**СОДЕРЖАНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ЗМК.09.02.03.19.205.ПЗ

Разраб.

*МедведевВ.А..*

Провер.

*Алемасов Е.П.*

Рецензент

*Бабкин Ю.А.*

Н. Контр.

*Демидова В.М.*

Пояснительная записка

Лит.

Листов

96

ГР 205

[АННОТАЦИЯ 5](#_Toc26196310)

[ВВЕДЕНИЕ 6](#_Toc26196311)

[1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ 7](#_Toc26196312)

[1.1 Основное функциональное назначение программного средства 7](#_Toc26196313)

[1.2 Полное наименование программного средства 7](#_Toc26196314)

[1.3 Условное обозначение программного средства 7](#_Toc26196315)

[1.4 Разработчики программного средства 7](#_Toc26196316)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 8](#_Toc26196317)

[2.1 Основание для разработки 8](#_Toc26196318)

[2.2 Назначение разработки 8](#_Toc26196319)

[2.3 Требование к программному средству](#_Toc26196320) 8

[2.4 Требования к программной документации 8](#_Toc26196321)

[2.5 Требования к эргономике и технической эстетике 9](#_Toc26196322)

[2.6 Стадии и этапы разработки](#_Toc26196323) 9

[2.7 Порядок контроля и приемки 11](#_Toc26196324)

[3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ 12](#_Toc26196325)

[3.1 Декомпозиция поставленной задачи 12](#_Toc26196326)

[3.2 Общая архитектура программного средства](#_Toc26196327) 13

[3.3 Разработка алгоритма решения задачи 15](#_Toc26196328)

[3.4 Реализация функционального назначения программного средства](#_Toc26196329) 15

[3.5 Структурная организация данных](#_Toc26196330) 16

[3.6 Разработка интерфейса ПС 23](#_Toc26196331)

[3.7 Описание структуры выходной информации](#_Toc26196332) 39

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ](#_Toc26196333) 40

[4.1 Назначение программного средства](#_Toc26196334) 40

[4.2 Условия выполнения программного средства](#_Toc26196335) 40

[4.3 Эксплуатация программного средства](#_Toc26196336) 40

[4.4 Сообщения пользователю](#_Toc26196337) 43

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА](#_Toc26196338) 45

[5.1 Обоснование необходимого количества тестов](#_Toc26196339) 45

[5.2 Описание тестовых пакетов](#_Toc26196340) 46

[ЗАКЛЮЧЕНИЕ 6](#_Toc26196341)7

[Оценка качества программного средства с помощью метрик](#_Toc26196342) 68

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_Toc26196343) 70

[Приложение А листинг программного модуля](#_Toc26196344) 72

[Приложение Б Спецификация ПО](#_Toc26196343) 94

[Приложение B диск с исполняемым кодом программного модуля](#_Toc26196345) 96

**АННОТАЦИЯ**

На дипломную работу студента Медведева В.А. на тему: «Автоматизированная информационная система «Фирма по обслуживанию организационной техники»». Дипломная работа представлена пояснительной запиской на 100 листах машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве фирмы по обслуживанию организационной техники. В главе 2 приведено техническое задание на проектирование БД согласно ГОСТ 19.201. Глава 3 содержит пояснительную записку к программному продукту согласно ГОСТ 19.404-79. В главе 4 содержится руководство пользователя согласно ГОСТ 19.504-79. Глава 5 содержит обоснование количества необходимых тестов и тестовые пакеты для оценки работоспособности программного продукта. В главе 6 приведены результаты ручного и автоматического тестирования программного продукта и выполнена оценка качества с помощью метрик.

В работе содержится общее описание программного средства, предназначенного для объекта проектирования – интегрированного программного модуля «Фирма по обслуживанию организационной техники», изложены требования к программному средству, описаны его архитектура, реализация функций, интерфейс, приведено руководство пользователя, обоснованы тестовые пакеты и выполнена оценка качества программного продукта с помощью метрик.

**ВВЕДЕНИЕ**

В дипломной работе была поставлена задача разработать программный модуль ремонта организационной техники. Результатом работы должна быть программа, которая позволяет формировать отчет по контингенту. Программа должна иметь возможность водить и хранить данные о заявках на ремонт. Также программа должна выводить в печатной форме отчет по контингенту.

Для достижения поставленной цели был составлен основной алгоритм работы программы, а также разработаны алгоритмы вспомогательных процедур.

Для реализации алгоритмов была выбрана связка PostgreSQL + java. PostgreSQL был выбран как наиболее широко распространенная СУБД не требующая специальной установки. Для создания программы была выбрана бесплатная версия java 17 community edition.

При разработке интерфейса программы, я учитывал наиболее нужные и важные функциональные возможности, которые должна выполнять данная программа.

**1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена для хранения заявок на ремонт организационной техники. Программа не требует владения современными информационными технологиями для эффективного использования данного программного средства, т.е. подходит для любого пользователя.

**1.2 Полное наименование программного средства**

Разработка интегрированного программного модуля «Фирма по обслуживанию орг техники».

**1.3 Условное обозначение программного средства**

«Org-Tech»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 205 Медведев В.А.

**2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Основание для разработки**

Работа ведётся на основании задания на дипломный проект по специальности 09.02.03 — «Программирование в компьютерных системах».

**2.2 Назначение разработки**

Программа «Org-tech» работает под управлением операционных систем Windows 7 и выше. Программа предназначена для формирования отчета по контингенту.

**2.3 Требование к программному средству**

«Org-tech» устанавливается на персональный компьютер, имеющий представленные минимальные системные требования:

Оперативная память: 2048Мб;

Процессор: Intel ® Pentium ® Core i2 2000 МГц;

Разрешение экрана: не менее 1024х768

«Org-tech» написана на объектно-ориентированном языке программировании java. Программа не может подлежать сторонней оптимизации и редактированию программного кода.

**2.4 Требования к программной документации**

Состав программной документации должен включать в себя:

1. техническое задание;

2. сроки выполнения работ;

3. программу и методику испытаний;

4. эксплуатационные инструкции пользователю;

**2.5 Требования к эргономике и технической эстетике**

Система обеспечивает удобный для пользователей системный интерфейс, отвечающий следующим требованиям:

- единый стиль оформления для пользовательских интерфейсов;

- удобная, интуитивно понятная навигация в интерфейсе пользователя;

- взаимодействие пользователя с системой осуществляется на двух языках по выбору (русский и английский), для наибольшего удобства

Пользовательские интерфейсы системы спроектированы и разработаны с применением единых принципов графического представления информации и организации доступа к функциональным возможностям и сервисам. Разработан графический дизайн пользовательских интерфейсов, цветовые, шрифтовые и композиционные решения для отображения текстов, изображений, таблиц, гиперссылок, управляющих и навигационных элементов (меню, кнопок, форм и т.п.).

Экранные формы спроектированы с учетом требований унификации:

- все экранные формы пользовательского интерфейса выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

- для обозначения сходных операций использованы сходные графические значки, кнопки и другие управляющие (навигационные) элементы. Термины, используемые для обозначения типовых операций (добавление информационной сущности, редактирование поля данных), а также последовательности действий пользователя при их выполнении унифицированы;

- внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки) реализованы одинаково для однотипных элементов.

**2.6 Стадии и этапы разработки**

Разработка должна быть проведена в три стадии:

1. разработка технического задания;

2. рабочее проектирование;

3. внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;

2. разработка программной документации;

3. испытания программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;

2. определение и уточнение требований к техническим средствам;

3. определение требований к программе;

4. определение стадий, этапов и сроков разработки программы и документации на нее;

5. выбор языков программирования;

6. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с пунктом Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1. разработка, согласование и утверждение порядка и методики испытаний;

2. проведение испытаний;

3. корректировка программы и программной документации по результатам испытаний.

**2.7 Порядок контроля и приемки**

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой и календарным планом. Сдача-приемка осуществляется комиссией, в состав которой входят представители ГАПОУ и работодатель. По результатам приемки подписывается акт приемочной комиссии.

В указанном подразделе, согласно таблице «Сроки выполнения работ» технического задания, будет описаны перечень участвующих организаций, место и сроки проведения работ, согласно п. 2.8 ГОСТ 34.602-89.

Порядок согласования и утверждения приемочной документации должен регламентироваться организационно-распорядительной документацией организации, принимающей участие в создании программного продукта. Согласно разделу «Приемка результатов разработки» ГОСТ 15.001-88 для согласования и утверждения приемочной документации создается приемочная комиссия (приказом).

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

**3. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ**

**3.1 Декомпозиция поставленной задачи**

Для достижения цели работы была поставлена следующая задача:

Создать модуль “CodeDatabase” отвечающий за сохранение БД. Результат работы модуля - сохранение БД.

Поставленные задачи наглядно отображаются на диаграмме деятельности.

**Изображение выглядит как диаграмма, текст, План, зарисовка

Автоматически созданное описание**

Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

В «Org-tech» используется событийно-ориентированная архитектура (Рис.3). Роль агента (источник событий) в программе выполняют: пользователи программы, за роль стоков (потребители событий) отвечают таблицы, входящие в базу данных. Например, когда пользователь выбирает определенное действие: добавить, удалить, редактировать, вывести и т.д. система осуществит выбранные действия, и база отреагирует соответствующим образом: запись добавлена, удалена, отредактирована, либо отчет выведен на экран. Системная архитектура пользователя рассматривает это изменение состояния как событие, создаваемое, публикуемое, определяемое и потребляемое различными приложениями в. составе архитектуры

Изображение выглядит как текст, снимок экрана, круг, линия

Автоматически созданное описание

Рис.2 Диаграмма вариантов использования

Выбор действия

Добавить запись в БД

Удалить запись из БД

Редактировать запись в БД

Искать запись в БД

Выход

Рис.3 Схема событийно-ориентированной архитектуры

**3.3 Разработка алгоритма решения задачи**

**Общий алгоритм**

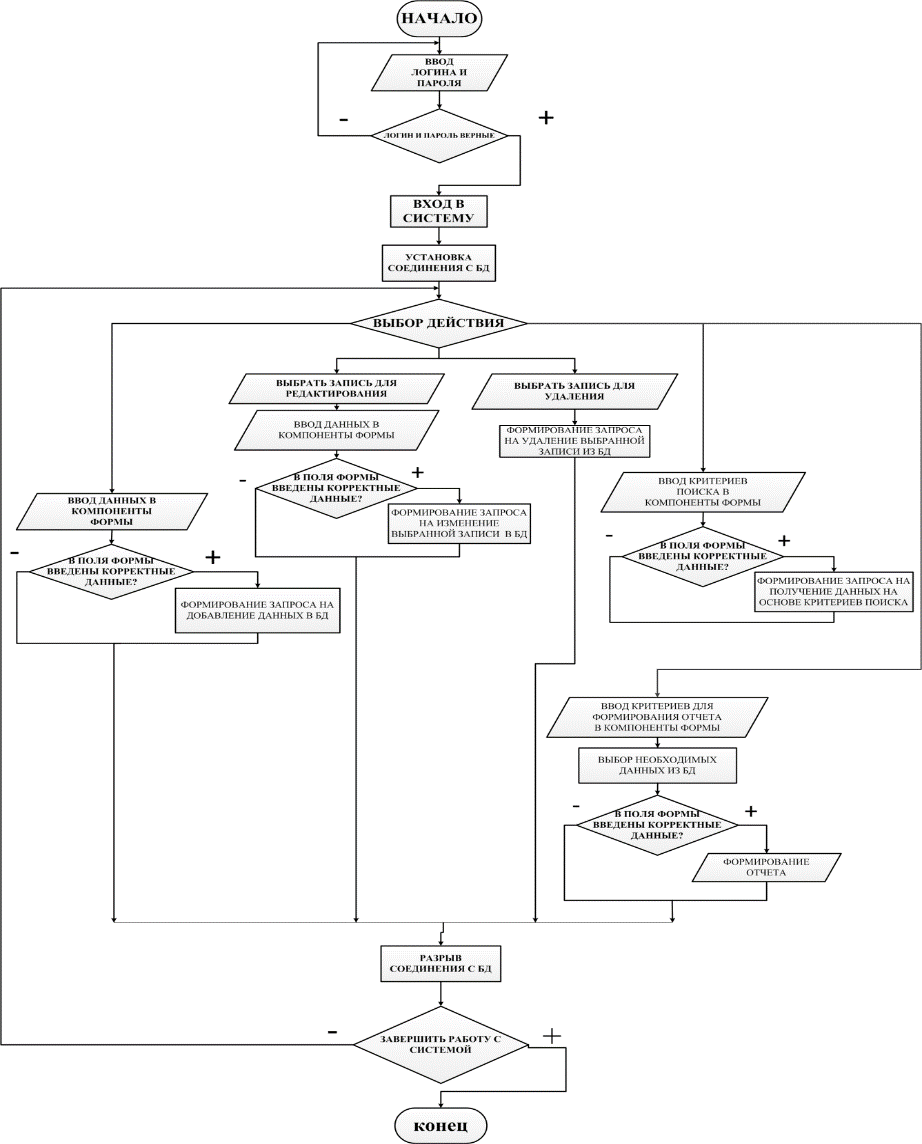
****

Рис.4 Алгоритм программы

**3.4 Реализация функционального назначения программного средства**

Программа «Org-tech» имеет определенный набор входных данных, такие как: справочник типов техники, справочник возможного ремонта, справочник Клиентов, справочник мастеров.

Данные вводятся мастером в соответствующие поля ввода, снабженные всплывающими подсказками.

Выходными данными являются: отображение нового списка заявок в виде панельного интерфейса.

Данные вводятся пользователем в соответствующие поля вывода, снабженные всплывающими подсказками.

Выходные данные редактировать вручную невозможно, т.к. они служат в качестве отчетной информации для пользователя.

**3.5 Структурная организация данных**

Для создания БД необходимо определиться с данными, которые необходимы для полноценного функционирования системы. Все эти данные указаны в реляционной модели «БД ремонт организационной техники» представленная на рисунке. Любая реляционная база данных и называется реляционной, что характеризуется отношениями (relation) между таблицами. На рисунке изображены таблицы моей базы данных. При этом одна таблица является родительской (главной), а вторая – дочерней (подчиненной). Главной таблицей является «Пользователь». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кода (рис.5).

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Рис.5 БД ремонт организационной техники

Первичный ключ в базе уникален, используется для организации отношений между таблицами, который не может иметь пустых и повторяющихся значений. Первичными ключами в базе являются поля: id (таблица client), id (таблица status), id (таблица repair), id (таблица type), id (таблица gender), id (таблица master), id (таблица application). Остальные ID-ключи являются внешними ключами.

Для организации более эффективной обработки данных применяется нормализация. Таблицы моей БД находятся в 3НФ:

* БД находится в форме -1НФ потому, что
  + Таблица не имеет повторяющихся записей;
  + Каждый атрибут отношения хранит одно-единственное значение и не является списком, ни множеством значений;
  + Таблица не имеет повторяющихся групп полей.
    - Вторая нормальная форма(2НФ):
      * Устранены атрибуты, зависящие только от части уникального (первичного) идентификатора, т.е. ID.
        + Третья нормальная форма(3НФ):

Отсутствуют атрибуты, зависящие от атрибутов, не входящих в уникальный (первичный) ключ.

На основе реляционной модели базы данных мною в MS SQL была построена следующая база данных В ней первичным ключом является поле таблицы “gender” – “id”. Связь таблиц “gender” и “master” по полю “id” один ко многим (1:М). Таблицы “status” и “application” связаны по полю “status\_id” типом связи 1:М. Таблицы “repair” и “application” связаны по полю “possible\_repair\_id” типом связи 1:М. Таблицы “type” и “application” связаны по полю “type\_id” типом связи 1:М. Таблицы “client” и “application” связаны по полю “client\_id” типом связи 1:М. Таблицы “master” и “application” связаны по полю “master\_id” типом связи 1:М.

Все связи между таблицами базы данных типа один ко многим:

Таблицы “gender” и “master” связаны по полю “id” типом связи 1:М.

Таблицы “status” и “application” связаны по полю “status\_id” типом связи 1:М.

Таблицы “repair” и “application” связаны по полю “possible\_repair\_id” типом связи 1:М.

Таблицы “type” и “application” связаны по полю “type\_id” типом связи 1:М.

Таблицы “client” и “application” связаны по полю “client\_id” типом связи 1:М.

Таблицы “master” и “application” связаны по полю “master\_id” типом связи 1:М.

Ниже представлено подробное описание таблиц.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 6

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 7

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 8

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 10

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 11

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 12

3.6 Разработка интерфейса ПС

Программное средство состоит из исполняемого файла Org-Tech.exe, файла конфигурации MyAppConfig.xml и базы данных orgtech.db.

При успешном запуске программы на экране появляется главное окно (рис. 13).

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 13

Окно авторизации состоит из следующих элементов:

* Два поля ввода: "Логин" и "Пароль". Пользователь должен ввести свои учетные данные в соответствующие поля для входа в систему.
* Кнопка "Войти". После ввода логина и пароля пользователь нажимает на эту кнопку для авторизации.
* Кнопка "Зарегистрироваться", расположенная в верхней части окна справа. При нажатии этой кнопки пользователь может перейти к процессу регистрации нового аккаунта.
* Главное окно и другие элементы интерфейса станут доступны пользователю после успешной авторизации.

В случае отсутствия акта регистрации пользователь нажмет на кнопку Зарегистрироваться и перейдет на окно регистрации (рис. 14).

Изображение выглядит как текст, снимок экрана, программное обеспечение, дизайн

Автоматически созданное описание

Рисунок 14

Три поля ввода для ввода имени, отчества и фамилии пользователя.

* Поле ввода для логина.
* Поле ввода для пароля.
* Выпадающий список для выбора пола пользователя.

Две кнопки: "Зарегистрироваться" и "Назад". Кнопка "Зарегистрироваться" позволяет пользователю завершить процесс регистрации, а кнопка "Назад" возвращает к предыдущему окну или экрану.

Все элементы ввода являются обязательными для заполнения перед нажатием на кнопку "Зарегистрироваться". Системное меню и другие элементы интерфейса станут доступны пользователю после успешной регистрации и авторизации.

После успешной авторизации либо регистрации будет открыто главное окно (рис. 15)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 15

Главное окно состоит из следующих элементов:

В верхней части окна расположен заголовок "Ремонт орг техники" и кнопка "Личный кабинет" для перехода в личный профиль пользователя.

В левой части окна находятся кнопки навигации:

* "Мои заявки" – кнопка для перехода к списку заявок мастера.
* "Категории" – кнопка для перехода к выбору категорий связанных с ремонтируемой техникой.
* "Отчеты" – кнопка для создания отчетов.
* "Создать" – кнопка для создания новой заявки на ремонт.
* Центральная часть окна содержит таблицу заявок с колонками:
* Название заявки (например, "Ремонт телефона").
* Статус заявки (например, "В работе").

Пользователь может видеть и управлять своими заявками с помощью данного интерфейса. По мере взаимодействия с программой, нажимая на соответствующие кнопки, пользователь может переходить к другим разделам программы, таким как "Категории" или "Отчеты".

При выборе пункта меню «Личный кабинет» открывается окно (рис. 16)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 16

Страница "Информация о мастере" состоит из следующих элементов:

Формы для ввода и редактирования данных мастера, включая следующие поля:

* Логин (например, " vladmed").
* Фамилия (например, " Медведев").
* Имя (например, " Владислав").
* Отчество (например, " Александрович").
* Пол (например, "Мужской").

Три кнопки в нижней части окна:

* "Редактировать" – кнопка для сохранения изменений в данных мастера.
* "Назад" – кнопка для возврата на предыдущее окно (главное окно программы).
* "Выход" – кнопка для выхода из учетной записи.

Пользователь может редактировать свои личные данные и сохранять изменения с помощью кнопки "Редактировать". Кнопка "Назад" позволяет вернуться на главное окно программы, а кнопка "Выход" завершает сессию пользователя.

При выборе пункта меню «Редактировать» открывается окно (рис. 17)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 17

Страница "Редактирование мастера" состоит из следующих элементов:

* Заголовок в верхней части окна "Редактирование мастера".
* Формы для редактирования данных мастера, включая следующие поля:
* Логин (например, " vladmed").
* Фамилия (например, " Медведев").
* Имя (например, " Владислав").
* Отчество (например, " Александрович").
* Пол (например, "Мужской").

Две кнопки в нижней части окна:

* "Сохранить" – кнопка для сохранения изменений и возврата на главную страницу
* "Отмена" – кнопка для отмены изменений и возврата на страницу "Информация о мастере"

При выборе пункта главной страницы меню «Категории» открывается окно (рис. 18)

Изображение выглядит как текст, снимок экрана, Прямоугольник, дизайн

Автоматически созданное описание

Рисунок 18

В левой части окна представлены три кнопки: "Мои заявки", "Категории" и "Отчеты". Эти кнопки позволяют пользователю перейти в соответствующие разделы для управления заявками на ремонт, категориями и формирования отчетов.

В центральной части окна расположены три больших кнопки: "Ремонт", "Клиенты" и "Типы". Эти кнопки обеспечивают доступ к различным функциональным областям программы, связанным с управлением ремонтными работами, информацией о клиентах и типами ремонтируемой техники.

Описание элементов окна:

* "Мои заявки" - нажатием этой кнопки пользователь переходит в раздел управления своими заявками на ремонт.
* "Категории" - нажатием этой кнопки открывается раздел управления категориями.
* "Отчеты" - эта кнопка позволяет перейти к разделу для формирования и просмотра отчетов.
* "Ремонт" - нажатием на эту кнопку пользователь переходит в раздел управления информацией о ремонте.
* "Клиенты" - эта кнопка открывает раздел для управления информацией о клиентах.
* "Типы" - нажатием на эту кнопку открывается раздел для управления типами ремонтируемой техники.

Все представленные элементы обеспечивают интуитивно понятный интерфейс для эффективного управления заявками и информацией, связанной с ремонтными работами.

При выборе пункта «Ремонт» страницы «Категории» открывается окно (рис. 19)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 19

В левой части окна расположены четыре кнопки: "Мои заявки", "Категории", "Отчеты" и "Создать". Эти кнопки позволяют пользователю переходить в соответствующие разделы для управления заявками, категориями, отчетами и создания новых записей.

В центральной части окна находится заголовок "Отчеты" и под ним таблица со списком отчетов.

В правой части окна расположен основной список отчетов, который может быть прокручен для просмотра всех доступных отчетов. Поле ввода текста над таблицей позволяет фильтровать или искать конкретные отчеты по ключевым словам.

На верхней панели окна указано название программы "Ремонт орг техники", которое помогает пользователю ориентироваться в приложении.

Это окно предоставляет пользователю удобный интерфейс для управления различными аспектами ремонта оргтехники, обеспечивая доступ к основным функциям через интуитивно понятные элементы управления.

При выборе пункта «Создать» страницы открывается окно (рис. 20)

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 20

Основная часть окна состоит из следующих элементов:

Поле для ввода названия категории с меткой "Название:", в котором указано текущее название категории - "Ремонт экрана".

Поле для ввода описания категории с меткой "Описание:", в котором также указано текущее описание - "Ремонт экрана".

В нижней части окна расположены три кнопки:

"Ок" - кнопка, предназначенная для сохранения изменений, внесенных в поля "Название" и "Описание".

"Отмена" - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие записи, а также возвращаться к предыдущему окну, если изменения не требуются.

При выборе ремонта на листе ремонтов открывается окно (рис. 21)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 21

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия категории с меткой "Название:", в котором указано текущее название категории - "Ремонт экрана".
* Поле для ввода описания категории с меткой "Описание:", в котором также указано текущее описание - "Ремонт экрана".

В нижней части окна расположены три кнопки:

* "Редактировать" - кнопка, предназначенная для сохранения изменений, внесенных в поля "Название" и "Описание".
* "Удалить" - кнопка для удаления выбранной категории.
* "Отмена" - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие записи, а также возвращаться к предыдущему окну, если изменения не требуются.

При выборе пункта «Редактировать» открывается окно (рис. 22)

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 22

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия с меткой "Название:". Это поле позволяет пользователю ввести название новой записи.
* Поле для ввода описания с меткой "Описание:". Это поле предназначено для ввода описания новой записи.

В нижней части окна расположены две кнопки:

* "Ок" - кнопка, предназначенная для сохранения введенной информации и добавления новой записи.
* "Отмена" - кнопка для отмены введенных изменений и закрытия окна без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять новые записи, а также отменять действия при необходимости.

При выборе пункта «Клиенты» страницы «Категории» открывается окно (рис. 23)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 23

Основная часть окна состоит из следующих элементов:

* Заголовок "Клиенты".
* Поле для ввода и поиска клиента с меткой "Клиенты:".
* Список клиентов, отображаемый в табличном виде
* В нижней части окна расположена кнопка "Создать", которая позволяет добавить нового клиента в систему.

Описание функциональности:

* "Мои заявки" - раздел для просмотра и управления заявками пользователя на ремонт.
* "Категории" - раздел для добавления, редактирования и удаления категорий ремонтируемой техники.
* "Отчеты" - раздел, в котором пользователь может просматривать, создавать и управлять отчетами о ремонтах.
* "Создать" - кнопка для создания новой записи в текущем разделе (например, нового клиента или заявки).

При выборе клиента на листе клиентов открывается окно (рис. 24)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 24

Основная часть окна состоит из следующих элементов:

* Поле для ввода имени клиента с меткой "Имя:", в котором указано текущее имя клиента - "Виктор Корнеплод".
* Поле для ввода номера телефона клиента с меткой "Номер:", в котором указан текущий номер - "88005553535".

В нижней части окна расположены три кнопки:

* "Редактировать" - кнопка, предназначенная для сохранения изменений, внесенных в поля "Имя" и "Номер".
* "Удалить" - кнопка для удаления информации о клиенте.
* "Отмена" - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие записи о клиентах, а также отменять действия при необходимости.

При выборе пункта «Редактировать» открывается окно (рис. 25)

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 25

Основная часть окна состоит из следующих элементов:

* Поле для ввода имени клиента с меткой "Имя:", в котором указано текущее имя клиента - "Виктор Корнеплод".
* Поле для ввода номера телефона клиента с меткой "Номер:", в котором указан текущий номер - "88005553535".

В нижней части окна расположены две кнопки:

* "Ок" - кнопка, предназначенная для сохранения изменений, внесенных в поля "Имя" и "Номер".
* "Отмена" - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие записи о клиентах, а также отменять действия при необходимости.

При выборе пункта «Создать» страницы открывается окно (рис. 26)

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 26

Основная часть окна состоит из следующих элементов:

* Поле для ввода имени клиента с меткой "Имя:". Это поле предназначено для ввода имени нового клиента.
* Поле для ввода номера телефона клиента с меткой "Номер:". Это поле предназначено для ввода номера телефона нового клиента.

В нижней части окна расположены две кнопки:

* "Ок" - кнопка, предназначенная для сохранения введенной информации и добавления нового клиента.
* "Отмена" - кнопка для отмены введенных данных и закрытия окна без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять информацию о новых клиентах, а также отменять действия при необходимости.

При выборе пункта «Ремонт» страницы «Категории» открывается окно (рис. 27)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 27

Основная часть окна состоит из следующих элементов:

* Заголовок "Типы".
* Поле для ввода и поиска типов с меткой "Типы:".
* Список типов.

В нижней части окна расположена кнопка "Создать", которая позволяет добавить новый тип в систему.

Элементы интерфейса:

* "Мои заявки" - кнопка для перехода в раздел управления заявками на ремонт.
* "Категории" - кнопка для перехода в раздел управления категориями ремонтируемой техники.
* "Отчеты" - кнопка для перехода в раздел формирования и просмотра отчетов.
* "Создать" - кнопка для создания новой записи в текущем разделе (например, нового типа или заявки).

Это окно предоставляет пользователю удобный интерфейс для управления различными типами ремонтируемой техники, обеспечивая доступ к основным функциям через интуитивно понятные элементы управления.

При выборе клиента на листе клиентов открывается окно (рис. 28)

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 28

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия типа с меткой "Название:", в котором указано текущее название типа - "Тип 1".
* Поле для ввода описания типа с меткой "Описание:". Это поле предназначено для ввода описания текущего типа (в настоящее время пустое).

В нижней части окна расположены три кнопки:

* "Редактировать" - кнопка, предназначенная для сохранения изменений, внесенных в поля "Название" и "Описание".
* "Удалить" - кнопка для удаления выбранного типа.
* "Отмена" - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать или удалять существующие типы, а также отменять действия при необходимости.

При выборе пункта «Редактировать» страницы открывается окно (рис. 29)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 29

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия типа с меткой "Название:", в котором указано текущее название типа - "Тип 1".
* Поле для ввода описания типа с меткой "Описание:". Это поле предназначено для ввода описания текущего типа.

В нижней части окна расположены две кнопки:

* "Ок" - кнопка, предназначенная для сохранения изменений, внесенных в поля "Название" и "Описание".
* "Отмена" - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко редактировать существующие типы, а также отменять действия при необходимости.

При выборе пункта «Создать» страницы открывается окно (рис. 30)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 30

Основная часть окна состоит из следующих элементов:

* Поле для ввода названия типа с меткой "Название:". Это поле предназначено для ввода названия нового типа.
* Поле для ввода описания типа с меткой "Описание:". Это поле предназначено для ввода описания нового типа.

В нижней части окна расположены две кнопки:

* "Ок" - кнопка, предназначенная для сохранения введенной информации и добавления нового типа.
* "Отмена" - кнопка для отмены введенных данных и закрытия окна без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять информацию о новых типах ремонтируемой техники, а также отменять действия при необходимости.

При выборе заявки на листе страницы «Заявки» клиентов открывается окно (рис. 31)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 31

Основная часть окна состоит из следующих элементов:

Поля для ввода информации о заявке с метками:

* "Имя:" - поле для ввода названия заявки, в данном случае "Ремонт телефона".
* "Цена:" - поле для ввода стоимости ремонта, в данном случае "1000.00".
* "Клиент:" - поле для ввода имени клиента, в данном случае "Виктор Корнеплод".
* "Ремонт:" - поле для ввода типа ремонта, в данном случае "Ремонт 1".
* "Тип:" - поле для ввода типа техники, в данном случае "Тип 1".
* "Статус:" - поле для ввода текущего статуса заявки, в данном случае "В работе".
* "Дата:" - поле для ввода даты заявки, в данном случае "4.6.2024".

В нижней части окна расположены три кнопки:

* "Редактировать" - кнопка, предназначенная для сохранения изменений, внесенных в поля.
* "Удалить" - кнопка для удаления выбранной заявки.
* "Назад" - кнопка для отмены внесенных изменений и возврата к предыдущему окну без сохранения изменений.

Эти элементы позволяют пользователю легко просматривать, редактировать или удалять существующие заявки, а также возвращаться к предыдущему окну при необходимости.

При выборе пункта «Редактировать» страницы открывается окно (рис. 32)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 32

Основная часть окна состоит из следующих элементов:

Поля для ввода информации о заявке с метками:

* "Имя:" - поле для ввода названия заявки, в данном случае "Ремонт телефона".
* "Цена:" - поле для ввода стоимости ремонта, в данном случае "1000.00".
* "Клиент:" - выпадающее поле для выбора клиента, в данном случае "Виктор Корнелплод", и кнопка "Добавить" для добавления нового клиента.
* "Ремонт:" - выпадающее поле для выбора типа ремонта, в данном случае "Ремонт 1", и кнопка "Добавить" для добавления нового типа ремонта.
* "Тип:" - выпадающее поле для выбора типа техники, в данном случае "Тип 1", и кнопка "Добавить" для добавления нового типа техники.
* "Статус:" - выпадающее поле для выбора текущего статуса заявки, в данном случае "В работе".

В нижней части окна расположены две кнопки:

* "Сохранить" - кнопка, предназначенная для сохранения введенной информации и добавления новой заявки.
* "Назад" - кнопка для отмены внесенных данных и возврата к предыдущему окну без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять информацию о новых заявках, а также возвращаться к предыдущему окну при необходимости.

При выборе пункта «Создать» страницы открывается окно (рис. 33)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 33

Основная часть окна состоит из следующих элементов:

Поля для ввода информации о заявке с метками:

* "Имя:" - поле для ввода названия заявки.
* "Цена:" - поле для ввода стоимости ремонта.
* "Клиент:" - выпадающее поле для выбора клиента с кнопкой "Добавить" для добавления нового клиента.
* "Ремонт:" - выпадающее поле для выбора типа ремонта с кнопкой "Добавить" для добавления нового типа ремонта.
* "Тип:" - выпадающее поле для выбора типа техники с кнопкой "Добавить" для добавления нового типа техники.
* "Статус:" - выпадающее поле для выбора текущего статуса заявки.

В нижней части окна расположены две кнопки:

* "Сохранить" - кнопка, предназначенная для сохранения введенной информации и добавления новой заявки.
* "Назад" - кнопка для отмены внесенных данных и возврата к предыдущему окну без сохранения информации.

Эти элементы позволяют пользователю легко вводить и сохранять информацию о новых заявках, а также возвращаться к предыдущему окну при необходимости.

Выходными данными являются: сформированный отчет по заявкам.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 34

**3.7 Описание структуры выходной информации**

Выходными данными являются списки заявок отфильтрованные по статусу.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рис.35 Список заявок

**4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Назначение программного средства**

Назначением программного средства является обработка заявки на ремонт организационной техники.

**4.2 Условия выполнения программного средства**

Условия, при соблюдении которых возможно выполнение работы с программным продуктом:

-Компьютер с установленной на нём «Org-Tech» и PostgreSQL.

Подготовительные действия:

-На иконке «Org-Tech» рабочего стола произвести двойной щелчок левой кнопкой мыши.

**4.3 Эксплуатация программного средства**

Этот раздел содержит обзор приложений «Org-Tech» и ее основных параметров.

Прежде чем приступить к процессу установки, нужно убедиться, что Вы начали сеанс работы Windows с правами администратора.

Создайте в желаемом месте папку, в которую будет установлена «Org-Tech». Скопируйте в эту папку, с носителя с дистрибутивом «Org-Tech», все содержимое.

Вводные сведения об интерфейсе пользователя

В окнах «Org-Tech» используются разнообразные элементы интерфейса, такие как:

однострочные текстовые редакторы;

текстовые таблицы;

меню;

управляющие кнопки;

кнопки работы с таблицами базы данных.

Однострочные текстовые редакторы предназначены для ввода произвольного текста длиной не более 255 символов.

Текстовые таблицы используются для отображения табличных данных.

Управляющие кнопки предназначены для выполнения указанных на них действий по нажатию кнопки.

Кнопки работы с таблицами базы данных предназначены для выполнения определенных операций с таблицей (рис. 35 слева на право):

* Добавить запись в таблицу;
* Удалить запись из таблицы;
* Редактировать запись.

Действия этих кнопок приведены в таблице 1:

Таблица-1 Действие кнопок

|  |  |
| --- | --- |
| **Кнопка** | **Назначение** |
| Изображение выглядит как Шрифт, текст, логотип, Графика  Автоматически созданное описание | Предназначена для добавления новой записи в таблицу. |
| Изображение выглядит как текст, Шрифт, логотип, Графика  Автоматически созданное описание | Предназначена для удаления записи из таблицы. |
| Изображение выглядит как текст, Шрифт, Графика, логотип  Автоматически созданное описание | Предназначена для подтверждения изменений, сделанных в таблице. |

**4.4 Сообщения пользователю**

Для уверенной и комфортной работы пользователю важно понимать поведение программы, а программе, в свою очередь, поведение пользователя. Для информирования и уточнения намерений пользователей в программах используются сообщения. Сообщения пользователю представлены на рис. 36, 37.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рис.36 Сообщение Решистрация прошла хорошо

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рис.37 Сообщение Заявка удалена

**5. РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Обоснование необходимого количества тестов**

Для разработанного алгоритма был построен граф.

Рисунок 39

Необходимое количество тестов определяется по формуле V(G) = E-N+2, где E – это число ребер, а N–это число вершин данного графа. Согласно представленному графу E=9, N=7. Далее по формуле рассчитаем необходимое количество тестовых сценариев (тест-кейсов).

V(G)=9-7+2=4.

Для данного программного средства необходимо разработать 4 тестовых сценария.

Так как каждый сценарий имеет два варианта завершения, то необходимо 8 тестовых прогонов.

**5.2 Описание тестовых пакетов**

Таблица 1 – Тестирование корректного добавления информации о Клиенте

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации о Клиенте | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем Окно Панель клиента | | Панель открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Bio = Иванов Иван Иванович,  Number=88005553535 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |

Таблица 2 – Тестирование некорректного добавления информации о Клиенте

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации о Клиенте | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем Окно Панель клиента | | Панель открыта |  |

Продолжение Таблицы 2

|  |  |  |
| --- | --- | --- |
| **Шаги теста:** |  |  |
| Нажмите кнопку добавить  Заполните форму  Bio =,  Number= |  |  |
| Нажмите кнопку «Сохранить» | Выведено сообщение об ошибке. Окно закрыто | **Тест пройден** |

Таблица 3 – Тестирование корректного добавления информации о Ремонте

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации о Ремонте | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем Окно Панель ремонта | | Окно Панель ремонта открыто |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Name = name,  Description=description | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |

Таблица 4 – Тестирование некорректного добавления информации о Ремонте

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации о Ремонте | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем Окно Панель ремонта | | Окно Панель ремонта открыто |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Name =,  Description= | |  |  |
| Нажмите кнопку «Сохранить» | | Выведено сообщение об ошибке. Окно закрыто | **Тест пройден** |

Таблица 5 – Тестирование корректного добавления информации о Типе

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Наличие файла базы данных | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем Окно Панель типа | | Окно Панель типа открыто |  |
| **Шаги теста:** | |  |  |

Продолжение Таблицы 5

|  |  |  |
| --- | --- | --- |
| Нажмите кнопку добавить  Заполните форму  Name = name,  Description=description | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | Данные успешно сохранены | **Тест пройден** |

Таблица 6 – Тестирование некорректного добавления информации о Типе

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации о Типе | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем Окно Панель типа | | Окно Панель типа открыто |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Name =,  Description= | |  |  |
| Нажмите кнопку «Сохранить» | | Выведено сообщение об ошибке. Программа закрыта. | **Тест пройден** |

Тест №7

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование завершения программы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Программа запущена. | |  |  |
| **Шаги теста:** | |  |  |
| Была нажата кнопка выхода из программы. | | Программа закрыта. Все данные успешно сохранены. | **Тест пройден** |

Тест №8

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование завершения программы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Программа запущена. | |  |  |
| **Шаги теста:** | |  |  |
| Кнопка выхода из программы не нажата. | | Программа функционирует в штатном режиме | **Тест пройден** |

**5.3 Составление отчета о тестировании**

Для форм программы были созданы и проведены тесты, результаты сведены в таблицу2:

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Прохождение | |
|  | Число ТК | пройдено | не пройдено |
| Наличие файла конфигурации | 2 | 2 | 0 |
| Корректность структуры файла конфигурации | 2 | 2 | 0 |
| Тестирование завершения программы | 2 | 2 | 0 |
| ИТОГО | 8 | 8 | 0 |

По результатам тестирования проблем выявлено не было.

**ЗАКЛЮЧЕНИЕ**

**Оценка качества программного средства с помощью метрик**

Оценка качества программного средства проводится исходя из стандарта оценки качества ISО 9126.

На компьютере следующей конфигурации:

Установленная операционная система – Windows 7 Ultimate 32бит.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 2 Gb

Таблица 29 – Требования к количественным характеристикам качества программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Надежность** |  | | |
| *Завершенность:*  *-* наработка на отказ при отсутствии рестарта; | Часы | 10 | 10 |
| *Устойчивость:*  - относительные ресурсы на обеспечение надежности и рестарта. | % | 0,5 | 0,5 |
| *Восстанавливаемость:*  -длительность восстановления. | Минуты | 2 | 2 |
| *Доступность-готовность:*  -относительное время работоспособного функционирования. | Вероятность | 0,998 | 0,998 |
| **Эффективность** |  | | |
| *Временная эффективность:*  *-*время отклика - получение результатов на типовое задание;  - пропускная способность число типовых заданий, исполняемых в единицу времени. | Секунды  Число в минуту | 3  60 | 3  60 |
| *Используемость ресурсов:*  *-* относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства. |  | 0,001 | 0,001 |

Таблица 30 – Требования к качественным характеристикам программного средства.

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Практичность** |  | | |
| *Простота использования:*  **-**среднее время ввода заданий;  -среднее время отклика на задание. | Секунды  Секунды | 10  3 | 10  3 |
| *Изучаемость :*  - трудоемкость изучения применения ПС;  -продолжительность изучения;  -объем эксплуатационной документации; | Чел.-часы  Часы  Страницы | 5  2  15 | 5  2  15 |
| **Сопровождаемость** |  | | |
| *Изменяемость:*  -трудоемкость подготовки изменений;  -длительность подготовки изменений. | Чел.-часы  Часы | 5  5 | 5  5 |
| *Тестируемость:*  *-трудоемкость тестирования изменений;*  *- длительность тестирования изменений.* | Чел.-часы  Часы | 3  3 | 3  3 |
| **Мобильность** |  | | |
| *Адаптируемость:*  - трудоемкость адаптации;  -длительность адаптации. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Простота установки:*  -трудоемкость инсталляции;  -длительность инсталляции. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Замещаемость:*  -трудоемкость замены компонентов;  - длительность замены компонентов. | Чел.-часы  Часы | 0,2  0,2 | 0,2  0,2 |

Таблица 31 – Сложность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип объекта | Количество | Вес | | | Итого |
|  |  | Простой | Средний | Сложный |  |
| Экран | 9 | x1 | x2 | **x3** | =27 |
| Отчет | 2 | x2 | **x5** | x8 | =10 |

Таблица 32 – Сложность экрана

|  |  |  |  |
| --- | --- | --- | --- |
| Экраны | Количество Категорических (клт) таблиц данных | | |
| Количество представлений | Всего <4 | **Всего <8** | Всего >8 |
|  | (<2 срв, <3 клт) | **(2-3 срв, 3-5 клт)** | (>3 срв, >5 клт) |
| <3 | Простой | Простой | Средний |
| 3-7 | Простой | Средний | Сложный |
| **>8** | Средний | **Сложный** | Сложный |

Таблица 33 – Сложность отчета

|  |  |  |  |
| --- | --- | --- | --- |
| Отчеты | Количество Категорических (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | Всего >8 |
|  | (<2 срв, <3 клт) | **(2-3 срв, 3-5 клт)** | (>3 срв, >5 клт) |
| 0 или 1 | Простой | Простой | Средний |
| **2 или 3** | Простой | **Средний** | Сложный |
| >4 | Средний | Сложный | Сложный |

Производительность = 4 тысяч строк кода/2 месяца= 2 тысяч строк кода/месяц

Качество = 2 ошибки /4 тысяч строк кода=0,5 ошибок/тысяч строк кода

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Гордон, Л. Изучаем Java: учебный курс. - Спб.: Питер, 2022. - 512с.
2. Гудрич, М., Тамассия, Р. Алгоритмы и структуры данных на Java. - М.: Вильямс, 2022. - 684 с.
3. Хорстманн К. Основы программирования на Java/К.Хорстманн.-М.:Лори, 2022.-862 с.
4. Блош, Д., Смит, Д. Эффективное программирование на Java. - М.: Русская редакция, 2022, 604 с.
5. Джонсон, Р., Джонс, Т. Разработка систем с использованием PostgreSQL и Java. Учимся на примерах. - М.: СОЛОН-ПРЕСС, 2022, 464 с., СД.
6. Иванов В.В. Java: создание приложений для Windows. - Мн.: Харвест, 2022 - 384 с.
7. Петров А.В., Сидоров Г.В. Визуальное проектирование приложений JavaFX. - М.: КУДИЦ-ОБРАЗ, 2022 - 512 с.
8. ГОСТ 19.201-78. ЕСПД. Техническое задание. Требования к содержанию и оформлению.
9. ГОСТ 19.102-77. ЕСПД. Стадии разработки.
10. ГОСТ 19.505-79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению.
11. ГОСТ 19.002-80. ЕСПД. Схемы алгоритмов и программ. Правила выполнения.
12. ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
13. Василий Павлов. Основы программирования на Java. 2016 г. [Электронный ресурс http://mexalib.com/view/12345]
14. Сергей Кузнецов. Основы программирования в JavaFX. 2017 г. [Электронный ресурс http://www.litres.ru/sergey-kuznetsov/osnovy-programmirovaniya-v-javafx-2022-2]
15. Сидоров А.В., Петров Г.В. Визуальное проектирование приложений JavaFX. - М.: КУДИЦ-ОБРАЗ, 2017. [Электронный ресурс http://www.twirpx.com/file/54321]
16. Марк Вудс. Программирование на Java и PostgreSQL. 2022 г. [Электронный ресурс http://rutracker.org/forum/viewtopic.php?t=9876543]

**Приложение А**

Разработка интегрированного программного модуля «Фирма по обслуживанию организационной техники»

Текст кода программного средства

460.ЗМК.00019-01 12 01

**Текст кода программного средства**

package project.orgtech;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.geometry.Rectangle2D;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Screen;

import javafx.stage.Stage;

import lombok.Getter;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import project.orgtech.config.dbConfig;

public class StartApplication extends Application {

@Getter

private static AnnotationConfigApplicationContext context;

public static void main(String[] args) {

// Initialize Spring context

context = new AnnotationConfigApplicationContext(dbConfig.class);

launch(args);

}

@Override

public void start(Stage primaryStage) throws Exception {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/project/orgtech/auth/login-view.fxml"));

loader.setControllerFactory(context::getBean); // Use Spring context to load controllers

Parent root = loader.load();

primaryStage.setTitle("Вход");

primaryStage.setScene(new Scene(root));

Rectangle2D primaryScreenBounds = Screen.getPrimary().getVisualBounds();

// Устанавливаем размеры окна на размеры экрана

primaryStage.setX(primaryScreenBounds.getMinX());

primaryStage.setY(primaryScreenBounds.getMinY());

primaryStage.setWidth(primaryScreenBounds.getWidth());

primaryStage.setHeight(primaryScreenBounds.getHeight());

primaryStage.setMaximized(true);

primaryStage.show();

}

@Override

public void stop() throws Exception {

super.stop();

context.close(); // Close Spring context when application stops

}

}

package project.orgtech.frontController.auth;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import project.orgtech.utils.AuthManager;

import project.orgtech.models.Master;

import project.orgtech.service.auth.MasterService;

import project.orgtech.utils.FxmlView;

import project.orgtech.utils.SceneManager;

import java.io.IOException;

@Controller

public class LoginController {

@Autowired

private MasterService masterService; // Внедрено Spring

@Autowired

private SceneManager sceneManager;

@FXML

private TextField loginText;

@FXML

private PasswordField passwordText;

@FXML

private Button loginButton;

@FXML

private Button registrationButton;

@FXML

private void handleLoginButton(ActionEvent event) throws IOException {

Master master = masterService.findByLogin(loginText.getText());

if (master != null && master.getPassword().equals(passwordText.getText())) {

AuthManager.setMaster(master);

sceneManager.openScene(loginButton, FxmlView.MAIN);

} else {

if(master != null){

sceneManager.showAlert("Ошибка","Пароль неверный");

}

else{

sceneManager.showAlert("Ошибка","Логин неверный");

}

}

}

@FXML

private void handleRegistrationButton(ActionEvent event) throws IOException {

sceneManager.openScene(registrationButton, FxmlView.REGISTRATION);

}

@FXML

void initialize() {

// Логика инициализации, если необходимо

}

}<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.PasswordField?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="1080.0" prefWidth="1920.0" style="-fx-background-color: #2E3348;" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="project.orgtech.frontController.auth.LoginController">

<children>

<AnchorPane layoutY="143.0" prefHeight="850.0" prefWidth="1920.0" style="-fx-background-color: #fafafa;">

<children>

<TextField fx:id="loginText" layoutX="537.0" layoutY="259.0" prefHeight="82.0" prefWidth="846.0" promptText="Логин">

<font>

<Font size="32.0" />

</font>

</TextField>

<PasswordField fx:id="passwordText" layoutX="537.0" layoutY="433.0" prefHeight="82.0" prefWidth="846.0" promptText="Пароль">

<font>

<Font size="32.0" />

</font>

</PasswordField>

<Button fx:id="loginButton" layoutX="758.0" layoutY="665.0" mnemonicParsing="false" onAction="#handleLoginButton" prefHeight="68.0" prefWidth="432.0" style="-fx-background-color: #F39c53;" text="Войти">

<font>

<Font name="Comic Sans MS" size="32.0" />

</font>

</Button>

</children>

</AnchorPane>

<Button fx:id="registrationButton" layoutX="1410.0" layoutY="35.0" mnemonicParsing="false" onAction="#handleRegistrationButton" prefHeight="68.0" prefWidth="432.0" style="-fx-background-color: #F39c53;" text="Зарегистрироваться">

<font>

<Font name="Comic Sans MS" size="32.0" />

</font>

</Button>

<Label layoutX="879.0" layoutY="27.0" prefHeight="84.0" prefWidth="306.0" text="Авторизация" textAlignment="CENTER" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="44.0" />

</font>

</Label>

</children>

</AnchorPane><?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.String?>

<?import javafx.collections.FXCollections?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ComboBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.PasswordField?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="1080.0" prefWidth="1920.0" style="-fx-background-color: #2E3348;" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="project.orgtech.frontController.auth.RegistrationController">

<children>

<AnchorPane layoutY="143.0" prefHeight="850.0" prefWidth="1920.0" style="-fx-background-color: #fafafa;">

<children>

<TextField fx:id="firstNameText" layoutX="450.0" layoutY="150.0" prefHeight="82.0" prefWidth="338.0" promptText="Фамилия">

<font>

<Font size="32.0" />

</font>

</TextField>

<TextField fx:id="middleNameText" layoutX="805.0" layoutY="150.0" prefHeight="82.0" prefWidth="338.0" promptText="Имя">

<font>

<Font size="32.0" />

</font>

</TextField>

<TextField fx:id="lastNameText" layoutX="1161.0" layoutY="150.0" prefHeight="82.0" prefWidth="338.0" promptText="Отчество">

<font>

<Font size="32.0" />

</font>

</TextField>

<TextField fx:id="loginText" layoutX="551.0" layoutY="261.0" prefHeight="82.0" prefWidth="846.0" promptText="Логин">

<font>

<Font size="32.0" />

</font>

</TextField>

<PasswordField fx:id="passwordText" layoutX="551.0" layoutY="368.0" prefHeight="82.0" prefWidth="846.0" promptText="Пароль">

<font>

<Font size="32.0" />

</font>

</PasswordField>

<ComboBox fx:id="genderComboBox" layoutX="551.0" layoutY="475.0" prefHeight="82.0" prefWidth="846.0" promptText="Выберите пол">

<items>

<FXCollections fx:factory="observableArrayList">

<String fx:value="Мужской" />

<String fx:value="Женский" />

</FXCollections>

</items>

</ComboBox>

<Button fx:id="registerButton" layoutX="589.0" layoutY="648.0" mnemonicParsing="false" onAction="#handleRegisterButton" prefHeight="68.0" prefWidth="362.0" style="-fx-background-color: #F39c53;" text="Зарегистрироваться">

<font>

<Font name="Comic Sans MS" size="32.0" />

</font>

</Button>

<Button fx:id="backButton" layoutX="993.0" layoutY="648.0" mnemonicParsing="false" onAction="#handleBackButton" prefHeight="68.0" prefWidth="362.0" style="-fx-background-color: #F39c53;" text="Назад">

<font>

<Font name="Comic Sans MS" size="32.0" />

</font>

</Button>

</children>

</AnchorPane>

<Label layoutX="735.0" layoutY="27.0" prefHeight="84.0" prefWidth="1030.0" text="Авторизация" textAlignment="CENTER" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="44.0" />

</font>

</Label>

<Label layoutX="879.0" layoutY="27.0" prefHeight="84.0" prefWidth="306.0" text="Регистрация" textAlignment="CENTER" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="44.0" />

</font>

</Label>

</children>

</AnchorPane>package project.orgtech.frontController.auth;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import project.orgtech.utils.AuthManager;

import project.orgtech.models.Master;

import project.orgtech.service.auth.MasterService;

import project.orgtech.utils.FxmlView;

import project.orgtech.utils.SceneManager;

import java.io.IOException;

@Controller

public class LoginController {

@Autowired

private MasterService masterService; // Внедрено Spring

@Autowired

private SceneManager sceneManager;

@FXML

private TextField loginText;

@FXML

private PasswordField passwordText;

@FXML

private Button loginButton;

@FXML

private Button registrationButton;

@FXML

private void handleLoginButton(ActionEvent event) throws IOException {

Master master = masterService.findByLogin(loginText.getText());

if (master != null && master.getPassword().equals(passwordText.getText())) {

AuthManager.setMaster(master);

sceneManager.openScene(loginButton, FxmlView.MAIN);

} else {

if(master != null){

sceneManager.showAlert("Ошибка","Пароль неверный");

}

else{

sceneManager.showAlert("Ошибка","Логин неверный");

}

}

}

@FXML

private void handleRegistrationButton(ActionEvent event) throws IOException {

sceneManager.openScene(registrationButton, FxmlView.REGISTRATION);

}

@FXML

void initialize() {

// Логика инициализации, если необходимо

}

}package project.orgtech.frontController.auth;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import project.orgtech.service.gender.GenderService;

import project.orgtech.utils.AuthManager;

import project.orgtech.models.Master;

import project.orgtech.models.Gender;

import project.orgtech.service.auth.MasterService;

import project.orgtech.utils.FxmlView;

import project.orgtech.utils.SceneManager;

import java.io.IOException;

@Controller

public class RegistrationController {

@Autowired

private MasterService masterService;

@Autowired

private GenderService genderService;

@Autowired

private SceneManager sceneManager;

@FXML

private TextField loginText;

@FXML

private PasswordField passwordText;

@FXML

private TextField firstNameText;

@FXML

private TextField lastNameText;

@FXML

private TextField middleNameText;

@FXML

private ComboBox<String> genderComboBox;

@FXML

private Button registerButton;

@FXML

private Button backButton;

@FXML

private void handleRegisterButton(ActionEvent event) throws IOException {

String login = loginText.getText();

String password = passwordText.getText();

String firstName = firstNameText.getText();

String lastName = lastNameText.getText();

String middleName = middleNameText.getText();

String genderName = genderComboBox.getId();

long genderId = 0;

if (genderName != null && genderName.matches("\\d+")) {

genderId = Long.parseLong(genderName);

}

if (login.isEmpty() || password.isEmpty() || firstName.isEmpty() || lastName.isEmpty() || genderName == null) {

sceneManager.showAlert("Ошибка", "Все поля должны быть заполнены");

return;

}

Master newMaster = new Master();

newMaster.setLogin(login);

newMaster.setPassword(password);

newMaster.setFirstName(firstName);

newMaster.setLastName(lastName);

newMaster.setMiddleName(middleName);

Gender gender = genderService.getById(genderId + 1);

//gender.setName(genderName);

newMaster.setGender(gender);

masterService.add(newMaster);

sceneManager.showAlert("Успех", "Регистрация прошла успешно");

AuthManager.setMaster(newMaster);

sceneManager.openScene(registerButton, FxmlView.MAIN);

}

@FXML

private void handleBackButton(ActionEvent event) throws IOException {

sceneManager.openScene(backButton, FxmlView.LOGIN);

}

@FXML

void initialize() {

//genderComboBox.getItems().addAll("Мужской", "Женский");

}

}package project.orgtech.frontController;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.geometry.Pos;

import javafx.scene.control.Button;

import javafx.scene.control.ListCell;

import javafx.scene.control.ListView;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Priority;

import javafx.scene.text.Text;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import project.orgtech.models.Application;

import project.orgtech.service.application.ApplicationService;

import project.orgtech.utils.FxmlView;

import project.orgtech.utils.SceneManager;

import java.io.IOException;

import java.util.List;

@Controller

public class MainController {

@Autowired

private SceneManager sceneManager;

@Autowired

private ApplicationService applicationService;

@FXML

private ListView<Application> ApplicationListView;

@FXML

private Button CategoryButton;

@FXML

private Button CreateButton;

@FXML

private Button MyApplicationButton;

@FXML

private Button ReportsButton;

@FXML

private Button MasterArea;

private ObservableList<Application> applicationObservableList;

@FXML

private void handleMyApplicationButton(ActionEvent event) throws IOException {

sceneManager.openScene(MyApplicationButton, FxmlView.MAIN);

}

@FXML

private void handleCategoryButton(ActionEvent event) throws IOException {

sceneManager.openScene(CategoryButton, FxmlView.CATEGORY);

}

@FXML

private void handleReportsButton(ActionEvent event) throws IOException {

sceneManager.getReports(ReportsButton);

}

@FXML

private void handleCreateButton(ActionEvent event) throws IOException {

sceneManager.openScene(CreateButton, FxmlView.APPLICATION\_ADD);

}

@FXML

private void handleMasterArea(ActionEvent event) throws IOException {

sceneManager.openScene(CreateButton, FxmlView.MASTER\_INFO);

}

@FXML

public void initialize() {

// Load data from the database

List<Application> applications = loadDataFromDatabase();

// Sort applications: In Progress first, then Completed

applications.sort((a1, a2) -> {

if (1L==a1.getStatus().getId() && 2L==a2.getStatus().getId()) {

return -1;

} else if (2L==a2.getStatus().getId() && 1L==a1.getStatus().getId()) {

return 1;

} else {

return 0;

}

});

// Convert list to ObservableList

applicationObservableList = FXCollections.observableArrayList(applications);

// Set ObservableList to ListView

ApplicationListView.setItems(applicationObservableList);

// Set ListView cell factory for custom rendering

ApplicationListView.setCellFactory(param -> new ListCell<Application>() {

@Override

protected void updateItem(Application application, boolean empty) {

super.updateItem(application, empty);

if (empty || application == null) {

setText(null);

setGraphic(null);

} else {

HBox hBox = new HBox(10); // Spacing between elements

Text nameText = new Text(application.getName());

Text statusText = new Text(application.getStatus().getName());

HBox.setHgrow(nameText, Priority.ALWAYS); // Push status text to the right

nameText.setWrappingWidth(1000); // Adjust this width as needed

hBox.getChildren().addAll(nameText, statusText);

hBox.setAlignment(Pos.CENTER\_LEFT); // Align children to the left

setGraphic(hBox);

}

}

});

// Set ListView click handler

ApplicationListView.setOnMouseClicked(this::handleListClick);

}

// Set ListView click handler

private List<Application> loadDataFromDatabase() {

return applicationService.getAll();

}

private void handleListClick(MouseEvent event) {

if (event.getClickCount() == 2) {

Application selected = ApplicationListView.getSelectionModel().getSelectedItem();

if (selected != null) {

try {

sceneManager.openScene(ReportsButton, FxmlView.APPLICATION\_INFO, selected);

} catch (IOException e) {

throw new RuntimeException(e);

}

}

}

}

}<?xml version="1.0" encoding="UTF-8"?>

<?import java.net.URL?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.ListView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="1080.0" prefWidth="1920.0" style="-fx-background-color: #2E3348;" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="project.orgtech.frontController.MainController">

<children>

<AnchorPane layoutY="150.0" prefHeight="840.0" prefWidth="1920.0" style="-fx-background-color: #fafafa;">

<children>

<ListView fx:id="ApplicationListView" layoutX="308.0" layoutY="72.0" prefHeight="717.0" prefWidth="1562.0" style="-fx-padding: 10px; -fx-font-size: 16px; -fx-font-family: 'Comic Sans MS'; -fx-background-color: #fafafa; -fx-border-color: #dddddd;">

<stylesheets>

<URL value="@listview.css" />

</stylesheets>

</ListView>

<Button fx:id="CreateButton" layoutX="47.0" layoutY="726.0" mnemonicParsing="false" onAction="#handleCreateButton" prefHeight="63.0" prefWidth="211.0" style="-fx-background-color: #F39c53;" text="Создать">

<font>

<Font name="Comic Sans MS" size="24.0" />

</font>

</Button>

<Button fx:id="MyApplicationButton" layoutX="47.0" layoutY="112.0" mnemonicParsing="false" onAction="#handleMyApplicationButton" prefHeight="63.0" prefWidth="211.0" style="-fx-background-color: #F39c53;" text="Мои заявки">

<font>

<Font name="Comic Sans MS" size="24.0" />

</font>

</Button>

<Button fx:id="CategoryButton" layoutX="47.0" layoutY="210.0" mnemonicParsing="false" onAction="#handleCategoryButton" prefHeight="63.0" prefWidth="211.0" style="-fx-background-color: #F39c53;" text="Категории">

<font>

<Font name="Comic Sans MS" size="24.0" />

</font>

</Button>

<Button fx:id="ReportsButton" layoutX="47.0" layoutY="306.0" mnemonicParsing="false" onAction="#handleReportsButton" prefHeight="63.0" prefWidth="211.0" style="-fx-background-color: #F39c53;" text="Отчеты">

<font>

<Font name="Comic Sans MS" size="24.0" />

</font>

</Button>

<Label layoutX="326.0" layoutY="23.0" prefHeight="30.0" prefWidth="345.0" text="Заявки" textAlignment="CENTER">

<font>

<Font name="Comic Sans MS" size="24.0" />

</font>

</Label>

</children>

</AnchorPane>

<Label layoutX="810.0" layoutY="41.0" prefHeight="63.0" prefWidth="428.0" text="Ремонт орг техники" textAlignment="CENTER" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="44.0" />

</font>

</Label>

<Label layoutX="838.0" layoutY="1016.0" prefHeight="30.0" prefWidth="400.0" text="Дипломная работа" textAlignment="CENTER" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="24.0" />

</font>

</Label>

<Button fx:id="MasterArea" layoutX="1519.0" layoutY="41.0" mnemonicParsing="false" onAction="#handleMasterArea" prefHeight="63.0" prefWidth="345.0" style="-fx-background-color: #F39c53;" text="Личный кабинет">

<font>

<Font name="Comic Sans MS" size="24.0" />

</font>

</Button>

</children>

</AnchorPane><?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ComboBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="1080.0" prefWidth="1920.0" style="-fx-background-color: #2E3348;" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="project.orgtech.frontController.application.EditApplicationController">

<children>

<AnchorPane layoutY="143.0" prefHeight="850.0" prefWidth="1920.0" style="-fx-background-color: #fafafa;">

<children>

<Label layoutX="562.0" layoutY="45.0" text="Имя" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="nameField" layoutX="841.0" layoutY="45.0" prefHeight="50.0" prefWidth="500.0" promptText="Имя">

<font>

<Font size="24.0" />

</font>

</TextField>

<Label layoutX="562.0" layoutY="125.0" text="Цена" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="priceField" layoutX="841.0" layoutY="125.0" prefHeight="50.0" prefWidth="500.0" promptText="Цена">

<font>

<Font size="24.0" />

</font>

</TextField>

<Label layoutX="562.0" layoutY="205.0" text="Клиент" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<ComboBox fx:id="clientComboBox" layoutX="841.0" layoutY="205.0" prefHeight="50.0" prefWidth="500.0" promptText="Выберите клиента">

</ComboBox>

<Button fx:id="addClientButton" layoutX="1362.0" layoutY="205.0" onAction="#handleAddClient" prefHeight="50.0" prefWidth="150.0" style="-fx-background-color: #F39c53;" text="Добавить">

<font>

<Font size="24.0" />

</font>

</Button>

<Label layoutX="562.0" layoutY="285.0" text="Ремонт" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<ComboBox fx:id="repairComboBox" layoutX="841.0" layoutY="285.0" prefHeight="50.0" prefWidth="500.0" promptText="Выберите ремонт">

</ComboBox>

<Button fx:id="addRepairButton" layoutX="1362.0" layoutY="285.0" onAction="#handleAddRepair" prefHeight="50.0" prefWidth="150.0" style="-fx-background-color: #F39c53;" text="Добавить">

<font>

<Font size="24.0" />

</font>

</Button>

<Label layoutX="562.0" layoutY="365.0" text="Тип" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<ComboBox fx:id="typeComboBox" layoutX="841.0" layoutY="365.0" prefHeight="50.0" prefWidth="500.0" promptText="Выберите тип">

</ComboBox>

<Button fx:id="addTypeButton" layoutX="1362.0" layoutY="365.0" onAction="#handleAddType" prefHeight="50.0" prefWidth="150.0" style="-fx-background-color: #F39c53;" text="Добавить">

<font>

<Font size="24.0" />

</font>

</Button>

<Label layoutX="562.0" layoutY="445.0" text="Статус" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<ComboBox fx:id="statusComboBox" layoutX="841.0" layoutY="445.0" prefHeight="50.0" prefWidth="500.0" promptText="Выберите статус">

</ComboBox>

<Button fx:id="saveButton" layoutX="841.0" layoutY="565.0" onAction="#handleSave" prefHeight="50.0" prefWidth="150.0" style="-fx-background-color: #F39c53;" text="Сохранить">

<font>

<Font size="24.0" />

</font>

</Button>

<Button fx:id="backButton" layoutX="1191.0" layoutY="565.0" onAction="#handleBackButton" prefHeight="50.0" prefWidth="150.0" style="-fx-background-color: #F39c53;" text="Назад">

<font>

<Font size="24.0" />

</font>

</Button>

</children>

</AnchorPane>

<Label layoutX="735.0" layoutY="27.0" prefHeight="84.0" prefWidth="1030.0" text="Создание заявки" textAlignment="CENTER" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="44.0" />

</font>

</Label>

<Label layoutX="824.0" layoutY="1017.0" prefHeight="30.0" prefWidth="400.0" text="Дипломная работа" textAlignment="CENTER" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="24.0" />

</font>

</Label>

</children>

</AnchorPane><?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<?import javafx.scene.control.TextField?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="1080.0" prefWidth="1920.0" style="-fx-background-color: #2E3348;" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="project.orgtech.frontController.application.InfoApplicationController">

<children>

<AnchorPane layoutY="143.0" prefHeight="850.0" prefWidth="1920.0" style="-fx-background-color: #fafafa;">

<children>

<Label layoutX="562.0" layoutY="45.0" text="Имя" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="nameField" editable="false" layoutX="841.0" layoutY="45.0" prefHeight="50.0" prefWidth="500.0" />

<Label layoutX="562.0" layoutY="125.0" text="Цена" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="priceField" editable="false" layoutX="841.0" layoutY="125.0" prefHeight="50.0" prefWidth="500.0" />

<Label layoutX="562.0" layoutY="205.0" text="Клиент" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="clientField" editable="false" layoutX="841.0" layoutY="205.0" prefHeight="50.0" prefWidth="500.0" />

<Label layoutX="562.0" layoutY="285.0" text="Ремонт" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="repairField" editable="false" layoutX="841.0" layoutY="285.0" prefHeight="50.0" prefWidth="500.0" />

<Label layoutX="562.0" layoutY="365.0" text="Тип" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="typeField" editable="false" layoutX="841.0" layoutY="365.0" prefHeight="50.0" prefWidth="500.0" />

<Label layoutX="562.0" layoutY="445.0" text="Статус" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="statusField" editable="false" layoutX="841.0" layoutY="445.0" prefHeight="50.0" prefWidth="500.0" />

<Label layoutX="562.0" layoutY="525.0" text="Дата" textFill="#000000">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="dateField" editable="false" layoutX="841.0" layoutY="525.0" prefHeight="50.0" prefWidth="500.0" />

<Button fx:id="editButton" layoutX="841.0" layoutY="625.0" onAction="#handleEditButton" prefHeight="50.0" prefWidth="200.0" style="-fx-background-color: #F39c53;" text="Редактировать">

<font>

<Font size="24.0" />

</font>

</Button>

<Button fx:id="deleteButton" layoutX="1061.0" layoutY="625.0" onAction="#handleDeleteButton" prefHeight="50.0" prefWidth="200.0" style="-fx-background-color: #F39c53;" text="Удалить">

<font>

<Font size="24.0" />

</font>

</Button>

<Button fx:id="backButton" layoutX="1281.0" layoutY="625.0" onAction="#handleBackButton" prefHeight="50.0" prefWidth="200.0" style="-fx-background-color: #F39c53;" text="Назад">

<font>

<Font size="24.0" />

</font>

</Button>

</children>

</AnchorPane>

<Label layoutX="735.0" layoutY="27.0" prefHeight="84.0" prefWidth="1030.0" text="Просмотр заявки" textAlignment="CENTER" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="44.0" />

</font>

</Label>

<Label layoutX="824.0" layoutY="1017.0" prefHeight="30.0" prefWidth="400.0" text="Дипломная работа" textAlignment="CENTER" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="24.0" />

</font>

</Label>

</children>

</AnchorPane>package project.orgtech.frontController.application;

import javafx.collections.FXCollections;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import project.orgtech.models.\*;

import project.orgtech.service.application.ApplicationService;

import project.orgtech.service.client.ClientService;

import project.orgtech.service.repair.RepairService;

import project.orgtech.service.status.StatusService;

import project.orgtech.service.type.TypeService;

import project.orgtech.utils.AuthManager;

import project.orgtech.utils.FxmlView;

import project.orgtech.utils.SceneManager;

import project.orgtech.frontController.utils.DataReceiver;

import java.io.IOException;

import java.math.BigDecimal;

import java.util.Date;

@Controller

public class EditApplicationController implements DataReceiver<Application> {

@Autowired

private ApplicationService applicationService;

@Autowired

private TypeService typeService;

@Autowired

private ClientService clientService;

@Autowired

private StatusService statusService;

@Autowired

private RepairService repairService;

@Autowired

private SceneManager sceneManager;

@FXML

private TextField nameField;

@FXML

private TextField priceField;

@FXML

private ComboBox<Client> clientComboBox;

@FXML

private ComboBox<Repair> repairComboBox;

@FXML

private ComboBox<Type> typeComboBox;

@FXML

private ComboBox<Status> statusComboBox;

@FXML

private Button saveButton;

@FXML

private Button backButton;

@FXML

private Button addClientButton;

@FXML

private Button addRepairButton;

@FXML

private Button addTypeButton;

private Application currentApplication;

@FXML

public void initialize() {

loadComboBoxData();

}

private void loadComboBoxData() {

clientComboBox.setItems(FXCollections.observableArrayList(clientService.getAll()));

repairComboBox.setItems(FXCollections.observableArrayList(repairService.getAll()));

typeComboBox.setItems(FXCollections.observableArrayList(typeService.getAll()));

statusComboBox.setItems(FXCollections.observableArrayList(statusService.getAll()));

}

@Override

public void setData(Application application) {

this.currentApplication = application;

if (application != null) {

nameField.setText(application.getName());

priceField.setText(application.getPrice().toString());

clientComboBox.setValue(application.getClient());

repairComboBox.setValue(application.getRepair());

typeComboBox.setValue(application.getType());

statusComboBox.setValue(application.getStatus());

}

}

@FXML

private void handleSave(ActionEvent event) throws IOException {

if (validateInput()) {

if (currentApplication == null) {

currentApplication = new Application();

currentApplication.setDate(new Date());

currentApplication.setMaster(AuthManager.getMaster());

}

currentApplication.setName(nameField.getText());

currentApplication.setPrice(new BigDecimal(priceField.getText()));

currentApplication.setClient(clientComboBox.getValue());

currentApplication.setRepair(repairComboBox.getValue());

currentApplication.setType(typeComboBox.getValue());

currentApplication.setStatus(statusComboBox.getValue());

if (currentApplication.getId() == null) {

applicationService.add(currentApplication);

sceneManager.showAlert("Успешно", "Заявка создана");

} else {

applicationService.update(currentApplication);

sceneManager.showAlert("Успешно", "Заявка обновлена");

}

sceneManager.openScene(saveButton, FxmlView.MAIN);

}

}

private boolean validateInput() throws IOException {

StringBuilder errorMessage = new StringBuilder();

if (nameField.getText() == null || nameField.getText().isEmpty()) {

errorMessage.append("Имя не может быть пустым\n");

}

if (priceField.getText() == null || priceField.getText().isEmpty()) {

errorMessage.append("Цена не может быть пустой\n");

} else {

try {

new BigDecimal(priceField.getText());

} catch (NumberFormatException e) {

errorMessage.append("Некорректный формат цены\n");

}

}

if (clientComboBox.getValue() == null) {

errorMessage.append("Выберите клиента\n");

}

if (repairComboBox.getValue() == null) {

errorMessage.append("Выберите ремонт\n");

}

if (typeComboBox.getValue() == null) {

errorMessage.append("Выберите тип\n");

}

if (statusComboBox.getValue() == null) {

errorMessage.append("Выберите статус\n");

}

if (!errorMessage.isEmpty()) {

sceneManager.showAlert("Ошибка", errorMessage.toString());

return false;

}

return true;

}

@FXML

private void handleBackButton(ActionEvent event) throws IOException {

sceneManager.openScene(backButton, FxmlView.MAIN);

}

@FXML

private void handleAddClient(ActionEvent event) throws IOException {

sceneManager.openScene(FxmlView.CLIENT\_APPLICATION\_ADD);

loadComboBoxData();

}

@FXML

private void handleAddRepair(ActionEvent event) throws IOException {

sceneManager.openScene(FxmlView.REPAIR\_APPLICATION\_ADD);

loadComboBoxData();

}

@FXML

private void handleAddType(ActionEvent event) throws IOException {

sceneManager.openScene(FxmlView.TYPE\_APPLICATION\_ADD);

loadComboBoxData();

}

public void handleSaveButton(ActionEvent actionEvent) {

}

}package project.orgtech.frontController.application;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import project.orgtech.models.Application;

import project.orgtech.service.application.ApplicationService;

import project.orgtech.utils.FxmlView;

import project.orgtech.utils.SceneManager;

import project.orgtech.frontController.utils.DataReceiver;

import java.io.IOException;

@Controller

public class InfoApplicationController implements DataReceiver<Application> {

@Autowired

private ApplicationService applicationService;

@Autowired

private SceneManager sceneManager;

@FXML

private TextField nameField;

@FXML

private TextField priceField;

@FXML

private TextField clientField;

@FXML

private TextField repairField;

@FXML

private TextField typeField;

@FXML

private TextField statusField;

@FXML

private TextField dateField;

@FXML

private Button editButton;

@FXML

private Button deleteButton;

@FXML

private Button backButton;

private Application currentApplication;

@Override

public void setData(Application application) {

this.currentApplication = application;

populateFields();

}

private void populateFields() {

if (currentApplication != null) {

nameField.setText(currentApplication.getName());

priceField.setText(currentApplication.getPrice().toString());

clientField.setText(currentApplication.getClient().getBio());

repairField.setText(currentApplication.getRepair().getName());

typeField.setText(currentApplication.getType().getName());

statusField.setText(currentApplication.getStatus().getName());

dateField.setText(currentApplication.getDate().getDay() + " : " + currentApplication.getDate().getMonth() + " : " + currentApplication.getDate().getYear());

}

}

@FXML

private void handleEditButton(ActionEvent event) throws IOException {

sceneManager.openScene(editButton, FxmlView.APPLICATION\_EDIT, currentApplication);

}

@FXML

private void handleDeleteButton(ActionEvent event) throws IOException {

applicationService.delete(currentApplication.getId());

sceneManager.showAlert("Успешно", "Заявка удалена");

sceneManager.openScene(deleteButton, FxmlView.MAIN);

}

@FXML

private void handleBackButton(ActionEvent event) throws IOException {

sceneManager.openScene(backButton, FxmlView.MAIN);

}

} package project.orgtech.frontController.application;

import javafx.collections.FXCollections;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import project.orgtech.models.\*;

import project.orgtech.service.application.ApplicationService;

import project.orgtech.service.client.ClientService;

import project.orgtech.service.repair.RepairService;

import project.orgtech.service.status.StatusService;

import project.orgtech.service.type.TypeService;

import project.orgtech.utils.AuthManager;

import project.orgtech.utils.FxmlView;

import project.orgtech.utils.SceneManager;

import java.io.IOException;

import java.math.BigDecimal;

import java.util.Date;

@Controller

public class AddApplicationController {

@Autowired

private ApplicationService applicationService;

@Autowired

private TypeService typeService;

@Autowired

private ClientService clientService;

@Autowired

private StatusService statusService;

@Autowired

private RepairService repairService;

@Autowired

private SceneManager sceneManager;

@FXML

private TextField nameField;

@FXML

private TextField priceField;

@FXML

private ComboBox<Client> clientComboBox;

@FXML

private ComboBox<Repair> repairComboBox;

@FXML

private ComboBox<Type> typeComboBox;

@FXML

private ComboBox<Status> statusComboBox;

@FXML

private Button saveButton;

@FXML

private Button backButton;

@FXML

private Button addClientButton;

@FXML

private Button addRepairButton;

@FXML

private Button addTypeButton;

@FXML

public void initialize() {

loadComboBoxData();

}

private void loadComboBoxData() {

// Create a Timeline to add a delay before executing the data loading

clientComboBox.setItems(FXCollections.observableArrayList(clientService.getAll()));

repairComboBox.setItems(FXCollections.observableArrayList(repairService.getAll()));

typeComboBox.setItems(FXCollections.observableArrayList(typeService.getAll()));

statusComboBox.setItems(FXCollections.observableArrayList(statusService.getAll()));

}

@FXML

private void handleSave(ActionEvent event) throws IOException {

if (validateInput()) {

Application application = new Application();

application.setName(nameField.getText());

application.setPrice(new BigDecimal(priceField.getText()));

application.setClient(clientComboBox.getValue());

application.setRepair(repairComboBox.getValue());

application.setType(typeComboBox.getValue());

application.setStatus(statusComboBox.getValue());

application.setDate(new Date());

application.setMaster(AuthManager.getMaster());

applicationService.add(application);

sceneManager.showAlert("Успешно", "Заявка создана");

sceneManager.openScene(saveButton, FxmlView.MAIN);

}

}

private boolean validateInput() throws IOException {

StringBuilder errorMessage = new StringBuilder();

if (nameField.getText() == null || nameField.getText().isEmpty()) {

errorMessage.append("Имя не может быть пустым\n");

}

if (priceField.getText() == null || priceField.getText().isEmpty()) {

errorMessage.append("Цена не может быть пустой\n");

} else {

try {

new BigDecimal(priceField.getText());

} catch (NumberFormatException e) {

errorMessage.append("Некорректный формат цены\n");

}

}

if (clientComboBox.getValue() == null) {

errorMessage.append("Выберите клиента\n");

}

if (repairComboBox.getValue() == null) {

errorMessage.append("Выберите ремонт\n");

}

if (typeComboBox.getValue() == null) {

errorMessage.append("Выберите тип\n");

}

if (statusComboBox.getValue() == null) {

errorMessage.append("Выберите статус\n");

}

if (!errorMessage.isEmpty()) {

sceneManager.showAlert("Ошибка", errorMessage.toString());

return false;

}

return true;

}

@FXML

private void handleBackButton(ActionEvent event) throws IOException {

sceneManager.openScene(backButton, FxmlView.MAIN);

}

@FXML

private void handleAddClient(ActionEvent event) throws IOException {

sceneManager.openScene(FxmlView.CLIENT\_APPLICATION\_ADD);

loadComboBoxData();

}

@FXML

private void handleAddRepair(ActionEvent event) throws IOException {

sceneManager.openScene(FxmlView.REPAIR\_APPLICATION\_ADD);

loadComboBoxData();

}

@FXML

private void handleAddType(ActionEvent event) throws IOException {

sceneManager.openScene(FxmlView.TYPE\_APPLICATION\_ADD);

loadComboBoxData();

}

} <?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ComboBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="1080.0" prefWidth="1920.0" style="-fx-background-color: #2E3348;" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="project.orgtech.frontController.application.AddApplicationController">

<children>

<AnchorPane layoutY="143.0" prefHeight="850.0" prefWidth="1920.0" style="-fx-background-color: #fafafa;">

<children>

<Label layoutX="400.0" layoutY="45.0" text="Имя">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="nameField" layoutX="600.0" layoutY="45.0" prefHeight="60.0" prefWidth="800.0" promptText="Имя">

<font>

<Font size="32.0" />

</font>

</TextField>

<Label layoutX="400.0" layoutY="145.0" text="Цена">

<font>

<Font size="32.0" />

</font>

</Label>

<TextField fx:id="priceField" layoutX="600.0" layoutY="145.0" prefHeight="60.0" prefWidth="800.0" promptText="Цена">

<font>

<Font size="32.0" />

</font>

</TextField>

<Label layoutX="400.0" layoutY="245.0" text="Клиент">

<font>

<Font size="32.0" />

</font>

</Label>

<ComboBox fx:id="clientComboBox" layoutX="600.0" layoutY="245.0" prefHeight="60.0" prefWidth="800.0" promptText="Выберите клиента" />

<Button fx:id="addClientButton" layoutX="1450.0" layoutY="245.0" onAction="#handleAddClient" prefHeight="60.0" prefWidth="200.0" style="-fx-background-color: #F39c53;" text="Добавить">

<font>

<Font size="24.0" />

</font>

</Button>

<Label layoutX="400.0" layoutY="345.0" text="Ремонт">

<font>

<Font size="32.0" />

</font>

</Label>

<ComboBox fx:id="repairComboBox" layoutX="600.0" layoutY="345.0" prefHeight="60.0" prefWidth="800.0" promptText="Выберите ремонт" />

<Button fx:id="addRepairButton" layoutX="1450.0" layoutY="345.0" onAction="#handleAddRepair" prefHeight="60.0" prefWidth="200.0" style="-fx-background-color: #F39c53;" text="Добавить">

<font>

<Font size="24.0" />

</font>

</Button>

<Label layoutX="400.0" layoutY="445.0" text="Тип">

<font>

<Font size="32.0" />

</font>

</Label>

<ComboBox fx:id="typeComboBox" layoutX="600.0" layoutY="445.0" prefHeight="60.0" prefWidth="800.0" promptText="Выберите тип" />

<Button fx:id="addTypeButton" layoutX="1450.0" layoutY="445.0" onAction="#handleAddType" prefHeight="60.0" prefWidth="200.0" style="-fx-background-color: #F39c53;" text="Добавить">

<font>

<Font size="24.0" />

</font>

</Button>

<Label layoutX="400.0" layoutY="545.0" text="Статус">

<font>

<Font size="32.0" />

</font>

</Label>

<ComboBox fx:id="statusComboBox" layoutX="600.0" layoutY="545.0" prefHeight="60.0" prefWidth="800.0" promptText="Выберите статус" />

<Button fx:id="saveButton" layoutX="800.0" layoutY="700.0" onAction="#handleSave" prefHeight="80.0" prefWidth="200.0" style="-fx-background-color: #F39c53;" text="Сохранить">

<font>

<Font size="32.0" />

</font>

</Button>

<Button fx:id="backButton" layoutX="1050.0" layoutY="700.0" onAction="#handleBackButton" prefHeight="80.0" prefWidth="200.0" style="-fx-background-color: #F39c53;" text="Назад">

<font>

<Font size="32.0" />

</font>

</Button>

</children>

</AnchorPane>

<Label layoutX="710.0" layoutY="27.0" prefHeight="84.0" prefWidth="500.0" text="Создание заявки" textFill="WHITE">

<font>

<Font name="Comic Sans MS" size="48.0" />

</font>

</Label>

</children>

</AnchorPane>

**Приложение Б**

«Автоматизированная информационная система «Фирма по обслуживанию организационной техники»»

Спецификация

460.ЗМК.00019-01 01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Обозначение** | | |  | | --- | | **Наименование** | | |  | | --- | | **Примечание** | |
|  | |  | | --- | | Документация | |  |
| 460.ЗМК.00019-01 12 01 | «Автоматизированная информационная система «Фирма по обслуживанию организационной техники»» |  |
|  | Текст программы |  |
| 460.ЗМК.00019-01 13 01 | «Автоматизированная информационная система «Фирма по обслуживанию организационной техники»» |  |
|  | Описание программы |  |
| 460.ЗМК.00019-01 34 01 | «Автоматизированная информационная система «Фирма по обслуживанию организационной техники»» |  |
|  | Руководство пользователя |  |
| 460.ЗМК.00019-01 81 01 | «Автоматизированная информационная система «Фирма по обслуживанию организационной техники»» |  |
|  | Пояснительная записка |  |

**Приложение B**

**Диск с исполняемым кодом программного модуля**