Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский экономический университет имени Г.В. Плеханова»
Воронежский филиал

Кафедра Информационных технологии в экономике

Направление Прикладная информатика

Профиль Прикладная информатика в экономике

КУРСОВОЙ ПРОЕКТ

по дисциплине: Разработка автоматизированной системы управления персоналом

тема: Разработка автоматизированной системы учета рабочего времени

Выполнил(а) _____ Иванов В.П.
(подпись) (ФИО)

Студент(ка) гр. ПРз-221

Руководитель _____ Епрынцева Н.А.
(подпись) (ФИО)

Оценка _____
Дата _____

Воронеж 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В условиях быстрого прогресса технологий и распространения смешанных форм занятости, таких как удаленная и частично офисная работа, возрастает потребность в эффективных инструментах для контроля рабочего времени сотрудников.

Управление временем становится ключевым фактором повышения производительности труда, оптимизации процессов и рационального распределения ресурсов. В этой связи особую значимость приобретает разработка приложений для мониторинга рабочего времени и учета переработок, что позволяет не только вести точный учет времени, но и обеспечивать прозрачность в распределении задач и справедливое вознаграждение за переработки.

Такие решения помогают не только работодателям контролировать использование рабочего времени, но и сотрудникам самостоятельно оценивать свою загруженность и эффективность, что в итоге способствует улучшению трудовой дисциплины и общему повышению мотивации.

Актуальность курсового проекта обусловлена необходимостью разработки решений, которые способствуют повышению эффективности управления рабочим временем в современных условиях труда. В условиях роста популярности удаленной работы и гибких графиков организации сталкиваются с проблемами переработок, неравномерного распределения нагрузки и отсутствия прозрачности в учете рабочего времени. Эти факторы могут приводить к снижению производительности и мотивации сотрудников, а также к их профессиональному выгоранию.

Целью курсового проекта является разработка приложения для мониторинга рабочего времени сотрудников, позволяющего вести учет переработок и предоставляющего руководителям инструмент для анализа эффективности работы персонала.

Объектом курсового проекта выступает приложение для мониторинга рабочего времени и учета переработок.

Предметом исследования является процесс разработки и внедрения приложения для учета рабочего времени и переработок сотрудников.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить понятие автоматизированных систем управления персоналом.
2. Провести анализ современных тенденций в управлении персоналом.
3. Проанализировать существующие системы управления персоналом.
4. Изучить требования к системе.
5. Создать модель данных системы учета рабочего времени.
6. Создать интерфейс пользователя системы учета рабочего времени.
7. Выбрать соответствующие технологии и инструменты для организации архитектуры системы.
8. Описать процесс разработки системы.
9. Выбрать язык программирования.
10. Провести тестирование и подготовку системы к внедрению в организацию.
11. Составить руководство пользователя для системы учета рабочего времени.

Таким образом, курсовой проект поможет повысить эффективность управления рабочим временем, предотвратить переработки и выгорание сотрудников, а также обеспечить прозрачность и контроль за трудовыми процессами. Разработанное приложение облегчит мониторинг времени работы, улучшит организацию трудовых ресурсов и повысит мотивацию сотрудников.

1 Обзор литературы

1.1 Понятие автоматизированных систем управления персоналом

Автоматизированные системы управления персоналом (АСУП) представляют собой комплекс программных решений и технологий, предназначенных для автоматизации процессов, связанных с управлением персоналом в организациях.

Такие системы включают в себя инструменты для выполнения задач по подбору, учету, развитию, обучению, оценке и мотивации сотрудников, а также ведению кадрового документооборота.

Основные функции АСУП [2]:

1. Учет кадров — ведение электронной базы данных сотрудников с информацией об их личных данных, опыте работы, навыках и квалификации, а также данных о трудовых договорах и другой кадровой документации.
2. Подбор персонала — автоматизация процесса рекрутинга, включая размещение вакансий, сбор и хранение резюме, проведение первичных оценок кандидатов и планирование собеседований.
3. Оценка эффективности и аттестация — использование инструментов для проведения оценки эффективности сотрудников, мониторинг выполнения поставленных задач, аттестации и разработки планов по повышению квалификации.
4. Учет рабочего времени — контроль рабочего времени сотрудников, учет переработок, отпусков, больничных и других видов отсутствия на рабочем месте.
5. Управление зарплатами и мотивацией — расчет заработной платы, премий, компенсаций и других вознаграждений, а также создание индивидуальных планов мотивации для сотрудников.
6. Планирование карьеры и обучения — разработка карьерных траекторий, планов развития сотрудников, а также организация курсов повышения квалификации и тренингов.

Преимущества автоматизированных систем управления персоналом:

– Эффективность и скорость обработки данных: АСУП автоматизируют многие рутинные задачи, такие как ведение личных дел сотрудников, расчет заработной платы и учет рабочего времени, что значительно сокращает время на выполнение этих процессов.

– Прозрачность и точность данных: Благодаря централизованной базе данных и инструментам контроля, информация о персонале становится более доступной и точной, что позволяет сократить ошибки и улучшить принятие управленческих решений.

– Оптимизация расходов: автоматизация позволяет сократить затраты на ведение кадрового учета и минимизировать вероятность ошибок при расчете заработной платы или ведении документации.

– Улучшение взаимодействия с персоналом: использование АСУП позволяет улучшить коммуникации между HR-отделом и сотрудниками за счет прозрачности процессов, таких как подача заявок на отпуск или предоставление информации о начислениях.

Таким образом, автоматизированные системы управления персоналом представляют собой важнейший инструмент для оптимизации управления человеческими ресурсами. Они интегрируют ключевые HR-процессы, сокращают время на выполнение рутинных задач и минимизируют ошибки, способствуют повышению прозрачности в работе с персоналом и улучшению коммуникаций.

Благодаря внедрению АСУП, компании могут не только повысить эффективность управления кадрами, но и создать более благоприятную рабочую среду, ориентированную на развитие и мотивацию сотрудников.

1.2 Современные тенденции в управлении персоналом

Современные организации вынуждены адаптироваться к постоянно меняющимся условиям труда и новым требованиям сотрудников. В последние

годы управление персоналом (HR) претерпевает серьезные изменения, связанные с внедрением технологий, изменением ожиданий сотрудников и необходимости создания гибких условий труда [7].

Современные тенденции в управлении персоналом включают:

- ✓ Цифровизация и автоматизация: внедрение технологий для автоматизации HR-процессов, таких как подбор кадров, учет рабочего времени, и расчет заработной платы. Это позволяет сократить рутинные задачи, повысить точность и эффективность управления.
- ✓ Гибкость рабочего времени и удаленная работа: развитие гибких графиков и возможностей для удаленной работы, что улучшает баланс между работой и личной жизнью сотрудников, а также привлекает и удерживает таланты.
- ✓ Забота о благополучии сотрудников: акцент на программы, поддерживающие физическое и психическое здоровье сотрудников, включая инициативы по предотвращению выгорания и создание комфортной рабочей среды.
- ✓ Персонализированное обучение и развитие: индивидуальные подходы к обучению и карьерному росту, которые учитывают личные потребности и цели сотрудников, способствуют их профессиональному росту и мотивации.
- ✓ Аналитика данных: использование аналитических инструментов для анализа данных о сотрудниках, что позволяет принимать более обоснованные решения, прогнозировать потребности и оптимизировать HR-процессы.
- ✓ Разнообразие и инклюзивность: создание инклюзивной рабочей среды, способствующей культурному и гендерному разнообразию, что улучшает корпоративную культуру и привлекает разнообразные таланты.
- ✓ Устойчивое развитие и корпоративная социальная ответственность: внедрение практик устойчивого развития и социальной

ответственности, что способствует улучшению имиджа компании и укрепляет ее связь с общественностью.

Современные тенденции в управлении персоналом отражают стремление к большей гибкости, персонализации и заботе о благополучии сотрудников. Внедрение новых технологий и аналитических инструментов позволяет HR-специалистам адаптировать процессы к изменяющимся условиям работы, поддерживая продуктивность и вовлеченность сотрудников.

Успешные компании уделяют внимание не только технологическим изменениям, но и развитию инклюзивной, разнообразной и ценностно-ориентированной корпоративной культуры [12].

1.3 Анализ существующих систем управления персоналом

Существующие системы управления персоналом могут значительно различаться по функциональности, сложности и масштабу применения [11].

Анализ существующих систем управления персоналом позволяет:

- Оценить функциональные возможности: определить, какие функции предлагает каждая система, чтобы выбрать наиболее подходящую для решения конкретных задач.
- Сравнить преимущества и недостатки: понять сильные и слабые стороны систем для выбора наиболее эффективного и безопасного решения.
- Планировать бюджет: оценить стоимость внедрения и обслуживания систем, чтобы выбрать оптимальное решение в рамках доступного бюджета.
- Упростить внедрение: определить степень сложности настройки и потребность в дополнительной поддержке, что помогает минимизировать риски и затраты на внедрение.
- Прогнозировать будущие потребности: выбрать систему, которая сможет адаптироваться к росту и изменениям в вашей организации, что предотвратит необходимость частой замены решений.

Анализ обеспечивает понимание доступных вариантов и помогает сделать обоснованный выбор, который оптимизирует процессы управления.

Анализ таких систем представлен в виде таблицы 1:

Таблица 1 - Существующие системы управления персоналом

Система	Функционал	Преимущества	Недостатки
SAP SuccessFactors	Управление кадрами, подбор, оценка производительности, обучение, расчёт зарплаты, аналитика	Широкий функционал, интеграция с другими системами, глобальная доступность	Высокая стоимость, сложность внедрения и настройки
ADP Workforce Now	Управление кадрами, подбор, оценка производительности, расчёт зарплаты, учёт рабочего времени	Широкий функционал, гибкость, облачные технологии	Высокая стоимость, сложность настройки, потребность в поддержке
BambooHR	Управление кадрами, подбор, оценка производительности, управление отпуском, корпоративная культура	Простота использования, доступная стоимость, хорошая поддержка	Ограниченный функционал, меньшая интеграция
Workday	Управление кадрами, подбор, оценка производительности, обучение, расчёт зарплаты, финансовый учёт	Интуитивно понятный интерфейс, облачные технологии, сильная аналитика	Высокие затраты, сложности с адаптацией
Namely	Управление кадрами, подбор, оценка производительности, управление зарплатой и льготами, корпоративная культура	Интуитивно понятный интерфейс, поддержка корпоративной культуры, гибкость	Ограниченные функции для крупных компаний, ограниченная интеграция

Анализ СУП демонстрирует, что у каждой из них есть свои сильные и слабые стороны. Оптимальный выбор системы определяется специфическими потребностями и задачами организации.

2 Анализ требований к системе

Анализ требований к системе — это процесс выявления и описания функциональных и нефункциональных характеристик, которые система должна обеспечить для эффективного выполнения своих задач [9].

Анализ требований к системе включает в себя следующие ключевые элементы:

1. Функциональные требования: описание задач, которые система должна выполнять, такие как управление временем, учет переработок и создание отчетов.
2. Нефункциональные требования: определение характеристик системы, таких как надежность, производительность, безопасность и масштабируемость.
3. Интерфейсные требования: требования к дизайну и взаимодействию пользователя с системой, включая удобство использования и доступность.
4. Технические ограничения: учет ограничений, связанных с бюджетом, сроками, оборудованием и совместимостью с существующими системами.
5. Интеграция: оценка необходимости взаимодействия с другими системами и базами данных.
6. Юридические и нормативные требования: соответствие законодательным нормам.

На основе рассмотренных этапов определим требования к приложению для мониторинга рабочего времени и учета переработок:

1. Функциональные требования:

- Учет рабочего времени: разработать модуль для регистрации рабочего времени сотрудников.
- Учет переработок: разработать модуль для автоматического вычисления и учета часов переработки.

- Создание отчетов: разработать модуль для генерации сводки о затраченном времени и переработках.

2. Нефункциональные требования:

- Надежность: стабильная работа приложения без сбоев.
- Производительность: разработать модуль для обработки данных и генерация отчетов.

3. Интерфейсные требования:

- Удобство использования: разработать модуль с понятным интерфейсом и простотой в навигации.
- Доступность: разработать систему с авторизацией для каждого пользователя.

4. Безопасность:

- Безопасность информации: разработать систему защиты данных пользователей и предотвращение несанкционированного доступа.

5. Технические ограничения:

- Стоимость: разработать приложение в рамках установленного бюджета.
- Оборудование: разработать приложение, которое совместимо с имеющимся оборудованием и программным обеспечением.

6. Юридические и нормативные требования:

- Соответствие законодательству: разработать приложение в соответствии требованиям законодательства о защите персональных данных и трудового права.

3 Проектирование системы

3.1 Архитектура системы

Архитектура системы — это структурное описание системы, определяющее, как различные компоненты взаимодействуют между собой для выполнения задач. Она описывает ключевые элементы системы, их взаимосвязи, а также принципы работы и распределение функций [4].

Архитектура системы включает:

1. Структура и модули: разделение на структурные модули.
2. Взаимодействие с компонентами: как модули обмениваются данными и взаимодействуют друг с другом.
3. Технологический стек: набор технологий и инструментов (серверная часть, базы данных, интерфейс пользователя).
4. Уровни системы: клиентская часть (интерфейс), серверная часть (логика), база данных (хранение данных).
5. Интеграция: подключение к внешним системам через API.

Архитектура приложения для мониторинга рабочего времени и учета переработок должна обладать гибкостью, удобством и возможностью масштабирования, одновременно гарантируя надежность и безопасность данных.

Архитектура для разрабатываемого приложения:

1. Удобный интерфейс: интуитивно понятный и простой в навигации для повышения эффективности пользователей.
2. Сервисы: сервисы приложения должны включать функционал мониторинга и учета переработок, а также создание отчетности. Сервисы включают в себя:

- Учет рабочего времени: отметка и контроль сотрудника на рабочем месте.
- Учет переработок: ведение и редактирование списков о подработках сотрудников.

- Отчетность: формирование отчетов об отработанных часах и допустимых переработок.

3. База данных: база данных будет хранить информацию о сотрудниках и времени пребывания их на рабочем месте.

4. Безопасность: защита данных с помощью авторизации и шифрования.

3.2 Модели данных

Модель данных — это абстрактное представление структуры данных, которое описывает, как данные организованы, связаны и взаимодействуют друг с другом в системе. Она служит основой для разработки базы данных и включает [5]:

1. Сущности: основные объекты, о которых ведется учет.
2. Атрибуты: характеристики или свойства сущностей.
3. Связи: отношения между сущностями.
4. Типы данных: определение формата и типа данных для каждого атрибута.
5. Ограничения: правила, которые должны соблюдаться для поддержания целостности данных.

Для приложения для мониторинга рабочего времени и учета переработок определены следующие сущности и их атрибуты:

1. Сущность данных Сотрудники: ID сотрудника, имя, пароль.
2. Сущность данных Рабочие дни: ID рабочего дня, ID сотрудника, дата рабочего дня.
3. Сущность данных Отметки: ID отметки, ID сотрудника, дата отметки.
4. Сущность данных Переработки: ID переработки, ID сотрудника, дата переработки.

Таким образом, эта модель позволит эффективно управлять данными о

сотрудниках, их рабочих днях, отметках и переработках, обеспечивая целостность и связь между данными в системе.

Графическое представление сущностей и атрибутов изображено на рисунке 1:

Рисунок 1 – Модель данных

На основе описанных сущностей и их атрибутов будет разработана база данных для приложения, предназначенного для мониторинга рабочего времени и учета переработок.

3.3 Интерфейс пользователя

Интерфейс пользователя — это совокупность элементов и средств взаимодействия между пользователем и системой. Он включает в себя все визуальные компоненты, такие как кнопки, меню, окна, формы и графики, которые помогают пользователю выполнять задачи и получать информацию от приложения [8].

Основная цель интерфейса — обеспечить удобство и интуитивность использования системы, минимизирует время и усилия, затрачиваемые на выполнение операций [6].

Составляющие интерфейса пользователя приложения, предназначенного для мониторинга рабочего времени и учета переработок:

- Кнопки: Элементы, позволяющие пользователю выполнять

действия, такие как «Отметиться» или «Выход».

- Текстовые поля: Места для ввода данных, например, логина и пароля.
- Меню: Структуры, позволяющие пользователю выбирать различные функции приложения, такие как управление графиками или просмотр сводок.
- Иконки: Графические символы, помогающие визуально ориентироваться в функционале.
- Сообщения об ошибках: Уведомления, информирующие пользователя о неправильных действиях или вводах.

Эти элементы должны быть интуитивно понятными и легкими.

Далее для наглядного представления, можно составить функциональную схему.

Схема служит наглядным инструментом, который помогает структурировать взаимодействие между элементами интерфейса и функциональностью приложения.

Функциональная схема приложения, предназначенного для мониторинга рабочего времени и учета переработок изображена на рисунке 2:

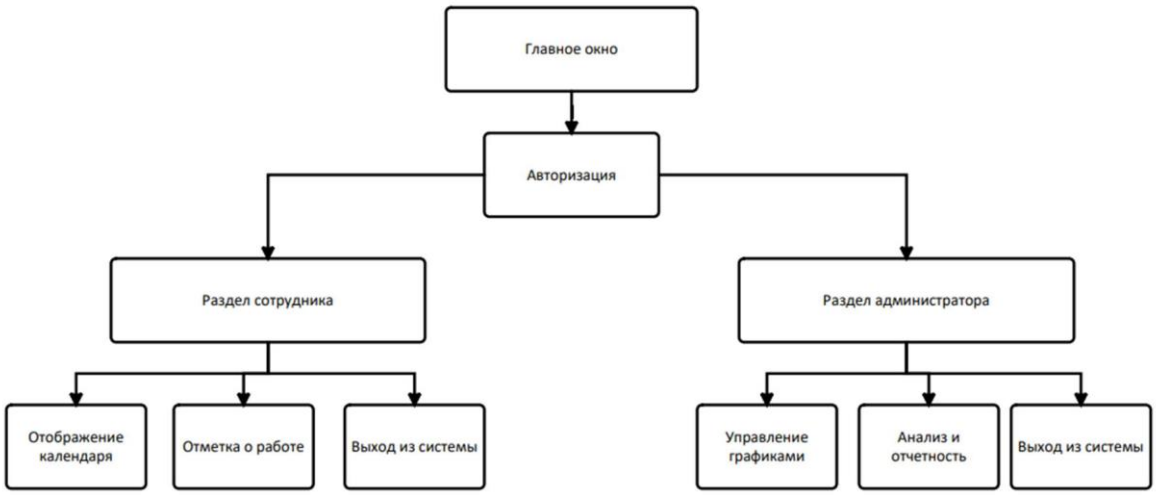

Рисунок 2 – Функциональная схема

Таким образом, можно отследить связь между элементами интерфейса и функциональными возможностями приложения.

4 Разработка системы

4.1 Выбор инструментов и технологий

Выбор инструментов для разработки — это ключевой элемент успеха проекта, так как они напрямую влияют на качество, скорость разработки, удобство поддержки и масштабируемость.

Для создания приложения была выбрана среда разработки IntelliJ IDEA на языке программирования Java.

IntelliJ IDEA - это мощная интегрированная среда разработки (IDE), которая обладает рядом преимуществ [3]:

1. Интеллектуальное автодополнение - IntelliJ IDEA использует сложные алгоритмы для предсказания возможных вариантов автодополнения кода, что позволяет писать код быстрее и с меньшим количеством ошибок.
2. Поддержка Java - IDEA полностью оптимизирована для Java и предлагает широкий спектр инструментов, таких как рефакторинг, статический анализ, интеграция с системами сборки.
3. Поддержка GUI-разработки - IDEA содержит встроенные инструменты для разработки интерфейсов на Java Swing, что удобно для данного приложения, так как используется библиотека Swing для создания пользовательского интерфейса.
4. Отличный отладчик - IDEA имеет мощные возможности для отладки, что значительно облегчает процесс выявления и исправления ошибок в коде.

Несмотря на ряд преимуществ, среда разработки имеет следующие недостатки [13]:

1. Потребление ресурсов - IntelliJ IDEA довольно требовательна к ресурсам компьютера, особенно при работе с большими проектами.
2. Коммерческая лицензия - Профессиональная версия IntelliJ IDEA платная, что может быть ограничением для некоторых разработчиков. Также существует бесплатная версия Community, она обладает меньшим

функционалом.

В случае с разработкой приложения для мониторинга рабочего времени и учета переработок бесплатная версия отлично справится с поставленными задачами.

4.2 Описание процесса разработки

Целью разработки приложения является создание системы учёта рабочего времени, которая позволит администраторам и сотрудникам взаимодействовать через графический интерфейс, отслеживать рабочие дни, проверять время прихода и управлять расписанием.

Ключевые компоненты приложения:

- ✓ Аутентификация пользователей: Доступ для администратора и сотрудников с помощью логина и пароля.
- ✓ Графический интерфейс: Использование Swing для создания окон, кнопок и панелей для удобного взаимодействия с пользователем.
- ✓ Хранение данных: Сериализация объектов для сохранения данных о рабочем времени, днях прихода и переработках в файл.

Этапы разработки приложения:

1. Инициализация данных: При создании экземпляра класса EmployeeTimeTrackingSystem инициализируются учетные данные администраторов и сотрудников, а также загружаются сохраненные данные, если они есть.
2. Создание основного окна: Реализуется главное окно с кнопками для входа в систему для администраторов и сотрудников.
3. Аутентификация: При нажатии на кнопку входа отображается окно для ввода логина и пароля. После проверки аутентификации, пользователю предоставляется доступ к соответствующим функциям.

4. Управление расписанием: Администратор может управлять расписанием сотрудников, добавляя или удаляя рабочие дни. Сотрудники могут отмечаться на работу и проверять свои рабочие дни.

5. Сводка: Возможность для администратора просматривать сводку о рабочем времени сотрудников, включая количество рабочих дней, дней переработки и пропущенных дней.

6. Сохранение и загрузка данных: Реализованы методы для сериализации и десериализации данных о рабочем времени, что позволяет сохранять и загружать информацию между запусками приложения.

Компоненты, которые используются в среде разработки представлены в таблице 2:

Таблица 2 – Используемые компоненты

Название	Описание
JFrame	Представляет окно приложения
JLabel	Отображает текстовые метки
JButton	Кнопки для выполнения действий
JTextField, JPasswordField	Поля для ввода логина и пароля
JTextArea	Текстовая область для отображения сводки
JCheckBox	Выбор рабочих дней сотрудника
JPanel	Контейнер для организации других компонентов, таких как кнопки и метки
JOptionPane	Диалоговые окна для отображения информации или ошибок

Таким образом, основная функциональность приложения сосредоточена на предоставлении удобного графического интерфейса для управления рабочим графиком сотрудников и их отметками.

4.3 Реализация основных модулей системы

Процесс разработки приложения для учёта рабочего времени сотрудников можно разбить на несколько основных шагов. Полный листинг программы будет отображен в Приложении 1.

Для начала создается новый проект Java и добавляются необходимые библиотеки, включая Swing для графического интерфейса.

Далее создается главное окно приложения (рисунок 3), которое будет содержать кнопки для входа администратора и сотрудников.

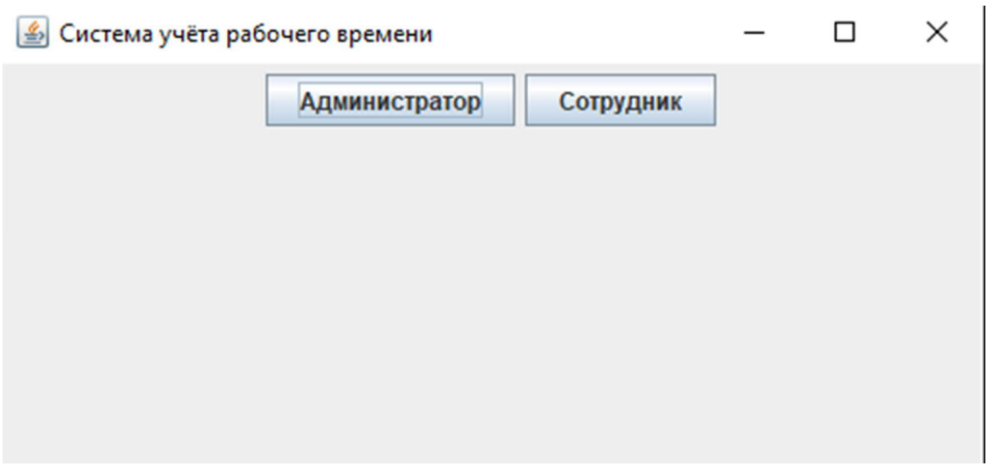

Рисунок 3 – Окно выбора режима работы

После выбора нужного режима высвечивается окно для аутентификации пользователей, которое изображено на рисунке 4:

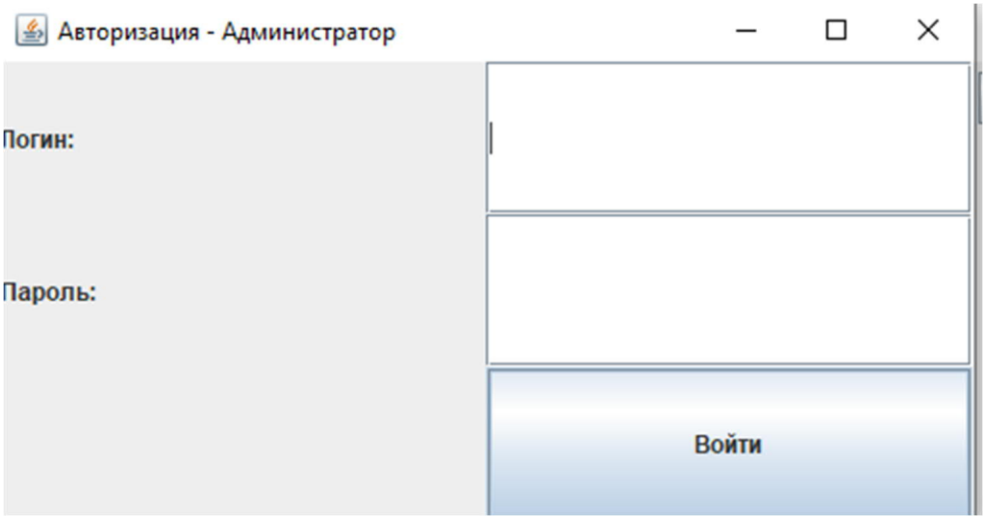

Рисунок 4 – Окно авторизации администратора

После успешной аутентификации администратор получает доступ к своему окну (рисунок 5), где он может управлять расписанием сотрудников.

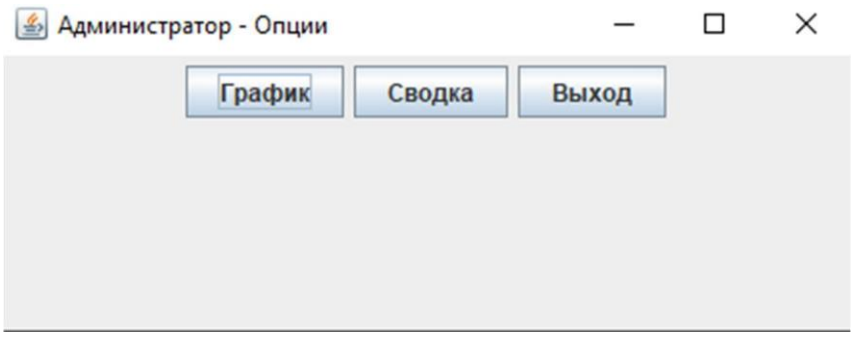

Рисунок – 5 Окно управления администратора

Администратору в разделе График (рисунок 6), предоставляется выбор конкретного сотрудника.

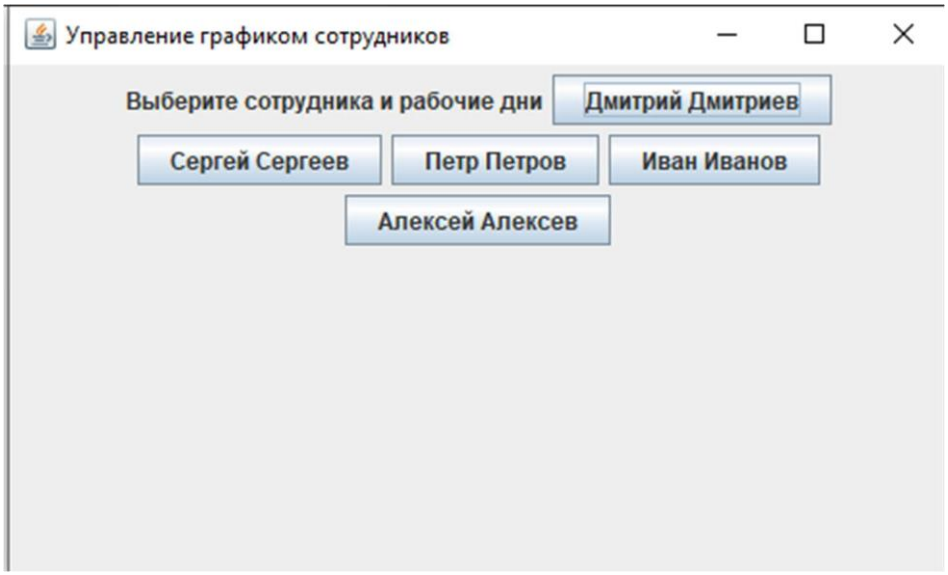

Рисунок 6 – Управление графиком сотрудников

При выборе конкретного сотрудника, Администратора перемещает в календарь (рисунок 7), где можно отметить рабочий день.

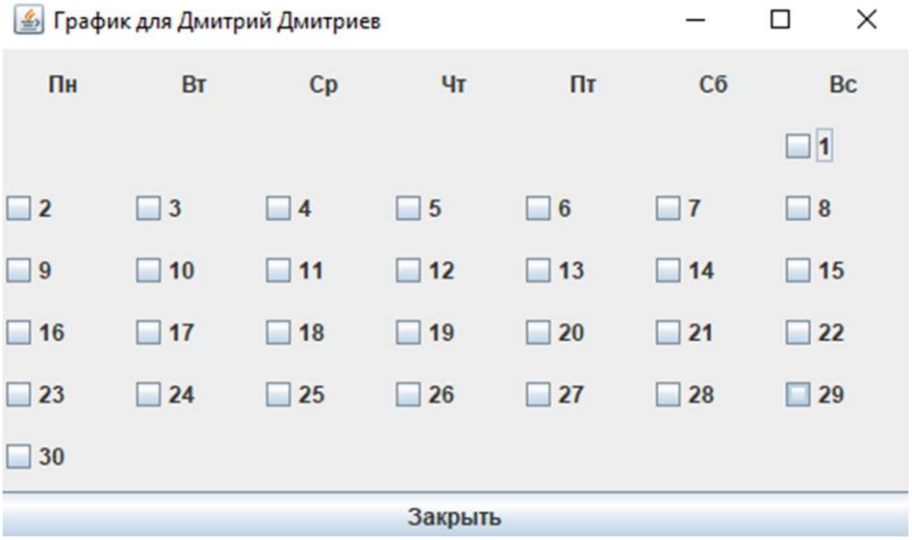

Рисунок 7 – График сотрудника Дмитрия Д.

После выбора нужных дней, в систему добавляются данные о рабочих и переработанных днях для каждого сотрудника.

Также администратор имеет доступ к формированию Сводки (рисунок 8) на основании имеющихся данных, где отображается деятельность сотрудников.

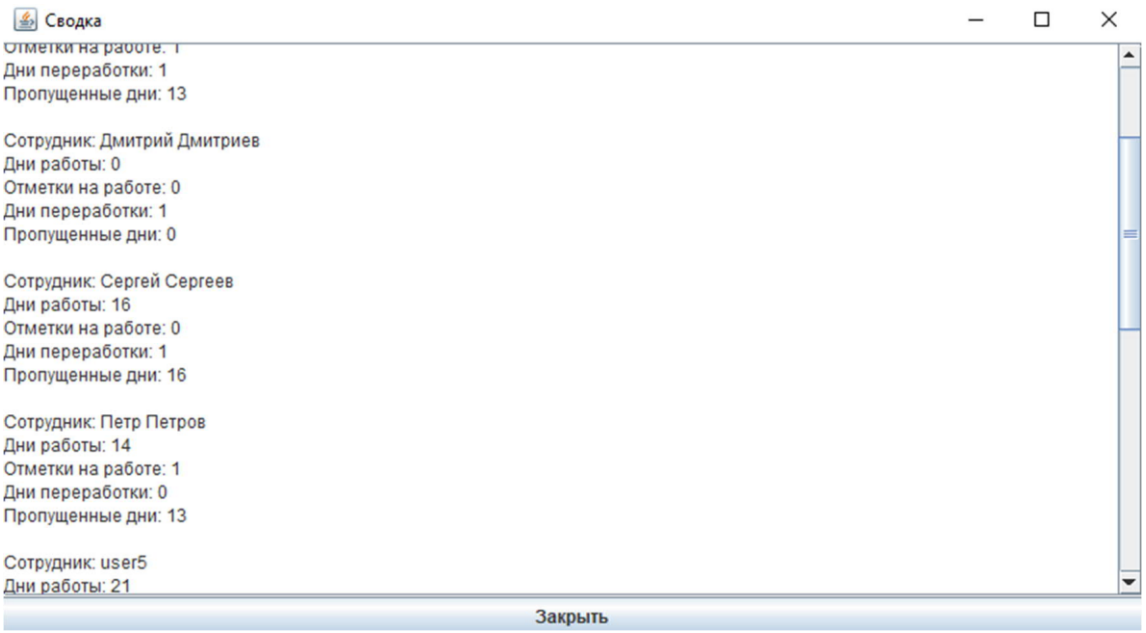

Рисунок – 8 Сводка сотрудников

Таким образом, разработка приложения для мониторинга рабочего времени и учета переработок завершена.

5 Тестирование и введение системы

5.1 Методология тестирования

Тестирование разработанного приложения для учёта рабочего времени сотрудников — это важный этап, который помогает гарантировать его качество и функциональность [1]. Шаги, которые будут использоваться в методологии тестирования:

1. Проверка интерфейса:

✓ Тестирование дизайна: Протестировать, что всё будет отображаться правильно и красиво.

✓ Тестирование навигации: Протестировать, будут ли кнопки и меню работать, и пользователю будет легко перемещаться по приложению.

2. Тестирование авторизации:

✓ Тестирование входа в систему: Протестировать, смогут ли пользователи войти с правильным логином и паролем.

✓ Тестирование обработки ошибок: Протестировать, что при неправильном вводе будет выдаваться правильное сообщение об ошибке.

3. Тестирование функциональности:

✓ Тестирование основных функций: Протестировать, что все важные функции приложения будут работать

4. Тестирование производительности:

✓ Тестирование нагрузки: Протестировать, как приложение будет справляться с большим количеством пользователей одновременно.

✓ Тестирование скорости работы: Протестировать, насколько быстро приложение будет отвечать на действия пользователя.

5. Тестирование безопасности:

- Тестирование на уязвимости: Протестировать, слабые места в приложении, чтобы защитить данные пользователей.
- Тестирование шифрования: Протестировать, что пароли и важная информация будут защищены.

Таким образом, по данной методологии проведет тестирование разработанного приложения для учёта рабочего времени сотрудников.

5.2 Проведение тестирования

Проведем тестирование разработанного приложения. После запуска приложения необходим выбор режима работы (рисунок 3).

Далее окно ввода логина и пароля и если данные введены неправильно, вылезает сообщение с ошибкой, которое изображено на рисунке 9:

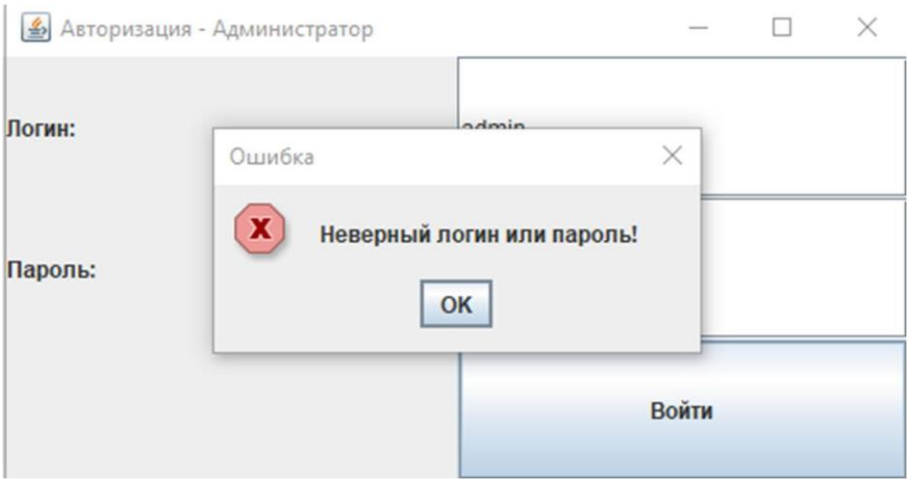

Рисунок 9 – Сообщение об ошибке входа

Если данные введены правильно, то в окне график (рисунок 7), необходимо выставить рабочие дни для сотрудника, как показано на рисунке 10:

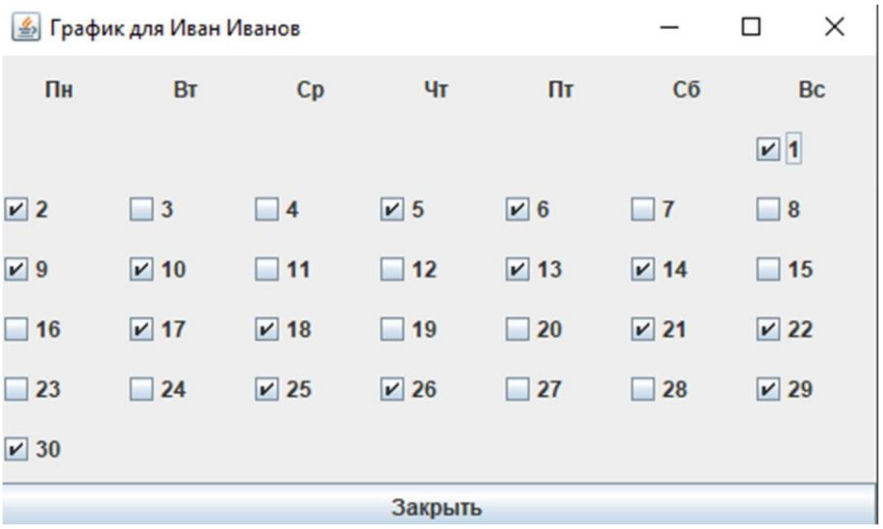

Рисунок 10 – Выставление рабочих дней

После того, как сотрудник отметится в приложении, администратор может сформировать сводку, в которой будет указаны следующие данные:

- Сотрудник.
- Дни работы.
- Отметки на работе.
- Дни переработки.
- Пропущенные дни.

Для сотрудника также создан свой раздел, где после успешной авторизации, высвечивается окно, где необходимо отметиться в системе, что присутствовал на рабочем месте, как отображено на рисунке 11:

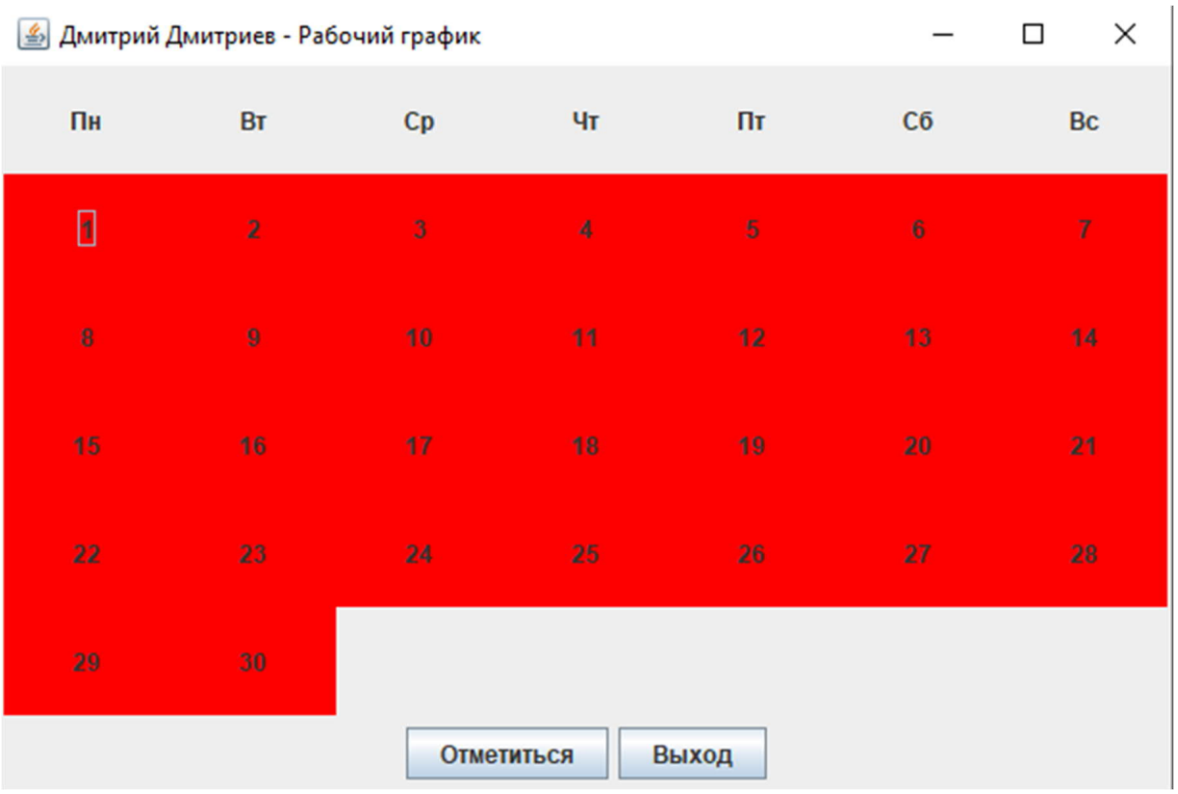

Рисунок 11 – Окно рабочего графика сотрудника

Здесь видно, что сотрудник еще не отмечался на работе и эти дни выделены красным цветом.

После того, как сотрудник выберет день и нажмет кнопку отметиться, он буден отмечен зеленым цветом (рисунок 12), что визуально хорошо отображает процесс представления информации в разрабатываемом приложении.

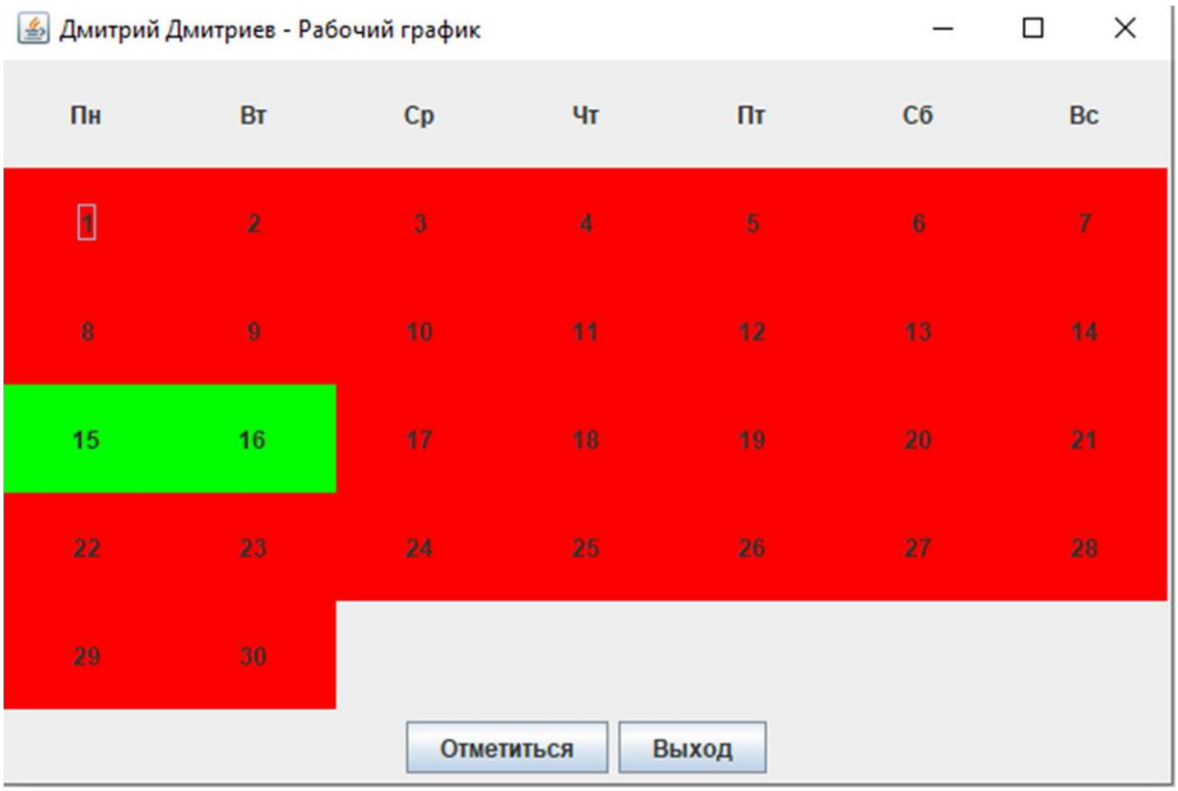

Рисунок 12 – Успешная отметка сотрудника

В ходе тестирования приложения, было выявлено, что оно работает без ошибок, правильно выводит информацию и выводит предупреждения об ошибках, если данные введены не верно.

5.3 Подготовка к внедрению

Внедрение приложения — это стратегический процесс, который требует внимательного подхода и вовлеченности всех участников. Успех зависит от ясных целей, понимания потребностей пользователей и готовности адаптироваться к изменениям.

Подготовка к внедрению приложения для мониторинга рабочего времени и учета переработок состоит из следующих этапов:

1. Анализ потребностей:

- Определение целей проекта: Уточните цели, связанные с внедрением приложения, включая требования к функциональности, интеграцию с текущими системами и обучение пользователей.
- Оценка ресурсов: Проанализируйте доступные ресурсы, включая

время, бюджет и персонал, необходимый для успешного внедрения.

2. Создание технической инфраструктуры:

- Настройте серверы для обработки и хранения данных приложения, обеспечьте безопасный доступ к нему через правильно сконфигурированные сети.

3. Разработка и настройка приложения

- Установка приложения: Установите программное обеспечение на серверы и выполните его начальную настройку в соответствии с потребностями компании.
- Проведение тестирования: Выполните тестирование приложения для проверки его производительности и функциональности.

4. Обучение и подготовка пользователей:

- Тренинг для сотрудников: Проведите обучение для пользователей, чтобы они могли уверенно использовать приложение.
- Создание вспомогательных материалов: Подготовьте руководства и инструкции для пользователей, облегчающие работу с приложением.
- Обеспечение службы поддержки: Установите систему поддержки для помощи пользователям при возникновении вопросов или проблем.

5. Мониторинг и оценка работы приложения:

- Настройте средства, позволяющие следить за работой приложения и выявлять проблемы.
- Разработайте график обновлений приложения.

6. Запуск пилотного проекта:

- Тестирование в реальных условиях: Запустите приложение в ограниченном объеме, чтобы протестировать его работу на практике.
- Сбор обратной связи: Соберите отзывы от пользователей, чтобы выявить недочеты и области для улучшения.

7. Полномасштабный запуск:

- Официальный запуск приложения: Реализуйте полное

развертывание приложения для всех сотрудников.

8. Мониторинг приложения:

- Непрерывный мониторинг и поддержка: Обеспечьте постоянный контроль за работой приложения и оперативную поддержку для пользователей.

В результате выполнения всех этапов внедрения приложение становится более эффективным, безопасным и удобным для пользователей, что, в свою очередь, улучшает процессы учета рабочего времени и переработок в компании.

5.4 Руководство пользователя

Руководство пользователя — это документ, который предоставляет инструкции и информацию о том, как использовать приложение или систему. Оно может включать в себя описание функций, пошаговые процедуры, советы по устранению неполадок и другую важную информацию.

Создадим руководство пользователя для приложения для мониторинга рабочего времени и учета переработок.

При открытии приложения появляется форма авторизации, где необходимо выбрать режим и ввести логин и пароль, как показано на рисунке 13:

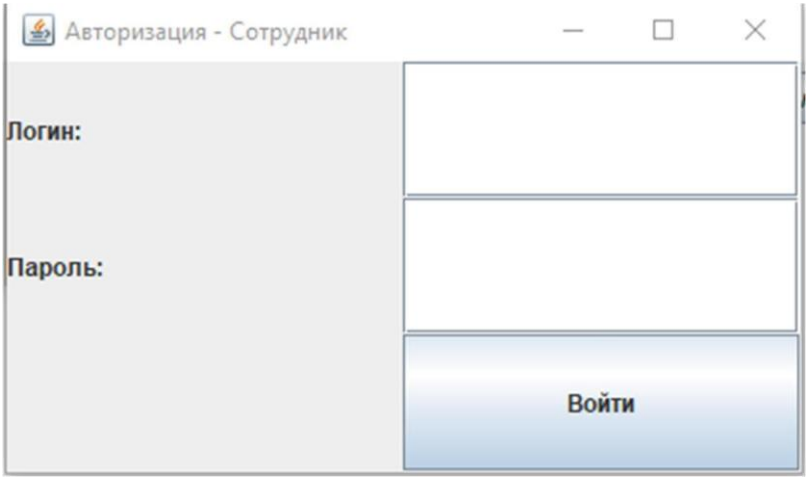

Рисунок 13 – Авторизация сотрудника

После успешного ввода данных, сотруднику будет доступно следующее окно, где ему необходимо отметиться в системе и если прошло успешно, то будет отображено соответствующее сообщение, как показано на рисунке 14:

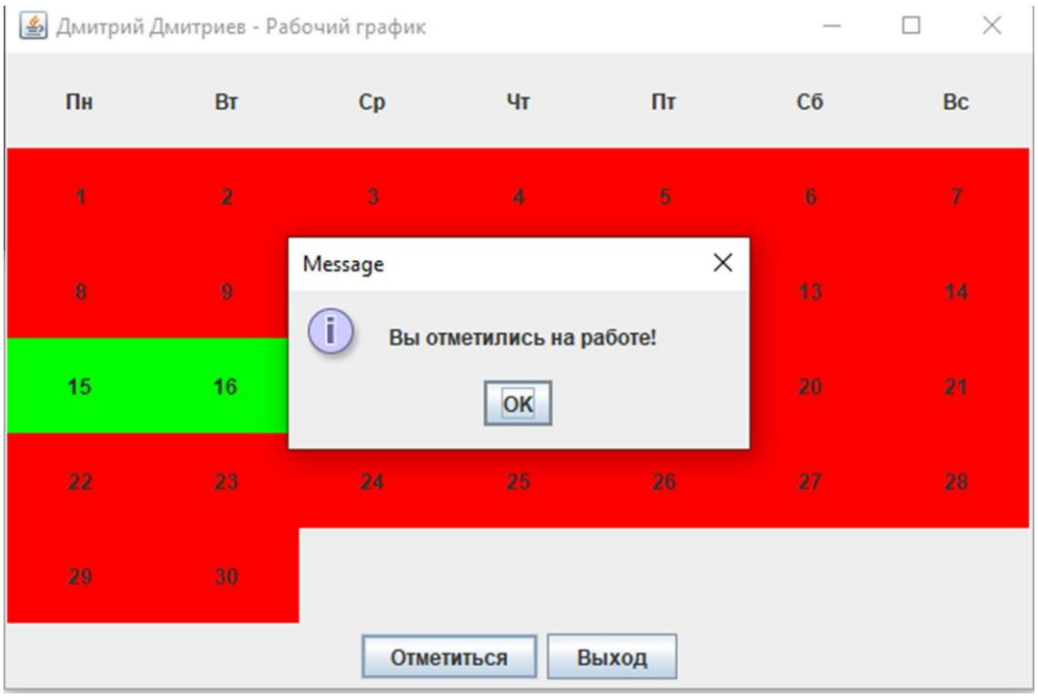

Рисунок 14 – Успешная отметка сотрудника

В приложении встроен календарь, и оно анализирует, какой текущий день недели и когда администратор выставил график, и сотрудник вышел на дополнительную схему, то отображается сообщение о том, что день будет отмечен, как переработка (рисунок 15).

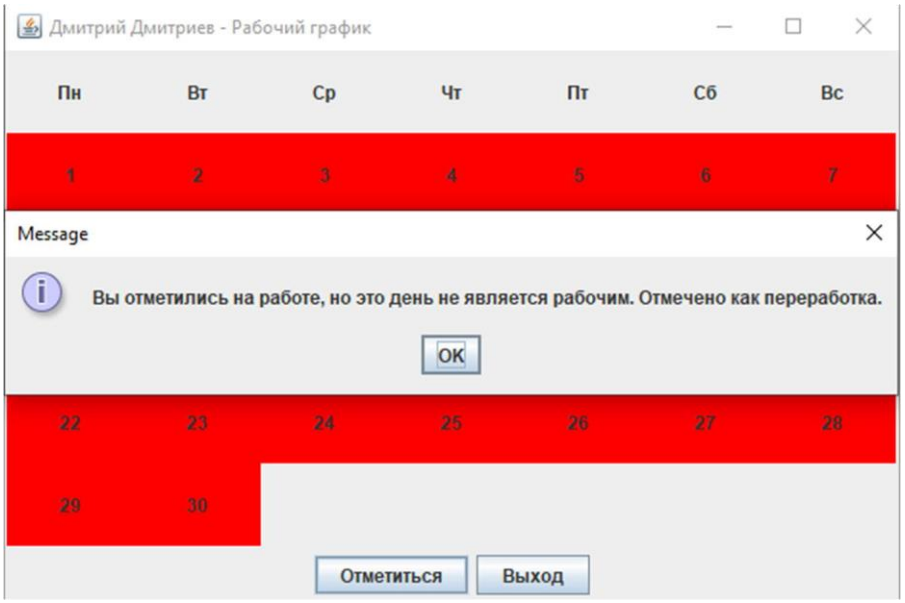

Рисунок 15 – Отметка о переработке

В целом, как выглядит окно сотрудника, когда администратор настроил график, изображено на рисунке 16:

Рисунок 16 – Рабочее окно сотрудника

Приложение систематизирует и анализирует отметки сотрудника и далее администратор формирует сводку, в которой будет отражаться сколько сотрудник дней работал в своем графике, а сколько дней было переработано (рисунок 17).

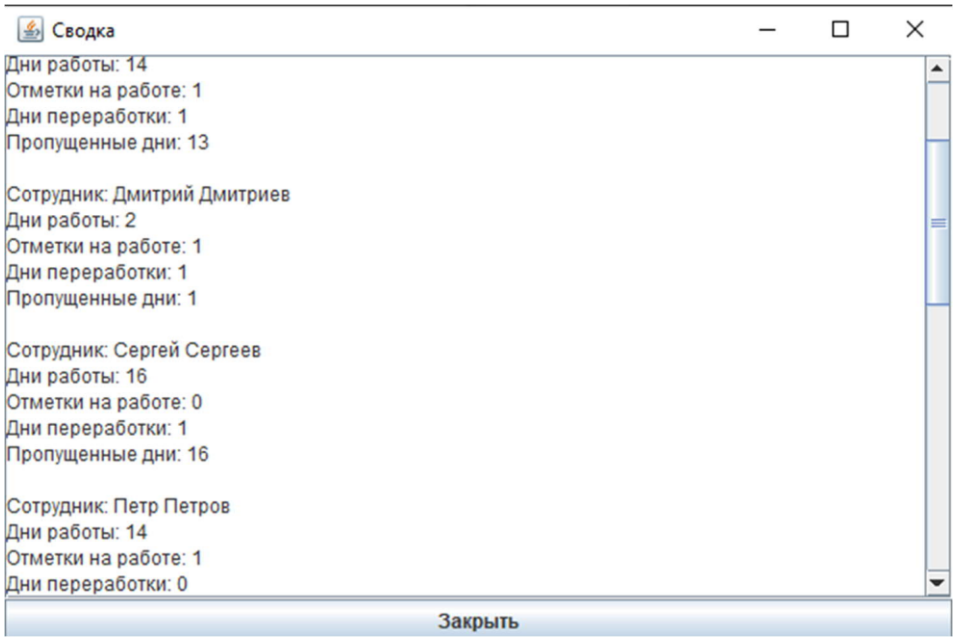

Рисунок 17 – Окончательная сводка

Исходя из введенных данных, наглядно видно, что приложение посчитало рабочие дни и переработки сотрудника Дмитрия Д.

Разработанное приложение имеет интуитивно понятный интерфейс, который позволяет пользователям быстро находить необходимые функции без необходимости глубокого обучения. Все кнопки и меню расположены логично, что облегчает навигацию.

ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта была разработана система учета рабочего времени, которая отвечает современным требованиям автоматизации управления персоналом. Основной целью проекта стало создание эффективного инструмента, способствующего упрощению процессов учета рабочего времени и переработок.

На первом этапе была проведена глубокая работа над изучением понятий автоматизированных систем управления персоналом, что позволило определить ключевые особенности и преимущества таких систем. Анализ современных тенденций в управлении персоналом выявил необходимость интеграции современных технологий в процессы учета рабочего времени, что обосновало актуальность разработки данной системы.

Далее был выполнен анализ существующих систем управления персоналом, что позволило выявить их сильные и слабые стороны. Исходя из этого, были сформулированы требования к новой системе, что стало основой для разработки модели данных. Созданная модель данных системы учета рабочего времени обеспечивает высокую степень структурированности и легкость в обработке информации.

Интерфейс пользователя был спроектирован с акцентом на удобство и интуитивность, что способствует быстрой адаптации сотрудников к новому инструменту. Выбор технологий и инструментов для архитектуры системы был основан на анализе требований, что гарантирует надежность и производительность приложения.

В качестве среды разработки была выбрана IntelliJ IDEA, что обусловлено её широкими возможностями, поддержкой множества инструментов и библиотек для разработки на языке Java. Эта среда предоставляет мощные средства для отладки, тестирования и оптимизации кода, что существенно упрощает процесс разработки и повышает его качество. Интуитивно понятный интерфейс и наличие множества плагинов делают работу разработчика более комфортной и продуктивной.

Процесс разработки системы был описан в деталях, начиная с выбора языка программирования — Java, и заканчивая этапами тестирования и подготовки к внедрению. Это обеспечило структурированный подход к разработке, что минимизировало возможные ошибки и повысило качество конечного продукта.

Кроме того, составленное руководство пользователя предоставляет полное описание функционала системы, что облегчает ее внедрение и использование в организации. Тестирование системы подтвердило ее работоспособность и соответствие заявленным требованиям.

Таким образом, курсовой проект показал важность автоматизации процессов учета рабочего времени и управления персоналом в организации. Реализация данной системы не только повысит эффективность учета рабочего времени, но и создаст более прозрачную и организованную рабочую среду, что в конечном итоге способствует улучшению общего результата работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аниче М. Эффективное тестирование программного обеспечения / пер. с англ. А. Н. Киселева / М. Аниче. - Москва : ДМК Пресс, 2023. - 370 с. - ISBN 978-5-97060-997-2. - URL: https://ibooks.ru/bookshelf/389561/reading

2. Беликова И.П. Управление персоналом / И.П. Беликова. - Ставрополь : Ставропольский государственный аграрный университет (АГРУС), 2021. - 68 с. - ISBN StGAU135. - URL: https://ibooks.ru/bookshelf/387998/reading

3. Вязовик Н.А. Программирование на Java / Н.А. Вязовик. - Москва : Национальный Открытый Университет ИНТУИТ, 2024. - 603 с. - ISBN intuit397. - URL: https://ibooks.ru/bookshelf/394426/reading

4. Гаряева В.В. Автоматизированные системы обработки информации [Электронный ресурс] : учебно-методическое пособие / Министерство науки и высшего образования Российской Федерации, Национальный исследовательский Московский государственный строительный университет, кафедра / В.В. Гаряева, А.Е. Давыдов. - Москва : МИСИ—МГСУ, 2021. - 60 с. - ISBN 978-5-7264-2816-1. - URL: https://ibooks.ru/bookshelf/391957/reading

5. Граничин О.Н. Информационные технологии в управлении / О.Н. Граничин, В.И. Кияев. - Москва : Национальный Открытый Университет ИНТУИТ, 2024. - 400 с. - ISBN 978-5-94774-986-1. - URL: https://ibooks.ru/bookshelf/394205/reading

6. Дейнека А. В. Управление персоналом организации : учебник для бакалавров. — 4-е изд., стер. / А.В. Дейнека. - Москва : Дашков и К, 2023. - 288 с. - ISBN 978-5-394-05433-4. - URL: https://ibooks.ru/bookshelf/393504/reading

7. Маглинец, Ю.А.. Анализ требований к автоматизированным информационным системам : Учебное пособие / Ю.А. Маглинец — Москва : Интуит НОУ, 2016. — 191 с. — ISBN 978-5-94774-865-9. — URL: https://book.ru/book/917556

8. Морозова Ю. В. Тестирование программного обеспечения: Учебное пособие / Ю.В. Морозова. - Томск : ТУСУР, 2020. - 120 с. - ISBN 978-5-4332-0279-5. - URL: https://ibooks.ru/bookshelf/384442/reading

9. Норенков, И. П. Автоматизированные информационные системы : учебное пособие / И. П. Норенков. - Москва : МГТУ им. Баумана, 2021. - 341 с. - (Информатика в техническом университете). - ISBN 978-5-7038-3446-6. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2009700

10. Самохвалов Э. Н. Введение в проектирование и разработку приложений на языке программирования Java : учебное пособие / Э.Н. Самохвалов, Г.И. Ревунков, Ю.Е. Гапанюк. - Москва : МГТУ им. Н.Э. Баумана, 2021. - 244 с. - ISBN 978-5-7038-4553-0. - URL: https://ibooks.ru/bookshelf/364476/reading

11. Семеновых, В. И. Проектирование автоматизированных систем : учебное пособие / В. И. Семеновых, А. А. Перминов. - Москва ; Вологда : Инфра-Инженерия, 2022. - 116 с. - ISBN 978-5-9729-1060-1. - Текст : электронный. - URL: https://znanium.com/catalog/product/1903144

12. Хазин, М. Л. Надежность, оптимизация и диагностика автоматизированных систем : учебник / М. Л. Хазин. - Москва ; Вологда : Инфра-Инженерия, 2022. - 248 с. - ISBN 978-5-9729-0890-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/1903137

13. Шитов, В. Н., Проектирование и разработка интерфейсов пользователя : учебное пособие / В. Н. Шитов, К. Е. Успенский. — Москва : КноРус, 2025. — 294 с. — ISBN 978-5-406-13754-3. — URL: https://book.ru/book/955527

Приложение 1

Листинг приложения

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.time.LocalDate;
import java.time.YearMonth;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

public class EmployeeTimeTrackingSystem {
    private boolean adminLoggedIn = false;
    private int employeesLoggedIn = 0;
    private final int MAX_EMPLOYEES = 5;

    private final Map<String, String> adminCredentials = new HashMap<>();
    private final Map<String, String> employeeCredentials = new HashMap<>();

    private final Map<String, Set<LocalDate>> employeeWorkDays = new HashMap<>();
    private final Map<String, Set<LocalDate>> employeeCheckInDays = new
HashMap<>();
    private final Map<String, Set<LocalDate>> employeeOvertimeDays = new
HashMap<>();

    private static final String DATA_FILE = "employee_data.ser";

    public EmployeeTimeTrackingSystem() {
        // Initialize credentials
        adminCredentials.put("admin", "admin123");
        employeeCredentials.put("Иван Иванов", "pass1");
```

```
        employeeCredentials.put("Петр Петров", "pass2");
        employeeCredentials.put("Сергей Сергеев", "pass3");
        employeeCredentials.put("Алексей Алексев", "pass4");
        employeeCredentials.put("Дмитрий Дмитриев", "pass5");

        loadData();
        createMainWindow();
    }

    private void createMainWindow() {
        JFrame mainFrame = new JFrame("Система учёта рабочего времени");
        mainFrame.setSize(300, 150);
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setLayout(new FlowLayout());

        JButton adminButton = new JButton("Администратор");
        JButton employeeButton = new JButton("Сотрудник");

        adminButton.addActionListener(e -> {
            if (!adminLoggedIn) {
                showLoginWindow("Администратор");
            } else {
                JOptionPane.showMessageDialog(mainFrame, "Администратор уже вошел в
систему!", "Ошибка", JOptionPane.ERROR_MESSAGE);
            }
        });

        employeeButton.addActionListener(e -> {
            if (employeesLoggedIn < MAX_EMPLOYEES) {
                showLoginWindow("Сотрудник");
            } else {
                JOptionPane.showMessageDialog(mainFrame, "Достигнуто максимальное
количество сотрудников!", "Ошибка", JOptionPane.ERROR_MESSAGE);
            }
        });
```

```
        mainFrame.add(adminButton);
        mainFrame.add(employeeButton);
        mainFrame.setVisible(true);
    }

    private void showLoginWindow(String userType) {
        JFrame loginFrame = new JFrame("Авторизация - " + userType);
        loginFrame.setSize(300, 200);
        loginFrame.setLayout(new GridLayout(3, 2));

        JLabel loginLabel = new JLabel("Логин:");
        JTextField loginField = new JTextField();
        JLabel passwordLabel = new JLabel("Пароль:");
        JPasswordField passwordField = new JPasswordField();
        JButton loginButton = new JButton("Войти");

        loginButton.addActionListener(e -> {
            String login = loginField.getText();
            String password = new String(passwordField.getPassword());

            if (userType.equals("Администратор") && adminCredentials.containsKey(login)
&& adminCredentials.get(login).equals(password)) {
                adminLoggedIn = true;
                loginFrame.dispose();
                showAdminOptionsWindow();
            } else if (userType.equals("Сотрудник") &&
employeeCredentials.containsKey(login) &&
employeeCredentials.get(login).equals(password)) {
                employeesLoggedIn++;
                loginFrame.dispose();
                showEmployeeOptionsWindow(login);
            } else {
                JOptionPane.showMessageDialog(loginFrame, "Неверный логин или
пароль!", "Ошибка", JOptionPane.ERROR_MESSAGE);
```

```
        }
    });

    loginFrame.add(loginLabel);
    loginFrame.add(loginField);
    loginFrame.add(passwordLabel);
    loginFrame.add(passwordField);
    loginFrame.add(new JLabel());
    loginFrame.add(loginButton);
    loginFrame.setVisible(true);
}

private void showEmployeeOptionsWindow(String userName) {
    JFrame employeeFrame = new JFrame(userName + " - Рабочий график");
    employeeFrame.setSize(600, 400);
    employeeFrame.setLayout(new  BorderLayout());

    JPanel calendarPanel = new JPanel();
    calendarPanel.setLayout(new GridLayout(0, 7));

    // Дни недели на русском языке
    String[] daysOfWeek = {"Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Вс"};
    for (String day : daysOfWeek) {
        calendarPanel.add(new JLabel(day, SwingConstants.CENTER));
    }

    LocalDate today = LocalDate.now();
    YearMonth yearMonth = YearMonth.of(today.getYear(), today.getMonth());
    int daysInMonth = yearMonth.lengthOfMonth();

    JLabel[] dayLabels = new JLabel[daysInMonth];
    for (int i = 1; i <= daysInMonth; i++) {
        LocalDate day = LocalDate.of(today.getYear(), today.getMonth(), i);
        JButton dayButton = new JButton(String.valueOf(i));
        Set<LocalDate> workDays = employeeWorkDays.getOrDefault(userName, new
```

```
HashSet<>());
            if (workDays.contains(day)) {
                dayButton.setBackground(Color.GREEN);
            } else {
                dayButton.setBackground(Color.RED);
            }
            dayButton.setOpaque(true);
            dayButton.setBorderPainted(false);
            dayButton.addActionListener(e -> JOptionPane.showMessageDialog(employeeFrame, "Этот день: " + (workDays.contains(day) ? "Рабочий" : "Не рабочий")));
            calendarPanel.add(dayButton);
        }

        JButton checkInButton = new JButton("Отметиться");
        JButton logoutButton = new JButton("Выход");

        checkInButton.addActionListener(e -> {
            LocalDate todayDate = LocalDate.now();
            Set<LocalDate> workDays = employeeWorkDays.getOrDefault(userName, new HashSet<>());
            if (!workDays.contains(todayDate)) {
                if (!employeeOvertimeDays.containsKey(userName)) {
                    employeeOvertimeDays.put(userName, new HashSet<>());
                }
                employeeOvertimeDays.get(userName).add(todayDate);
                JOptionPane.showMessageDialog(employeeFrame, "Вы отметились на работе, но это день не является рабочим. Отмечено как переработка.");
            } else {
                if (!employeeCheckInDays.containsKey(userName)) {
                    employeeCheckInDays.put(userName, new HashSet<>());
                }
                employeeCheckInDays.get(userName).add(todayDate);
                JOptionPane.showMessageDialog(employeeFrame, "Вы отметились на работе!");
```

```
            }
            saveData();
        });

        logoutButton.addActionListener(e -> {
            employeesLoggedIn--;
            employeeFrame.dispose();
            createMainWindow();
        });

        employeeFrame.add(calendarPanel, BorderLayout.CENTER);
        JPanel buttonsPanel = new JPanel();
        buttonsPanel.add(checkInButton);
        buttonsPanel.add(logoutButton);
        employeeFrame.add(buttonsPanel, BorderLayout.SOUTH);
        employeeFrame.setVisible(true);
    }

    private void showAdminOptionsWindow() {
        JFrame adminFrame = new JFrame("Администратор - Опции");
        adminFrame.setSize(400, 200);
        adminFrame.setLayout(new  FlowLayout());

        JButton scheduleButton = new JButton("График");
        JButton summaryButton = new JButton("Сводка");
        JButton logoutButton = new JButton("Выход");

        scheduleButton.addActionListener(e -> manageSchedule());
        summaryButton.addActionListener(e -> showSummary());
        logoutButton.addActionListener(e -> {
            adminLoggedIn = false;
            adminFrame.dispose();
            createMainWindow();
        });
```

```
        adminFrame.add(scheduleButton);
        adminFrame.add(summaryButton);
        adminFrame.add(logoutButton);
        adminFrame.setVisible(true);
    }

    private void manageSchedule() {
        JFrame scheduleFrame = new JFrame("Управление графиком сотрудников");
        scheduleFrame.setSize(500, 300);
        scheduleFrame.setLayout(new  FlowLayout());

        JLabel employeeLabel = new JLabel("Выберите сотрудника и рабочие дни");
        scheduleFrame.add(employeeLabel);

        for (String employee : employeeCredentials.keySet()) {
            JButton employeeButton = new JButton(employee);
            employeeButton.addActionListener(e -> manageEmployeeSchedule(employee));
            scheduleFrame.add(employeeButton);
        }

        scheduleFrame.setVisible(true);
    }

    private void manageEmployeeSchedule(String employee) {
        JFrame empScheduleFrame = new JFrame("График для " + employee);
        empScheduleFrame.setSize(500, 300);
        empScheduleFrame.setLayout(new BorderLayout());

        JPanel calendarPanel = new JPanel();
        calendarPanel.setLayout(new GridLayout(7, 7)); // 7 rows for days of week and dates

        // Дни недели на русском языке
        String[] daysOfWeek = {"Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Вс"};
        for (String day : daysOfWeek) {
            calendarPanel.add(new JLabel(day, SwingConstants.CENTER));
```

```
        }

        LocalDate today = LocalDate.now();
        YearMonth yearMonth = YearMonth.of(today.getYear(), today.getMonth());
        int daysInMonth = yearMonth.lengthOfMonth();
        int startDayOfWeek = today.withDayOfMonth(1).getDayOfWeek().getValue(); // Определение дня недели для первого дня месяца

        // Пустые ячейки до первого дня месяца
        for (int i = 1; i < startDayOfWeek; i++) {
            calendarPanel.add(new JLabel(""));
        }

        // Заполнение дней месяца
        for (int i = 1; i <= daysInMonth; i++) {
            LocalDate day = LocalDate.of(today.getYear(), today.getMonth(), i);
            JCheckBox dayCheckBox = new JCheckBox(String.valueOf(i));
            Set<LocalDate> workDays = employeeWorkDays.getOrDefault(employee, new HashSet<>());
            dayCheckBox.setSelected(workDays.contains(day));
            dayCheckBox.addActionListener(e -> {
                if (dayCheckBox.isSelected()) {
                    workDays.add(day);
                } else {
                    workDays.remove(day);
                }
                employeeWorkDays.put(employee, workDays);
                saveData();
            });
            calendarPanel.add(dayCheckBox);
        }

        empScheduleFrame.add(calendarPanel, BorderLayout.CENTER);
        JButton closeButton = new JButton("Закрыть");
        closeButton.addActionListener(e -> empScheduleFrame.dispose());
```

```
        empScheduleFrame.add(closeButton, BorderLayout.SOUTH);
        empScheduleFrame.setVisible(true);
    }

    private void showSummary() {
        JFrame summaryFrame = new JFrame("Сводка");
        summaryFrame.setSize(600, 400);
        summaryFrame.setLayout(new BorderLayout());

        JTextArea summaryArea = new JTextArea();
        summaryArea.setEditable(false);
        summaryFrame.add(new JScrollPane(summaryArea), BorderLayout.CENTER);

        for (Map.Entry<String, Set<LocalDate>> entry : employeeWorkDays.entrySet()) {
            String employee = entry.getKey();
            Set<LocalDate> workDays = entry.getValue();
            Set<LocalDate> checkInDays = employeeCheckInDays.getOrDefault(employee,
new HashSet<>());
            Set<LocalDate> overtimeDays = employeeOvertimeDays.getOrDefault(employee,
new HashSet<>());

            int totalWorkDays = workDays.size();
            int totalCheckIns = checkInDays.size();
            int totalOvertime = overtimeDays.size();
            int missedWork = totalWorkDays - totalCheckIns;

            StringBuilder sb = new StringBuilder();
            sb.append("Сотрудник: ").append(employee).append("\n");
            sb.append("Дни работы: ").append(totalWorkDays).append("\n");
            sb.append("Отметки на работе: ").append(totalCheckIns).append("\n");
            sb.append("Дни переработки: ").append(totalOvertime).append("\n");
            sb.append("Пропущенные дни: ").append(missedWork).append("\n");
            sb.append("\n");

            summaryArea.append(sb.toString());
```

```
        }

        JButton closeButton = new JButton("Закрыть");
        closeButton.addActionListener(e -> summaryFrame.dispose());
        summaryFrame.add(closeButton, BorderLayout.SOUTH);
        summaryFrame.setVisible(true);
    }

    private void saveData() {
        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(DATA_FILE))) {
            out.writeObject(employeeWorkDays);
            out.writeObject(employeeCheckInDays);
            out.writeObject(employeeOvertimeDays);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void loadData() {
        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(DATA_FILE))) {
            employeeWorkDays.putAll((Map<String, Set<LocalDate>>) in.readObject());
            employeeCheckInDays.putAll((Map<String, Set<LocalDate>>) in.readObject());
            employeeOvertimeDays.putAll((Map<String, Set<LocalDate>>) in.readObject());
        } catch (FileNotFoundException e) {
            // File not found, no data to load
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(EmployeeTimeTrackingSystem::new);
```