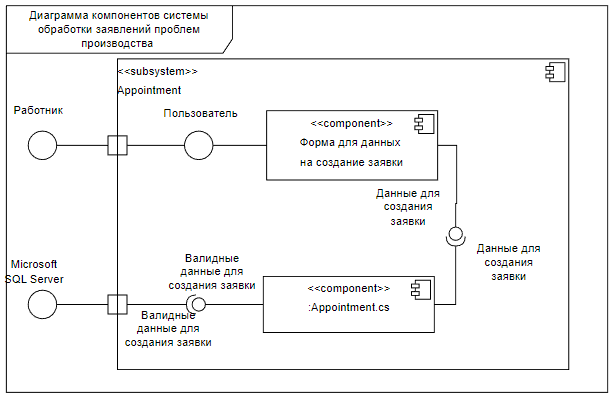


Рисунок 8. Диаграмма компонентов системы обработки заявлений на решение проблем производства

* + 1. Диаграмма потоков данных

Диаграмма потоков данных (рисунок 9) иллюстрирует процесс создания заявки на решение какой-либо проблемы, во время которого система предоставляет доступ к внешним источникам данных, потокам и хранилищам данных.

Работник заполняет форму в клиентском приложении данными, которые обрабатываются компьютером (сервером), запрос на создание заявки, данные которой записываются в базу данных. После записи данных пользователю возвращается сообщение об успешном создании заявки.

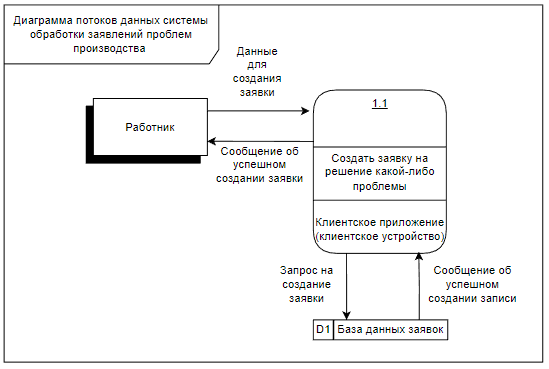


Рисунок 9. Диаграмма потоков данных системы обработки заявлений на решение проблем производства

* 1. **Разработка программного модуля**

Актуальность разрабатываемой информационной система для обработки заявок описана в пункте 2.1. Анализ предметной области.

Сама система состоит из клиентской части (frontend), которая связана с помощью программного интерфейса приложения (API) с базой данных.

* + 1. Задание на разработку части информационной системы

Необходимо разработать API, который при размещении на сервере связывает клиентскую часть системы с базой данных.

Основными задачами данного интерфейса являются манипуляция данными в базе данных в соответствии с требованиями, описанными в техническом задании.

* + 1. Процесс разработки

Основными компонентами, участвующими в разработке, являются:

* платформа .NET 7 (для языка программирования C#)
* ASP .NET Core Web Api приложение (основа для создания API);
* EntityFramevorkCore (готовое решение для управления базами данных с помощью программного кода C#);
* Microsoft.EntityFrameworkCore.SqlServer (компонент для EntityFramevorkCore, реализующий работу с базами данных на языке Microsoft Sql Server);
* Microsoft.EntityFrameworkCore.Design (вспомогательный компонент для работы с EntityFramevorkCore);
* AutoMapper (готовое решения для автоматического сопоставления типов данных различных структур);
* AutoMapper.Extensions.Microsoft.DependencyInjection (вспомогательный компонент для работы с AutoMapper, внедряющий само решение в программный продукт);
* xUnit Test Progect (проект для тестирования API),
* Swagger (Автоматическое средство документирования API).

После установки вышеперечисленных компонентов был создан проект ASP .NET Core Web Api, при создании которого была подтверждена работа с защищённым протоколом https и была создана внутренняя структура проекта из папок, в которых были помещены соответствующие файлы.

Структура проекта (результат показан на рисунке 1Ж):

* DTO (сторонние модели представления данных);
* Controllers (контроллеры);
* Interfaces (интерфейсы для методов и грамотного наследования);
* Mapping (файлы составления сопоставления сущностей для обработки данных);
* Models (модели базы данных, перенесённые непосредственно из созданной базы данных);
* Repositories (реализация методов взаимодействия с базой данных);
* Another (реализация дополнительного функционала – хеширование пароля);
* LocalFiles (локальные файлы, которые необходимы в работе API (документы, использующиеся в системе)).

После создания проекта, в соответчики с Er-диаграммой была создана база данных, из которой, с помощью команды «dotnet ef dbcontext scaffold "Server=localhost;Database=RequestHandler;Trusted\_Connection=True;Encrypt=False;" Microsoft.EntityFrameworkCore.SqlServer -f -o Models» были получены модели со связями (результат показан на рисунке 2Ж). На основе данных сущностей и происходит взаимодействие с базой данных, которое описано в методах классов, находящихся в папке «Repositories» (пример реализации методов для модели «User» можно увидеть в листинге [Таблица – 1З – UserRepository](#_Таблица_–_1)). Каждый класс из данной папки имеет свой интерфейс, который позволяет обращаться с помощью зависимости к реализации методов класса, который реализовал данный интерфейс.

Стоит отметить, что все методы, реализованы как асинхронные задачи, что позволяет API иметь высокую отказоустойчивость и быстроту обработки операций при работе с большим количеством данных или большим количеством запросов.

Поскольку в системе, для которой создавался программный модуль, предусматривается авторизация, все пароли в ней хешируюся (кодируются) по стандарту «SHA256» (реализация представлена в листинге [Таблица – 2З – Hashing](#_Таблица_–_2)).

Реализация API происходит через контроллеры, которые описаны в классах папки «Controllers». Каждый метод в данных классиках имеет свою ситуативную обработку корректности данных, которая соответствует специфики запроса.

Так же в методах работы с документацией были реализованы методы загрузки файлов на сервер (листинг [Таблица – 3З – UploadFile](#_Таблица_–_3)) и с сервера клиенту (листинг [Таблица – 4З – DownloadFile](#_Таблица_–_4)).

Весь функционал, который должен реализовать API описан в анализе предметной области в пункте 2.1. Анализ предметной области.

Данные методы автоматически документируются в «Swagger» (результат – рисунок 3Ж).

Информационная система загружена на систему контроля версий Gogs для наблюдения за изменениями в программе и возможности возвращения к предыдущим версиям.

* 1. **Тестирование программного продукта**

После разработки было проведено ручное и автоматизированное тестирование программного продукта, документация к которому описана в приложении И. Тестовая документация, а реализация автоматизированного тестирования (unit-тестов) в приложении К. Тест кейсы.

Для Unit тестирования API в инструментальной среде разработки Visual Studio был создан проект тестирования типа «xUnit», который позволяет тестировать web интерфесы, чем и является API.

В качестве тестовых данных для тестирования выступают временные «фейковые» данные, которые находятся в специальном файле в виде коллекции данных, похоже на таблицу в базе данных. Такой файл является «фейк репозиторием» и повторяет работу алгоритмов манипуляции данными в базе данных, не подключаясь к ней.

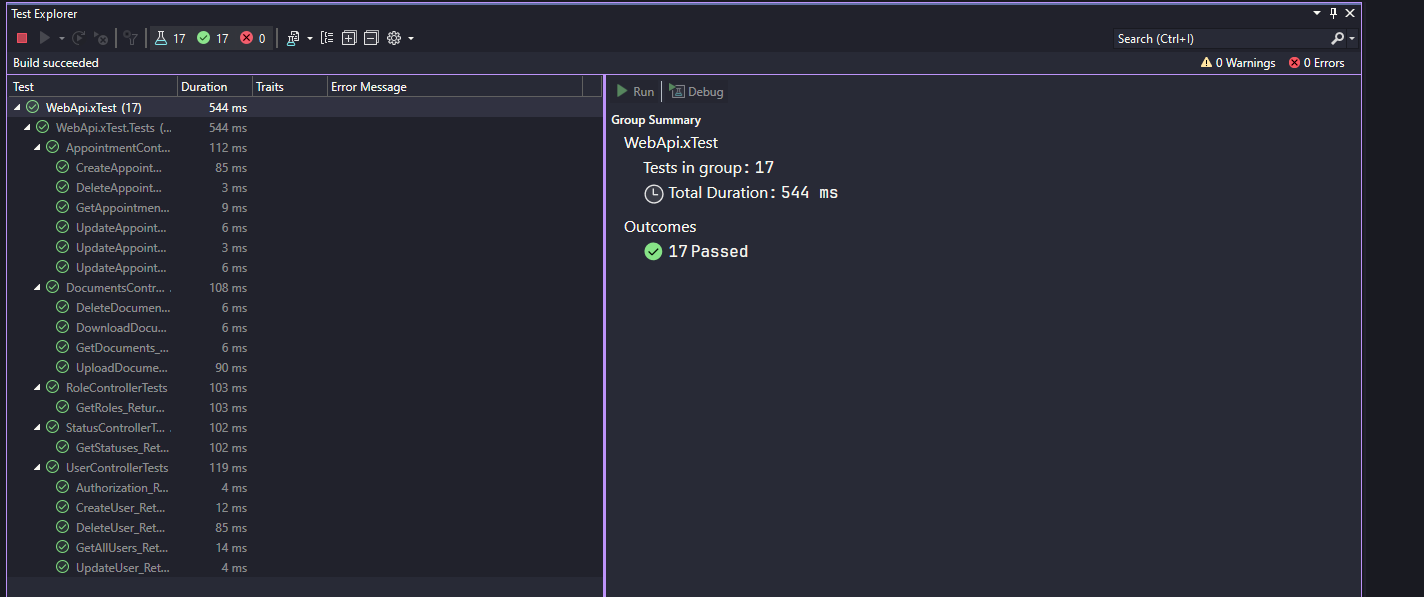


Рисунок 50. Результаты Unit тестирования

Подводя итоги Unit тестирования, можно сказать, что API хорошо обрабатывает данные и соответствует требованиям системы.

**ЗАКЛЮЧЕНИЕ**

Производственная практика по ПМ.02 «Осуществление интеграции программных модулей» проходящая в организации АО «НЗ 70-летия Победы», предоставляющая возможность применить знания, полученные в ходе обучения и учебной практики в реальном производственном процессе для решения проблемы обозначенной руководителем практики от предприятия, была успешно пройдена.

Знания, полученные в ходе обучения, были применены для создания модуля информационной системы, работающего с данными предприятия заказчика – API для системы обработки заявок на исправление каких-либо нарушений работ систем или на улучшения в предприятии от его работников, в соответствии с чем были достигнуты все поставленные цели и задачи.

Были приобретены следующие практические навыки:

* составление требований к разрабатываемому модулю и системе в целом;
* составление технического задания;
* составление описания структуры информационной системы и её взаимодействия с пользователем;
* разработка модуля информационной системы;
* составление тестовой документации и тестирование по ней разработанного модуля.

Из вышесказанного можно сделать вывод, что разработанный программный продукт удовлетворил требования заказчика.

**СПИСОК ИСПОЛЬЗОВАНОЙ ЛИТЕРАТУРЫ**

1. Галиаскаров, Э. Г.  Анализ и проектирование систем с использованием UML: учебное пособие для вузов / Э. Г. Галиаскаров, А. С. Воробьев. — Москва: Издательство Юрайт, 2024. — 125 с. — (Высшее образование). — ISBN 978-5-534-14903-6. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/544559 (дата обращения: 12.06.2024).
2. Черткова, Е. А.  Программная инженерия. Визуальное моделирование программных систем: учебник для среднего профессионального образования / Е. А. Черткова. — 3-е изд., испр. и доп. — Москва: Издательство Юрайт, 2024. — 146 с. — (Профессиональное образование). — ISBN 978-5-534-18094-7. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/539955 (дата обращения: 13.06.2024).
3. Советов, Б. Я.  Базы данных: учебник для вузов / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 4-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2024. — 403 с. — (Высшее образование). — ISBN 978-5-534-18479-2. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/535113 (дата обращения: 14.06.2024).
4. Маркин, А. В.  Программирование на SQL: учебное пособие для среднего профессионального образования / А. В. Маркин. — Москва: Издательство Юрайт, 2023. — 435 с. (дата обращения 15.06.2024).
5. Справочник по Transact-SQL (ядро СУБД) – URL: https://learn.microsoft.com/ru-ru/sql/t-sql/language-reference?view=sql-server-ver16 (дата обращения 15.06.2024).
6. Руководство по MS SQL Server 2022 – URL: https://metanit.com/sql/sqlserver/ (дата обращения 15.06.2024).
7. T-SQL Syntax and database concepts (Справочник по Transact-SQL) – URL: https://victorromeo.github.io/2020/01/02/TSQL-Syntax.html (дата обращения 15.06.2024).
8. Среда для разработки Microsoft Visual Studio – URL: <https://visualstudio.microsoft.com/ru/> (дата обращения 15.06.2024).
9. Документация по системе контроля версий Git – URL: https://git-scm.com/doc (дата обращения 15.06.2024).
10. Документация по языку программирования C# – URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения 19.06.2024).
11. ASP.NET CORE (Создание API) – URL: https://metanit.com/sharp/aspnet6/2.11.php (дата обращения 20.06.2024).
12. Руководство по созданию веб-API с помощью ASP.NET Core – URL: https://learn.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio (дата обращения 21.06.2024).
13. Документация по Git (Работа с системой контроля версий) – URL: https://devdocs.io/git/ (дата обращения 24.06.2024).
14. Документация по Unit-тестированию в языке программирования C# (Си Шарп) – URL: https://dev.to/bytehide/unit-testing-with-c-and-net-full-guide-5c7p (дата обращения 25.06.2024).
15. Документация по Unit-тестированию ASP.Net Web API 2 – URL: https://learn.microsoft.com/en-us/aspnet/web-api/overview/testing-and-debugging/unit-testing-with-aspnet-web-api (дата обращения 25.06.2024).

**ПРИЛОЖЕНИЯ**

**Приложение А**

**Структура предприятия**

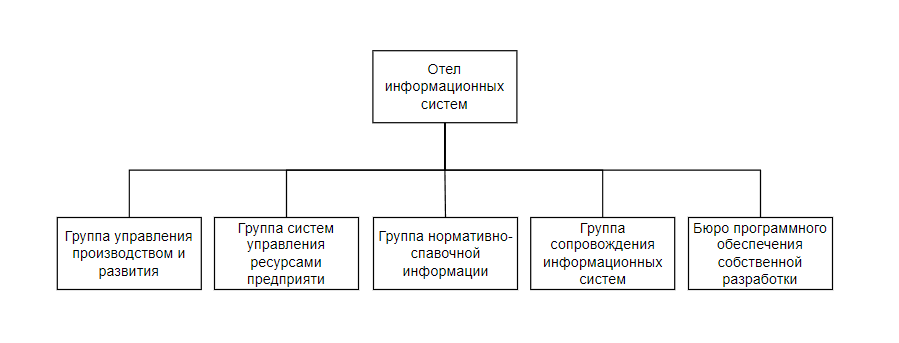


Рисунок 1. Структура информационного отдела АО «НЗ 70-летия Победы»

**Приложение Б**

**Топология сети предприятия**

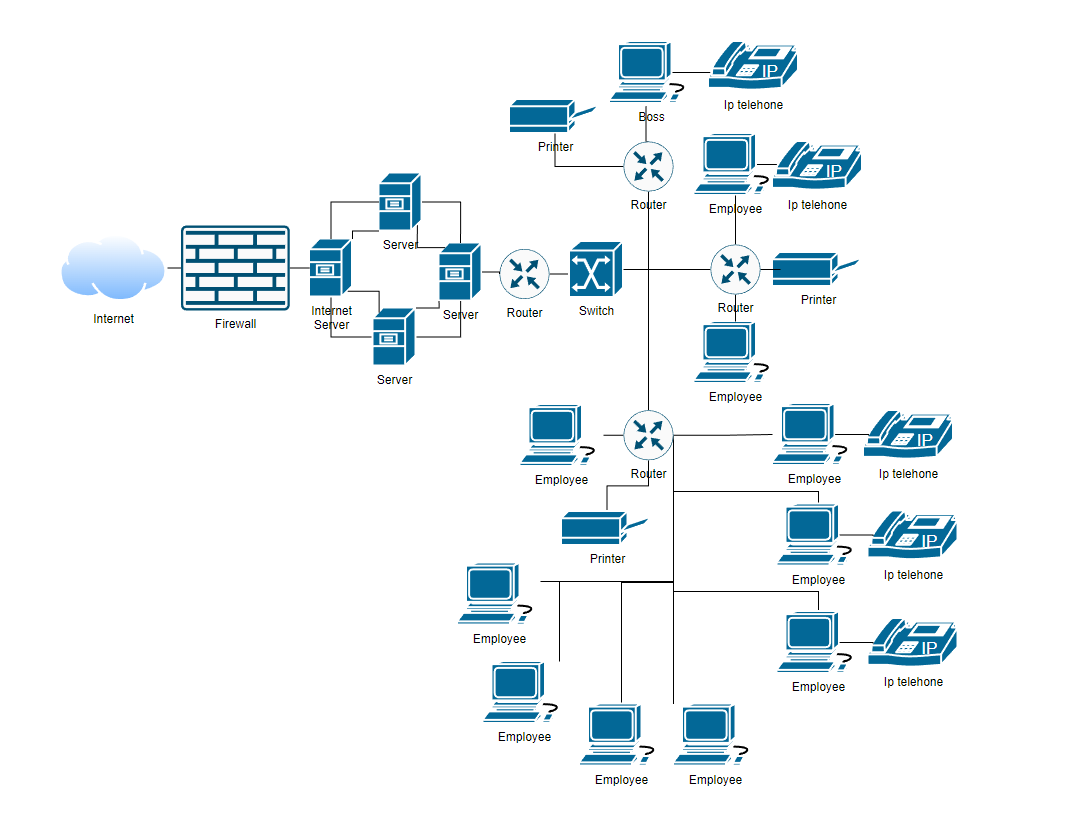


Рисунок 1. Примерная топология сети информационного отдела АО «НЗ 70-летия Победы»

**Приложение В**

**Структура базы данных информационной системы**

Таблица – 1 – Роли (Role)

|  |  |  |
| --- | --- | --- |
| Ключевое поле | Название | Назначение |
| ПК (первичный ключ) | Код (role\_id) | Идентификация роли в виде числа |
|  | Название (tite) | Перечень названий ролей |

Таблица – 2 – Пользователи (User)

|  |  |  |
| --- | --- | --- |
| Ключевое поле | Название | Назначение |
| ПК (первичный ключ) | Код (user\_id) | Идентификация пользователя в виде числа |
|  | Логин (login) | Логин пользователя для входа в систему |
|  | Пароль (password) | Пароль пользователя для входа в систему |
|  | Фамилия (surname) | Фамилия пользователя |
|  | Имя (name) | Имя пользователя |
| ФК (вторичный ключ) | Роль (role) | Числовой идентификатор роли, присвоенной пользователю |

Таблица – 3 – Статусы заявок (Status)

|  |  |  |
| --- | --- | --- |
| Ключевое поле | Название | Назначение |
| ПК (первичный ключ) | Код (status\_id) | Идентификация статуса заявки в виде числа |
|  | Название (tite) | Перечень названий статусов заявок |

Таблица – 4 – Заявки (Appointment)

|  |  |  |
| --- | --- | --- |
| Ключевое поле | Название | Назначение |
| ПК (первичный ключ) | Код (appointment\_id) | Идентификация заявки в виде уникальной хеш последовательности |
|  | Проблема (problem) | Краткое описание причины заявки пользователем |
|  | Описание проблемы (discription\_problem) | Более развёрнутое описание причины заявки |
|  | Место (place) | Фамилия пользователя |
|  | Дата создания (date\_create) | Дата, когда была создана заявка |
|  | Дата одобрения (date\_approve) | Дата, когда была одобрена заявка |
|  | Дата исправления (date\_fix) | Дата, когда были устранена причина заявки |
| ФК (вторичный ключ) | Статус (status) | Числовой идентификатор статуса заявки |

Таблица – 5 – Документы (Documents)

|  |  |  |
| --- | --- | --- |
| Ключевое поле | Название | Назначение |
| ПК (первичный ключ) | Код (document\_id) | Идентификация документа в виде уникальной хеш последовательности |
|  | Название (tite) | Перечень названий документов |
| ФК (вторичный ключ) | Заявка (appointment) | Уникальная хеш последовательность - идентификатор заявки, к которой принадлежит документ |

Таблица – 6 – Заявки пользователя (Users\_appointment)

|  |  |  |
| --- | --- | --- |
| Ключевое поле | Название | Назначение |
| ФК (вторичный ключ) | Заявка (appointment) | Уникальная хеш последовательность - идентификатор заявки, которую создал пользователь |
| ФК (вторичный ключ) | Пользователь (user) | Уникальная хеш последовательность - идентификатор пользователя, который создал заявку |

Таблица – 7 – Связи таблиц базы данных информационной системы

|  |  |
| --- | --- |
| Названия таблиц | Тип связи |
| Статусы - Заявки | 1 : М |
| Пользователи - Заявки | М : М |
| Роли - Пользователи | 1 : М |
| Заявки - Документы | 1 : М |

**Приложение Г**

**Схема базы данных для информационной системы**

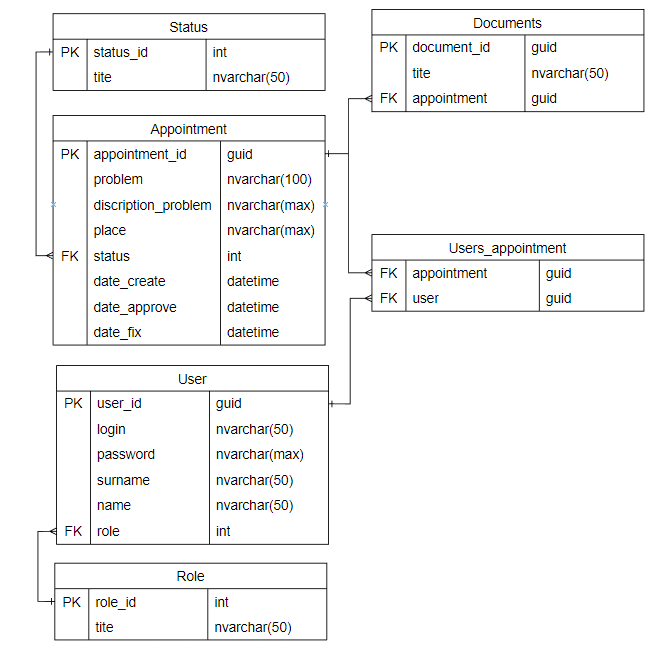


Рисунок 1. ER-диаграмма базы данных информационной системы

**Приложение Д**

**Техническое задание**

1. Общие требования
   1. Назначение документа

Настоящее Техническое задание определяет требования и порядок разработки модуля программного обеспечения – API для работы с заявками на исправление каких-либо нарушений работ систем или на улучшения в предприятии от его работников и необходимой документацией для составляемых заявок.

* 1. Наименование Исполнителя и Заказчика

Заказчик – АО «НЗ 70-летия Победы»; г. Нижний Новгород, Сормовское шоссе 21;

Исполнитель – Чернов С.В.; Адрес: г. Н. Новгород, Московское ш., 1.

* 1. Информация о компании

Название: АО «НЗ 70-летия Победы»; г. Нижний Новгород;

Отрасль и направления деятельности: военное производство;

Перечень услуг:

* регистрация пользователя администратором;
* авторизация пользователя в системе;
* изменение данных пользователя администратором;
* предоставление данных о пользователях;
* удаление пользователя администратором;
* создание заявки обычным пользователем;
* изменение заявки обычным пользователем;
* удаление заявки;
* изменение документа из заявки пользователем с ролью руководителя;
* изменение статуса заявки на выполненную исполнителем;
* предоставление документации для составления заявки;
* предоставление информации о заявках в соответствии с фильтрацией по выполнению, подтверждению, пользователям в разных ролях (создатель, подтверждающая сторона, исполнитель), дате и подобному.

Перечень услуг может изменяться в соответствии с требованиями Заказчика.

Публичные контактные данные: +7 (831) 249-82-38;

* 1. Основание для разработки сайта

Основанием для разработки сайта является Договор № 00000001 от 06.06.2024 между Исполнителем и Заказчиком.

* 1. Назначение программного продукта

Программный модуль необходим для полу автоматизации документооборота и отслеживания деятельности различных сфер жизнеобеспечения для предприятия.

Актуальность разработки:

* разрабатываемый программный модуль позволяет реализовать различные вариации клиентских приложений, повышая удобство пользователей.
* благодаря реализации разрабатываемого программного модуля можно безопасно обмениваться данными.
* благодаря реализации разрабатываемого программного модуля можно отслеживать недостатки в функционировании предприятия.
  1. Целевая аудитория

Пол: женский, мужской.

Возраст: от 18 до 60 лет.

Занятость: работники предприятия.

Решаемая услугами проблема целевой аудитории: удобный документооборот и общение с другими структурами предприятия, способными решить проблему работника предприятия.

* 1. Цели и задачи программного модуля

Тип модуля: REST API

Цели модуля:

* создание заявления на решение какой-либо проблемы предприятия, для улучшения его работы;
* реализация документооборота для удобства оформления/ подтверждения/документирования проделанной пользователем системы работы.
* предоставление информации о пользователях;
* отслеживание проблем на предприятии.

Целевые действия пользователе:

* создание заявки на исправление какой-либо проблемы предприятия;
* получение документации, по запросу.
  1. Стадии разработки

Этапы разработки, ответственные за данные этапы и сроки этапов разработки программного продукта указаны в таблице 2.

Таблица – 2 – Этапы разработки программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| Стадии | Этапы | Ответственность | Сроки |
| Определение требований | Выявление и документирование потребностей и ожиданий заказчика/пользователя. | С.В. Чернов | 06.06.2024 – 09.06.2024 |
| Анализ существующих систем и выявление проблем. |
| Определение функциональных и нефункциональных требований к новому ПО. |
| Проектирование | Разработка архитектуры системы. | 10.06.2024 – 15.06.2024 |
| Проектирование модулей, интерфейсов и взаимодействия между ними. |
| Реализация (кодирование) | Написание исходного кода программы. | 16.06.2024 – 20.06.2024 |
| Разработка и подключение базы данных. |
| Модульное тестирование отдельных компонентов. |
| Тестирование | Проверка функциональности, производительности, безопасности и других характеристик. | 22.06.2024 – 23.06.2024 |
| Выявление и устранение ошибок и дефектов. |
| Обеспечение качества и соответствие требованиям. |
| Развертывание | Подготовка программного продукта к эксплуатации. | 24.06.2024 – 27.06.2024 |
| Установка, настройка и предоставление доступа пользователям. |
| Обучение персонала работе с новой системой. |
| Сопровождение и поддержка | Исправление ошибок и внесение изменений по запросам пользователей. | - |
| Модернизация, обновление и масштабирование системы. |
| Мониторинг работоспособности и решение возникающих проблем. |

* 1. Ресурсы

Для разработки необходимы ресурсы, указанные далее.

1. Человеческие ресурсы – разработчики и пользователи.
2. Инфраструктурные ресурсы – база данных центра животных.
3. Программные ресурсы:

* ОС Windows 10;
* Visual Studio;
* система контроля версий Git;
* система хранения данных Gogs;
* Microsoft SQL Server.

1. Информационные ресурсы:

* документация по языкам программирования.

1. Финансовые ресурсы – оплата труда участников разработки.
2. Нефункциональные требования
   1. Требования к персоналу

Технические специалисты узкого профиля для поддержания программного модуля необходимы, а именно специалист по базам данных и разработчик на языке программирования C#, понимающий ASP.NetCore.

Необходимыми и достаточными навыками будет владение общими понятиями работы с персональным компьютером на уровне базового пользователя и выше.

* 1. Требования к разграничению доступа

Информация, представленная на сайте, является общедоступной и размещается с целью ознакомления.

В соответствии с правами доступа можно выделить 2 категории пользователей сайта:

* посетители – доступ к всей общедоступной информации, чтение и просмотр, создание заявки;
* Начальник – доступ к всей общедоступной информации, чтение и просмотр, создание заявки, возможность одобрять заявки других пользователей;
* Мастер – доступ к всей общедоступной информации, чтение и просмотр, создание заявки, отметка заявки, как обработанной и закрытой (проблема, описанная в заявке, была решена);
* администратор (сотрудник компании Заказчика) – доступ к изменению и удалению информации.

Обеспечение защиты от несанкционированного доступа к данным учетных записей Пользователей осуществляется путём хеширования паролей, которые Пользователи вводят при регистрации, а также передаче данных от клиента к серверу (и обратно) по защищённому протоколу https.

1. Функциональные требования
   1. Примерный набор модулей
2. Регистрация и аутентификация пользователей;
3. Манипуляция заявками;
4. Манипуляция профилями пользователей.
   1. Возможности администрирования
5. Доступ к административному модулю должен быть ограничен и требовать аутентификации;
6. Администратор должен иметь возможность добавлять, редактировать и удалять пользователей, заявки, документацию;
7. Дополнительно
   1. Дополнительные пожелания
8. Дополнительные пожелания диктуются Заказчиком в процессе разработки и принимаются после тщательного анализа и за документирования;
9. Вследствие возможных временных издержек, по независящим от Исполнителя обстоятельствам, не исключается риск изменения типа информационного продукта на консольное приложение. Заказчик принимает во внимание возможные риски разработки и не имеет претензий в отношении Исполнителя в случае, описанном выше.

**Приложение Ж**

**Фотографии программного продукта**

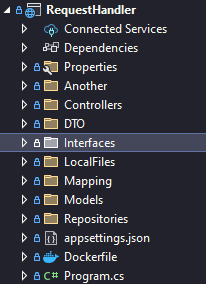


Рисунок 1. Структура проекта

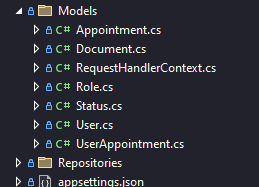


Рисунок 2. Модели из базы данных

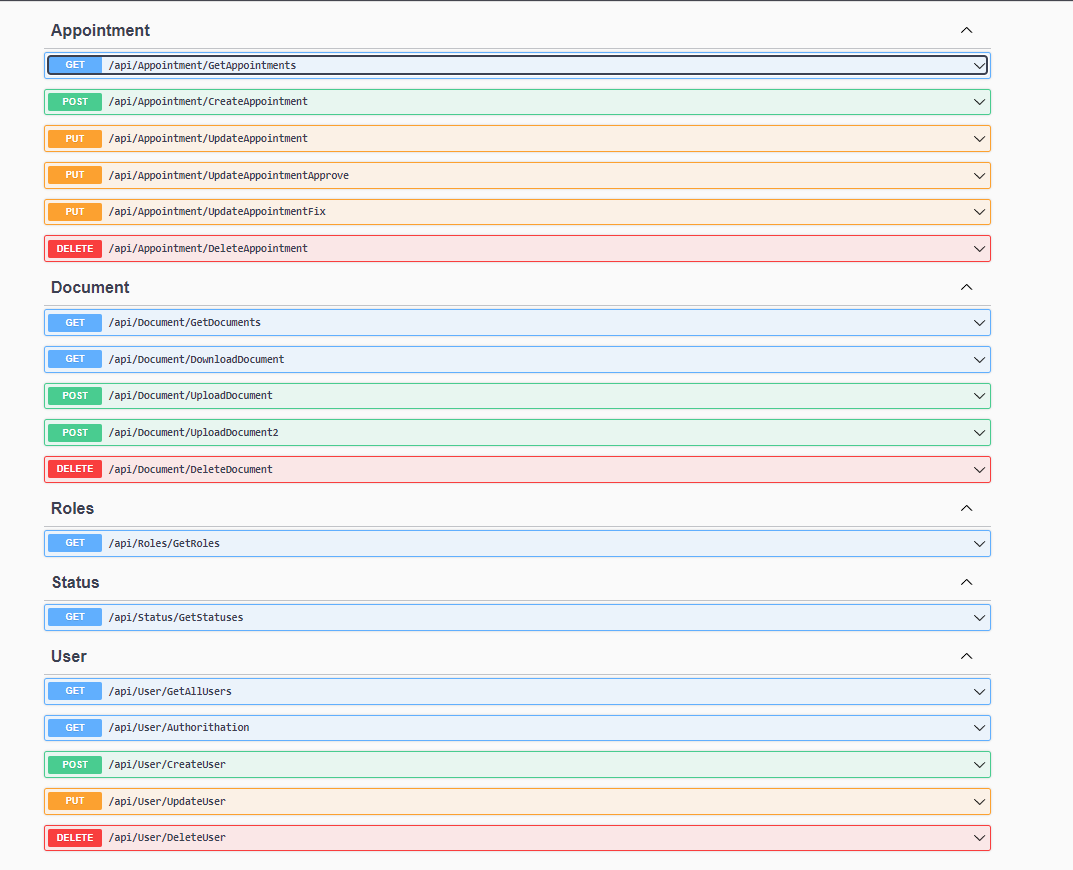


Рисунок 3. Результат создания API

**Приложение З**

**Листинги программного продукта**

### Таблица – 1 – UserRepository

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  using RequestHandler.Another;  using RequestHandler.Interfaces;  using RequestHandler.Models;  namespace RequestHandler.Repositories  {  public class UserRepository : IUserRepository  {  private readonly RequestHandlerContext \_context;  public UserRepository(RequestHandlerContext context)  {  \_context = context;  }  //проверка наличия пользователя по логину  public async Task<bool> UserExists(string login)  {  return await \_context.Users.AnyAsync(u => u.Login == login);  }  //проверка наличия пользователя по id  public async Task<bool> UserExists(Guid UserId)  {  return await \_context.Users.AnyAsync(u => u.UserId == UserId);  }  //получние списка всех пользователей  public async Task<ICollection<User>> GetAllUsers(int? roleId = null)  {  List<User> users;  if (roleId != null)  users = \_context.Users.Where(u => u.Role == (int)roleId).  Include(r => r.RoleNavigation).ToList();  else  users = \_context.Users.  Include(r => r.RoleNavigation).ToList();  return users.OrderBy(r => r.RoleNavigation.RoleId).ToList();  }  //проверка на администратора  public async Task<bool> ValidateAdmin(Guid logUserId)  {  if (\_context.Users  .FirstAsync(u => u.UserId == logUserId)  .Result.Role == 1)  return true;  return false;  }  //проверка на мастера  public async Task<bool> ValidateMaster(Guid logUserId)  {  if (\_context.Users  .FirstAsync(u => u.UserId == logUserId)  .Result.Role == 2)  return true;  return false;  }  //проверка на управляющего  public async Task<bool> ValidateApproval(Guid logUserId)  {  if (\_context.Users  .FirstAsync(u => u.UserId == logUserId)  .Result.Role == 3)  return true;  return false;  }  //авторизация  public async Task<User> Authorithation(string login, string password)  {  string hexPassword = Hashing.ToSHA256(password);  return await \_context.Users  .Include(r => r.RoleNavigation)  .FirstOrDefaultAsync(u => u.Login == login  && u.Password == hexPassword);  }  //создание пользователя  public async Task<bool> CreateUser(string login, string password, string surname, string name, int role)  {  password = Hashing.ToSHA256(password);  User user = new User()  {  UserId = new Guid(),  Login = login,  Password = password,  Surname = surname,  Name = name,  Role = role  };  await \_context.Users.AddAsync(user);  return await Save();  }  //сохранение результата  public async Task<bool> Save()  {  var save = await \_context.SaveChangesAsync();  return save > 0 ? true : false;  }  //обновление пользователя  public async Task<bool> UpdateUser(Guid id, string? login = null, string? password = null, string? surname = null, string? name = null, int? role = null)  {  var user = await \_context.Users.FirstOrDefaultAsync(u => u.UserId == id);  user.Login = string.IsNullOrEmpty(login) ? user.Login : login;  user.Password = string.IsNullOrEmpty(password) ? user.Password : Hashing.ToSHA256(password);  user.Surname = string.IsNullOrEmpty(surname) ? user.Surname : surname;  user.Name = string.IsNullOrEmpty(name) ? user.Name : name;  if (role != null && role >=1 && role <= 4)  {  user.Role = (int)role;  }  \_context.Users.Update(user);  return await Save();  }  //удаление пользователя администратором  public async Task<bool> DeleteUser(Guid id)  {  var user = await \_context.Users.FirstOrDefaultAsync(u => u.UserId == id);  \_context.Users.Remove(user);  return await Save();  }  }  } |

### Таблица – 2 – Hashing

|  |
| --- |
| using System.Security.Cryptography;  using System.Text;  namespace RequestHandler.Another  {  public static class Hashing  {  public static string ToSHA256(string s)  {  using var sha256 = SHA256.Create();  var bytes = Encoding.UTF8.GetBytes(s);  var hash = sha256.ComputeHash(bytes);  var hex = BitConverter.ToString(hash).Replace("-", "").ToLower();  return hex;  }  }  } |

### Таблица – 3 – UploadFile

|  |
| --- |
| //отправка файла на сервер  public async Task<bool> UploadFile(IFormFile file, string title)  {  try  {  var filepath = Path.Combine(Directory.GetCurrentDirectory(), "LocalFiles");  if (!Directory.Exists(filepath))  {  Directory.CreateDirectory(filepath);  }  var exactpath = Path.Combine(Directory.GetCurrentDirectory(), "LocalFiles", title);  using (var stream = new FileStream(exactpath, FileMode.Create))  {  await file.CopyToAsync(stream);  }  return true;  }  catch (Exception ex)  {  return false;  }  } |

### Таблица – 4 – DownloadFile

|  |
| --- |
| //получение файла  public async Task<FileContentResult> DownloadFile(string title)  {  var filepath = Path.Combine(Directory.GetCurrentDirectory(), "LocalFiles", title);  var provider = new FileExtensionContentTypeProvider();  if (!provider.TryGetContentType(filepath, out var contenttype))  {  contenttype = "application/octet-stream";  }  var bytes = await System.IO.File.ReadAllBytesAsync(filepath);  return new FileContentResult(bytes, contenttype)  {  FileDownloadName = title  };  } |

**Приложение И**

**Тестовая документация**

1. Перечень работ

Предметом для тестирования является программный продукт, а именно API для работы с заявками на исправление каких-либо нарушений работ систем или на улучшения в предприятии от его работников и необходимой документацией для составляемых заявок

Тестируемыми частями данного объекта являются программный код, разбитый на модули.

Сама программа представлена в части 2, пункте 2.3. Разработка информационной системы.

1. Критерии качества

Наиболее важной секцией является программный код, реализующий основную логику программы, поскольку только благодаря ему объект может использоваться по прямому назначению – являться промежуточным безопасным интерфейсом между клиентской частью системы и базой данных.

Оценочные критерии:

* функциональность – программа выполняет необходимые пользователю функции.
* надёжность – программа устойчива к сбоям;
* лёгкость применения – программа имеет интуитивно понятный интерфейс и не требует особой квалификации пользователя;
* эффективность – программа использует маленькое и/или оптимальное для своей работы количество ресурсов, не сильно нагружая систему;
* сопровождаемость – программа имеет поддержку со стороны разработчика;
* мобильность – программа легко переносится между устройствами и системами.

1. Оценка рисков

Оценка возможных (предполагаемых) рисков приведена в таблице 3. В качестве метрик для оценки в поле «Вероятность» используется градация «часто/ редко/очень редко/никогда», а в поле «Критичность» - «Серьёзная/Средняя/ Маленькая».

Таблица – 3 – Оценка возможных рисков эксплуатации

|  |  |  |  |
| --- | --- | --- | --- |
| Риск | Вероятность | Критичность | Алгоритм решения |
| Сбой алгоритма работы программы | Очень редко | Серьёзная | Перезапустить программу.  Если ошибка не будет исправлена перезапуском– переустановить программу и проверить свежеустановленную версию на работоспособность. |
| Не корректное отображение интерфейса | Редко | Маленькая | Перезапустить программу.  Если ошибка не будет исправлена перезапуском– переустановить программу и проверить свежеустановленную версию на работоспособность. |
| Поломка обработчика исключений | Очень редко | Средняя | Перезапустить программу.  Если ошибка не будет исправлена перезапуском– переустановить программу и проверить свежеустановленную версию на работоспособность. |
| Антивирус указывает на наличие в программе вирусов. | Никогда | Серьёзная | Удалить программу и обратиться в поддержку в месте приобретения программы. |

1. Документация

По итогу проводимых тестов будет создана документация по использованию тестируемого объекта, а также тест кейсы (как тестировался API) и баг-репорты (результаты тестов из тест-кейсов и дефекты или ошибки, которые произошли).

1. Стратегия тестирования

Стратегия тестирования будет одна – тестирование белым ящиком с помощью тестовых классов (Unit тестирование).

1. Ресурсы

Для выполнения тестирования необходимы следующие ресурсы:

* программные: «Диспетчер задач» или «Монитор процессов», текстовый редактор для записей, исполняемая среда для запуска программы, операционная система способная поддерживать API, интегрированная среда разработки Visual Studio 2022, браузер;
* аппаратные: компьютер, способный хранить и реализовывать потенциал операционной системы, программной среды и самой программы, а также текстового редактора для составления документации или оформления отчётов;
* человеческие: 1 человек – один специалист по языку программирования, на котором написана программа и по алгоритмам;
* временные: примерно 5 часов (около 3 часов тестирование, остальное время на написание документации по заготовленному заранее шаблону);
* финансовые: зафиксированы в техническом задании (часть 2, пункт 2.1.1.10. Ресурсы).

1. Метрики

Оценка проводится по следующим метрикам:

* пройден/не пройден – описание характеристики прохождения теста.

1. Расписание и ключевые точки

Расписание испытаний в виде временных промежутков, указанных ранее временных общих рамок (уже при приобретённом инвентаре и готовых шаблонов для заполнения документации), указано в таблице 4.

Таблица – 4 – Расписание тестирования

|  |  |
| --- | --- |
| Время (часы) | Действие/Испытание |
| 1 | подготовка инвентаря;  установка необходимого ПО;  расписание тестов; |
| 2-3 | White-box тестирование; |
| 4-5 | Анализ тестов и окончательная фиксация результатов;  Оформление документации по прошедшему тестированию; |

**Приложение К**

**Тест-кейсы**

Таблица – Х – Тестирование

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Тест-кейс | Входные данные | Ожидаемый результат | Фактический результат | Прохождение теста |
| 1 | Авторизация пользователя | login = "Ivan\_Kuznetsov";  password = "12345"; | IsType< OkObjectResult> | IsType< OkObjectResult> | Authorization\_ReturnOk |
| 2 | Создание профиля пользователя | logUserId = "38C13A21-ABE4-40C3-9877-33126F386E7B";  login = "Chernov\_Sergey";  password = "12345";  surname = "Чернов";  name = "Сергей";  role = 1; | IsType< OkObjectResult> | IsType< OkObjectResult> | CreateUser\_ReturnOk |
| 3 | Получение списка всех пользователей | logUserId = "38C13A21-ABE4-40C3-9877-33126F386E7B";  role = null; | IsType< OkObjectResult> | IsType< OkObjectResult> | GetAllUsers\_ReturnOk |
| 4 | Обновление профиля пользователя | logUserId = "38C13A21-ABE4-40C3-9877-33126F386E7B";  id = "86716A63-8093-4706-A975-0046FBEC60F8";  login = "Sergey\_Chernov"; | IsType<NoContentResult> | IsType<NoContentResult> | UpdateUser\_ReturnOk |
| 5 | Удаление профиля пользователя | logUserId = "38C13A21-ABE4-40C3-9877-33126F386E7B";  id = "86716A63-8093-4706-A975-0046FBEC60F8"; | IsType<OkObjectResult> | IsType<OkObjectResult> | DeleteUser\_ReturnOk |
| 6 | Получение списка статусов |  | IsType<OkObjectResult> | IsType<OkObjectResult> | GetStatuses\_ReturnOk |

Продолжение таблицы Х.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 | Получение списка ролей |  | IsType<OkObjectResult> | IsType<OkObjectResult> | GetRoles\_ReturnOk |
| 8 | Получение списка документов | userId = 38C13A21-ABE4-40C3-9877-33126F386E7B; | IsType<OkObjectResult> | IsType<OkObjectResult> | GetDocuments\_ReturnOk |
| 9 | Скачивание документа с сервера | itle = "fi.txt"; | IsType<FileContentResult> | IsType<FileContentResult> | DownloadDocument\_FileContentResult |
| 10 | Загрузка документа на сервер | logUserId = «38C13A21-ABE4-40C3-9877-33126F386E7B»;  appointmentId = null;  var filepath = Path.Combine("D:\\Progect\\forPractice\\progect\\RequestHandler\\WebApi.xTest\\LocalFiles\\", "fi.txt");  byte[] fileBytes = await System.IO.File.ReadAllBytesAsync(filepath);  IFormFile file = new FormFile(new MemoryStream(fileBytes), 0, fileBytes.Length, "Data", "fi.txt"); | IsType<OkObjectResult> | IsType<OkObjectResult> | UploadDocument\_ReturnOk |
| 11 | Удаление документа с сервера | logUserId = "38C13A21-ABE4-40C3-9877-33126F386E7B";  doucumentId = «0F739B90-2185-4524-958E-77B62C5F003A"; | IsType<OkObjectResult> | IsType<OkObjectResult> | DeleteDocument\_ReturnOk |
| 12 | Получение списка заявок | logUserId = "38C13A21-ABE4-40C3-9877-33126F386E7B";  roleId = 4; | IsType<OkObjectResult> | IsType<OkObjectResult> | GetAppointments\_ReturnOk |

Продолжение таблицы Х.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 13 | Создание заявки | userId = "86716A63-8093-4706-A975-0046FBEC60F8";  Problem = "Сломался чайник";  DiscriptionProblem = null;  Place = "5-тый корпус, 11 этаж, 107 кабинет"; | IsType<OkObjectResult> | IsType<OkObjectResult> | CreateAppointment\_ReturnOk |
| 14 | Обновление заявки создателем | appointmentId = "D0451E80-5123-41F1-9BDE-E9D51E646C52";  Problem = "Сломался чайник";  DiscriptionProblem = null;  Place = "5-тый корпус, 11 этаж, 107 кабинет"; | IsType<OkObjectResult> | IsType<OkObjectResult> | UpdateAppointment\_ReturnOk |
| 15 | Одобрение заявки | userId = "28ABB33E-D9AF-449E-AD72-C3510935612B";  appointmentId = "D0451E80-5123-41F1-9BDE-E9D51E646C52"; | IsType<OkObjectResult> | IsType<OkObjectResult> | UpdateAppointmentApprove\_ReturnOk |
| 16 | «Закрытие» заявки | userId = "86716A63-8093-4706-A975-0046FBEC60F8";  appointmentId = "D0451E80-5123-41F1-9BDE-E9D51E646C52"; | IsType<OkObjectResult> | IsType<OkObjectResult> | UpdateAppointmentFix\_ReturnOk |
| 17 | Удаление заявки | appointmentId = "D0451E80-5123-41F1-9BDE-E9D51E646C52"; | IsType<OkObjectResult> | IsType<OkObjectResult> | DeleteAppointment\_ReturnOk |