

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ»

БИРСКИЙ ФИЛИАЛ УУН_{ИТ}
ФАКУЛЬТЕТ ФИЗИКИ И МАТЕМАТИКИ
КАФЕДРА ИНФОРМАТИКИ И ЭКОНОМИКИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ПО ПРОГРАММЕ БАКАЛАВРИАТА
(ПРОЕКТНОГО ТИПА)

ШАРИПОВ ФЛАРИТ ФАРИТОВИЧ

ИНФОРМАЦИОННАЯ СИСТЕМА
ОТДЕЛА ПРАКТИКИ ПО УЧЁТУ ПРИКАЗОВ

Заведующий кафедрой
к.э.н., доцент
Г. С. Мухаметшина

Выполнила:
Студент 4 курса очной формы
обучения
Направление подготовки:
09.03.03 Прикладная информатика
Направленность (профиль):
Прикладная информатика в
информационной сфере

Руководитель
к.х.н., доцент кафедры ИиЭ
Д. В. Мальцев

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
ГЛАВА 1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ.....	5
1.1. Краткая характеристика организации.....	5
1.2. Анализ предметной области	5
1.3. Структурно-функциональная диаграмма организации «Как есть» и её описание	7
1.4. Обзор программных средств и технологий.....	11
Выводы по главе 1.....	15
ГЛАВА 2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ.....	16
2.1. Структурно-функциональная диаграмма «Как должно быть»	16
2.2. Анализ информационных потребностей пользователей	18
2.3. Диаграмма состояний	19
2.4. Инфологическая модель базы данных	21
2.5. Даталогическая модель базы данных.....	24
Вывод по главе 2	28
ГЛАВА 3. РЕАЛИЗАЦИЯ КОМПОНЕНТОВ ИНФОРМАЦИОННОЙ СИСТЕМЫ.....	30
3.1. Реализация интерфейса	30
3.2. Реализация подключения интерфейса к базе данных	31
3.3. Руководство пользователя.....	33
Вывод по главе 3	37
ЗАКЛЮЧЕНИЕ	38
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ.....	39

ВВЕДЕНИЕ

В современном образовательном процессе высшего учебного заведения играет важную роль практика, которая позволяет студентам применять полученные знания и навыки на практике, а также осуществлять профессиональную ориентацию. Организация и контроль практических занятий требует системного подхода и эффективной работы отдела практики.

Отдел практики высшего учебного заведения БФ УУНиТ (Бирский филиал Уфимского университета науки и технологий) играет важную роль в организации и координации процесса прохождения практик студентами. Однако, в связи с растущим объемом информации и сложностью работы, необходимо создание информационной системы, которая позволит автоматизировать процесс учета и формирования приказов на прохождение практики.

Данная выпускная квалификационная работа направлена на разработку и реализацию информационной системы, предназначенной для отдела практики в учебном заведении БФ УУНиТ. Система будет способствовать автоматизации учета и формированию приказов, необходимых для организации и координации процесса прохождения практик студентами. Современные информационные технологии, которые будут использованы при разработке, позволят упростить и ускорить процесс формирования приказов, а также обеспечить надежное хранение и доступ к информации о студентах и их практическом обучении.

В рамках этой работы будет проведен анализ существующих систем учёта практик, выявлены их преимущества и недостатки, а также определены требования и функциональные возможности разрабатываемой информационной системы. Будет проведено проектирование и разработка системы, включающая выбор технических средств, разработку архитектуры, создание базы данных, реализацию функциональных модулей и интерфейса пользователя.

Ожидается, что информационная система отдела практики по учету приказов значительно повысит эффективность работы отдела, упростит процессы учета и

формирования приказов, сократит время на выполнение административных задач и обеспечит более надежное и удобное хранение информации о практиках студентов.

Объект исследования – информационный процесс учёта приказов в отделе практики Бирского филиала Уфимского университета науки и технологий.

Предмет исследования – автоматизация информационного процесса учёта приказов в отделе практики Бирского филиала Уфимского университета науки и технологий посредством разработки информационной системы.

Цель исследования – разработка и реализация информационной системы отдела практики, автоматизирующей процесс учёта приказов.

Задачи исследования:

- Анализ предметной области;
- Проектирование базы данных;
- Проектирование информационной системы;
- Разработка информационной системы.

Практическая значимость – возможность применения данной системы для автоматизации учёта приказов в отделе практики образовательной организации.

Структура выпускной квалификационной работы: Выпускная квалификационная работа состоит из трех глав, заключения и списка используемых источников и литературы.

В первой главе произведено исследование предметной области, приведены структурно-функциональные диаграммы и обзор программных средств и технологий для разработки приложения.

Во второй главе выполнено проектирование автоматизированного рабочего места и выявлены требования к интерфейсу.

В третьей главе описана реализация подключения интерфейса к базе данных, создание базы данных и написано руководство пользователя.

Заключение содержит краткие результаты по каждой главе.

ГЛАВА 1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Краткая характеристика организации

Бирский филиал Уфимского университета науки и технологий – это образовательно-научный центр с многолетней историей. В филиале функционируют 6 факультетов: физики и математики; филологии и межкультурных коммуникаций; биологии и химии; факультет педагогики; социально-гуманитарный факультет; инженерно-технологический факультет. Филиал располагает достаточной материальной базой для эффективной работы. Имеются 7 учебных корпусов, 5 общежитий, библиотека, 2 спортивные учебно-тренировочные базы, спортивные лагеря «Дружба» и «Шамсутдин», агробиостанция, дендрарий, картинная галерея, музей истории академии, музей башкирской писательницы Х. Давлетшиной, зоомузей, зимний сад.

1.2. Анализ предметной области

Анализ предметной области для информационной системы отдела практики по учёту приказов включает в себя описание основных аспектов и процессов, связанных с организацией и учётом практик для студентов в высшем учебном заведении БФ УУНиТ.

1. Обзор отдела практики:

- Отдел практики является структурным подразделением высшего учебного заведения и отвечает за организацию и координацию практической деятельности студентов.
- Основная цель отдела практики заключается в обеспечении студентам возможности применения и закрепления теоретических знаний, полученных в процессе обучения, на практике.

2. Значение приказов в системе практик:

- Приказ является документом, который содержит информацию о месте и времени прохождения практики для определенной группы студентов.
- Приказы играют важную роль в организации и планировании практик, позволяя своевременно и точно определить распределение студентов по местам практики.

3. Текущие проблемы и недостатки существующей системы:

- Ручное формирование приказов требует значительных ресурсов и времени от сотрудников отдела практики.
- Возможны ошибки и неточности при ручном заполнении данных, что может привести к неправильному распределению студентов на практику.
- Отсутствие автоматизированной информационной системы затрудняет контроль и мониторинг процесса организации практик.

4. Описание информационной системы:

- Информационная система отдела практики по учету приказов будет разработана с целью автоматизации процесса формирования приказов на основе предоставленных данных.
- Система будет включать модуль ввода и обработки данных о студентах, модуль планирования практик, а также модуль формирования приказов.
- Возможны дополнительные функции системы, такие как отслеживание статуса практик, генерация отчетов и уведомлений.

5. Ожидаемые выгоды от внедрения информационной системы:

- Сокращение времени, затрачиваемого на формирование приказов и обработку данных о практиках.
- Снижение вероятности ошибок и неточностей при формировании приказов.
- Улучшение контроля и мониторинга процесса организации практик.
- Увеличение эффективности работы отдела практики и повышение удовлетворенности студентов и преподавателей.

В результате анализа предметной области можно выделить проблемы существующей системы и определить основные требования к информационной системе отдела практики. Это позволит более точно определить функциональность и особенности разрабатываемой системы, а также оценить ожидаемые выгоды от ее внедрения.

1.3. Структурно-функциональная диаграмма организации «Как есть» и её описание

Для описания информационной системы были использованы 3 методологии IDEF0, DFD, IDEF3, позволяющие анализировать работу системы:

- IDEF0 – это методология описания бизнес-процессов с помощью функциональных диаграмм. Нотация IDEF0 позволяет системно изобразить функции, обозначить их взаимосвязь между собой и внешней средой, обозначить материальные, интеллектуальные потоки, которые влияют на движение бизнес-процесса [2].

- DFD – это нотация, которая используется при моделировании информационных систем с точки зрения хранения, обработки и передачи данных. С помощью этой методологии проводится графический структурный анализ, в котором описаны внешние для системы источники данных, функции, потоки и хранилища данных, к которым имеется доступ. С помощью этой диаграммы проводится структурный анализ и проектируются информационные системы [3].

- IDEF3 – способ описания процессов с использованием структурированного метода, позволяющего эксперту в предметной области представить положение вещей как упорядоченную последовательность событий с одновременным описанием объектов, имеющих непосредственное отношение к процессу [4].

Благодаря средствам нотации IDEF0 была создана контекстная диаграмма (рис. 1.1), направленная на отражение процесса учёта приказа с точки зрения разработчика информационной системы.

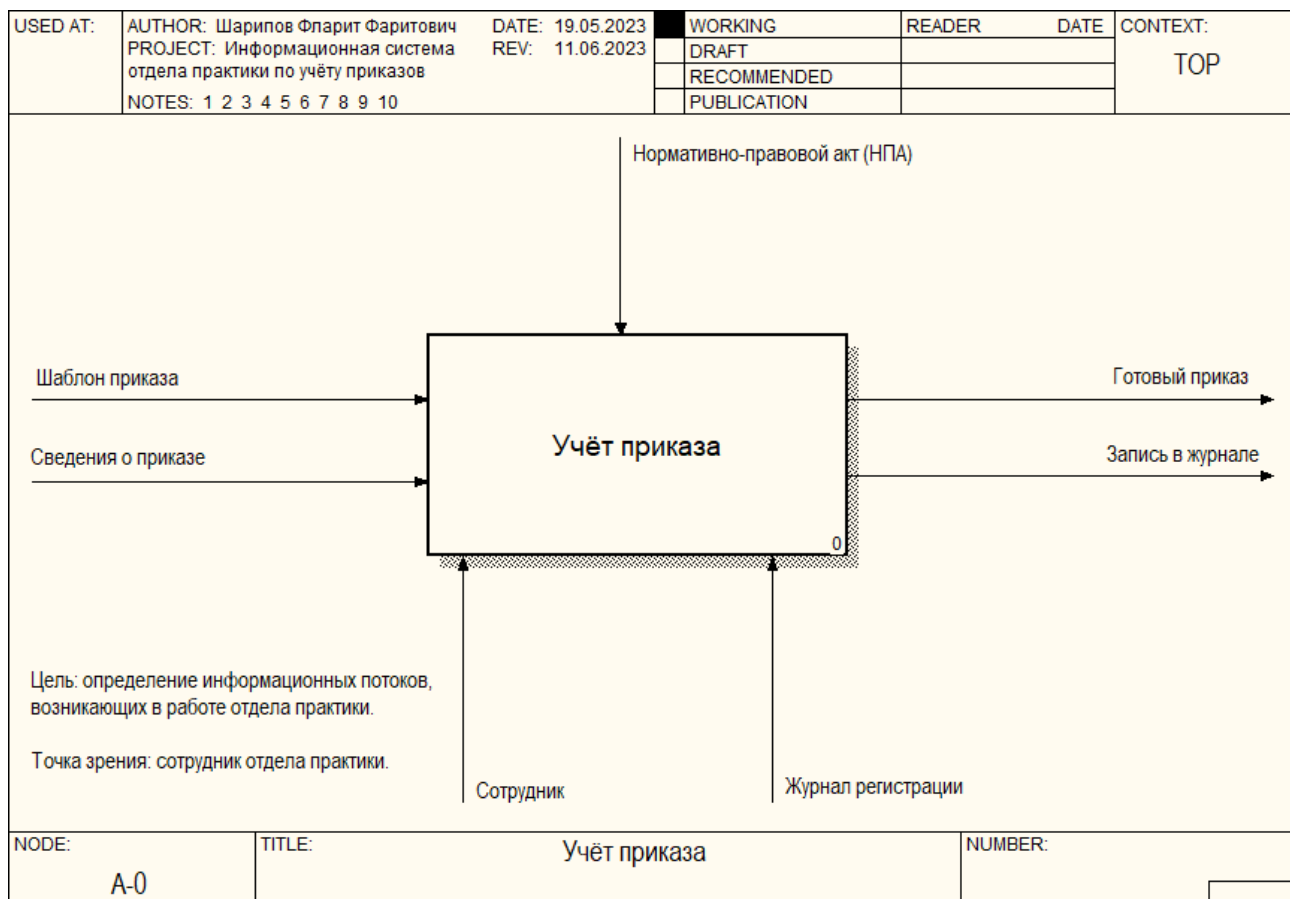


Рисунок 1.1. Контекстная диаграмма в нотации IDEF0 по типу «Как есть».

Функции различных видов стрелок данной диаграммы:

- **Вход** – стрелка, входящая в блок с левой стороны. Обозначает получаемые данные или ресурсы, которые в процессе работы будут преобразованы или использованы.
- **Выход** – стрелка, выходящая из блока с правой стороны. Обозначающая ресурс или информацию, полученную в результате работы блока.
- **Управление** – стрелка, входящая в блок с верхней стороны. Обозначает различные правила или документы, влияющие на работу.
- **Механизм** – стрелка, входящая в блок с нижней стороны. Обозначает какой-либо ресурс или пользователя, которые будут производить, использоваться в работе.

Декомпозиция контекстной диаграммы включает в себя следующие функциональные блоки (рис. 1.2):

- Обработка сведений о приказе;
- Формирование приказа;
- Запись приказа в журнал.

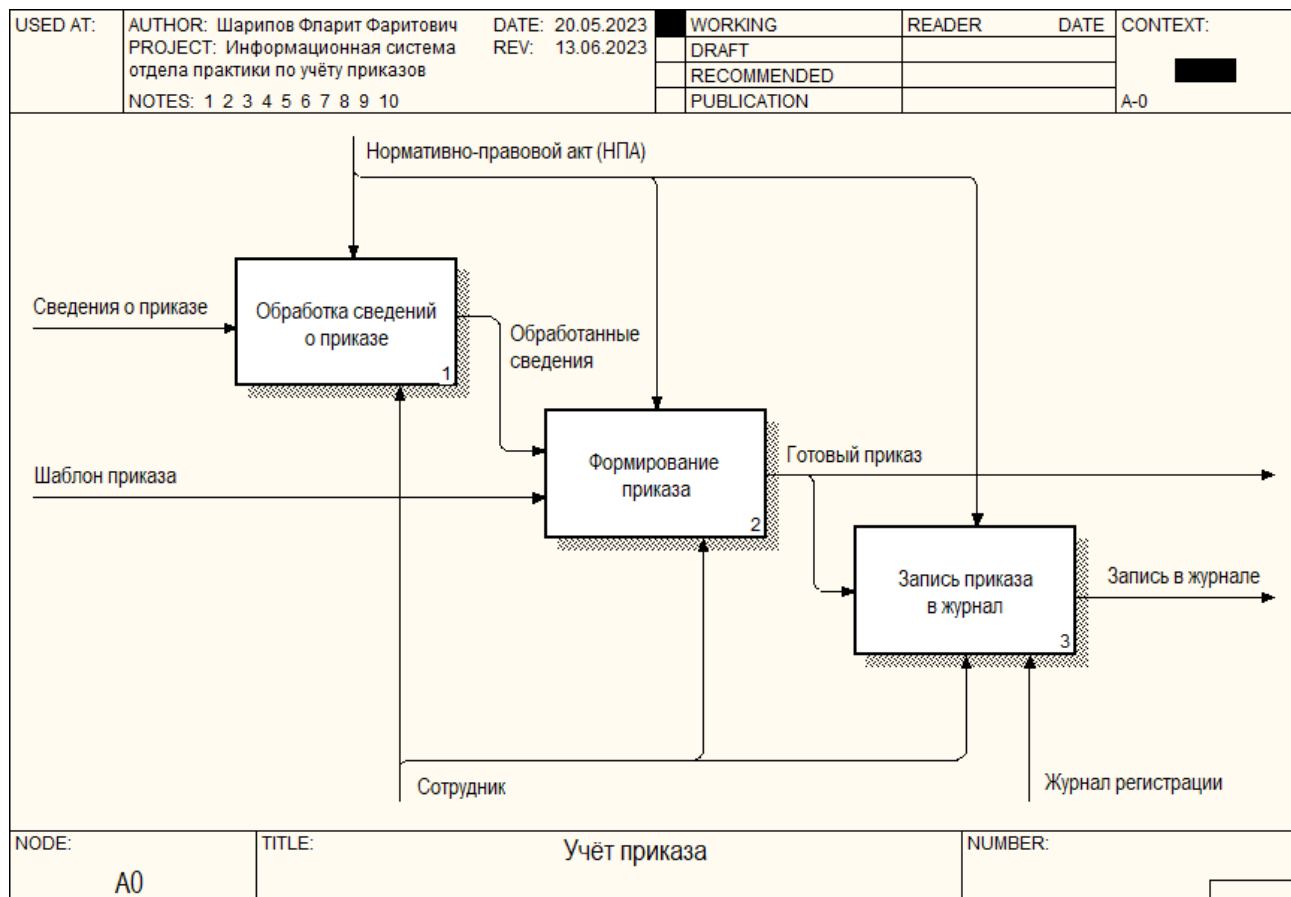


Рисунок 1.2. Результат декомпозиции контекстной диаграммы.

Декомпозиция для блока (рис. 1.3) представляет собой ввод вида работ в нотации IDEF3.

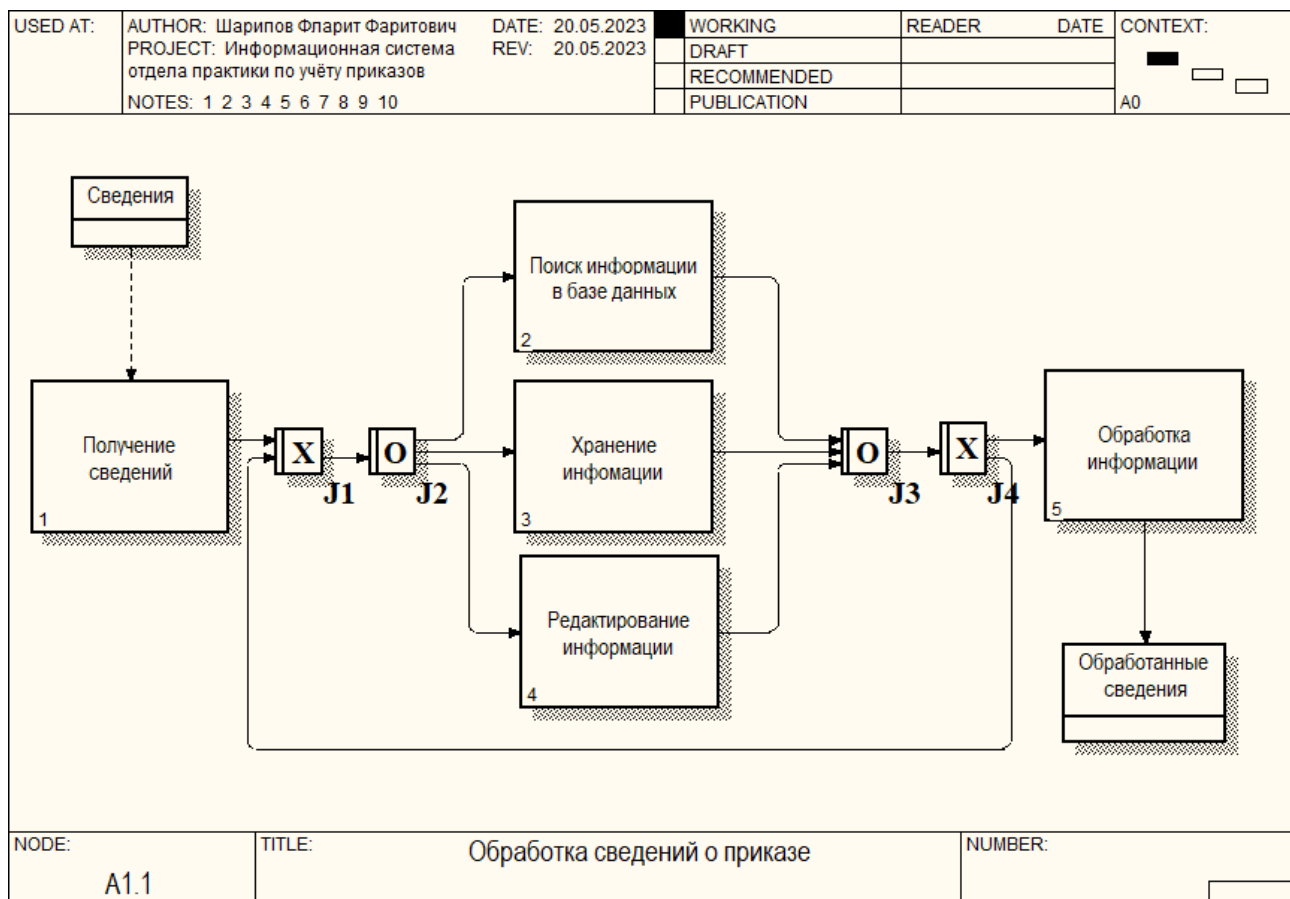


Рисунок 1.3. Декомпозиция блока «Анализ и обработка данных» в нотации IDEF3.

Декомпозиция, показанная на рисунке 1.4, описывает то, как происходит процесс формирования приказа на практику, т.е.:

- Сначала сотрудник ищет необходимые данные в базе данных для заполнения приказа на практику с учётом обработанных сведений;
- После нахождения этих данных, сотрудник заполняет приказ и сохраняет его.

С учетом выполненных операций, на выходе получается готовый приказ, который впоследствии будет сохранён в журнале регистрации.

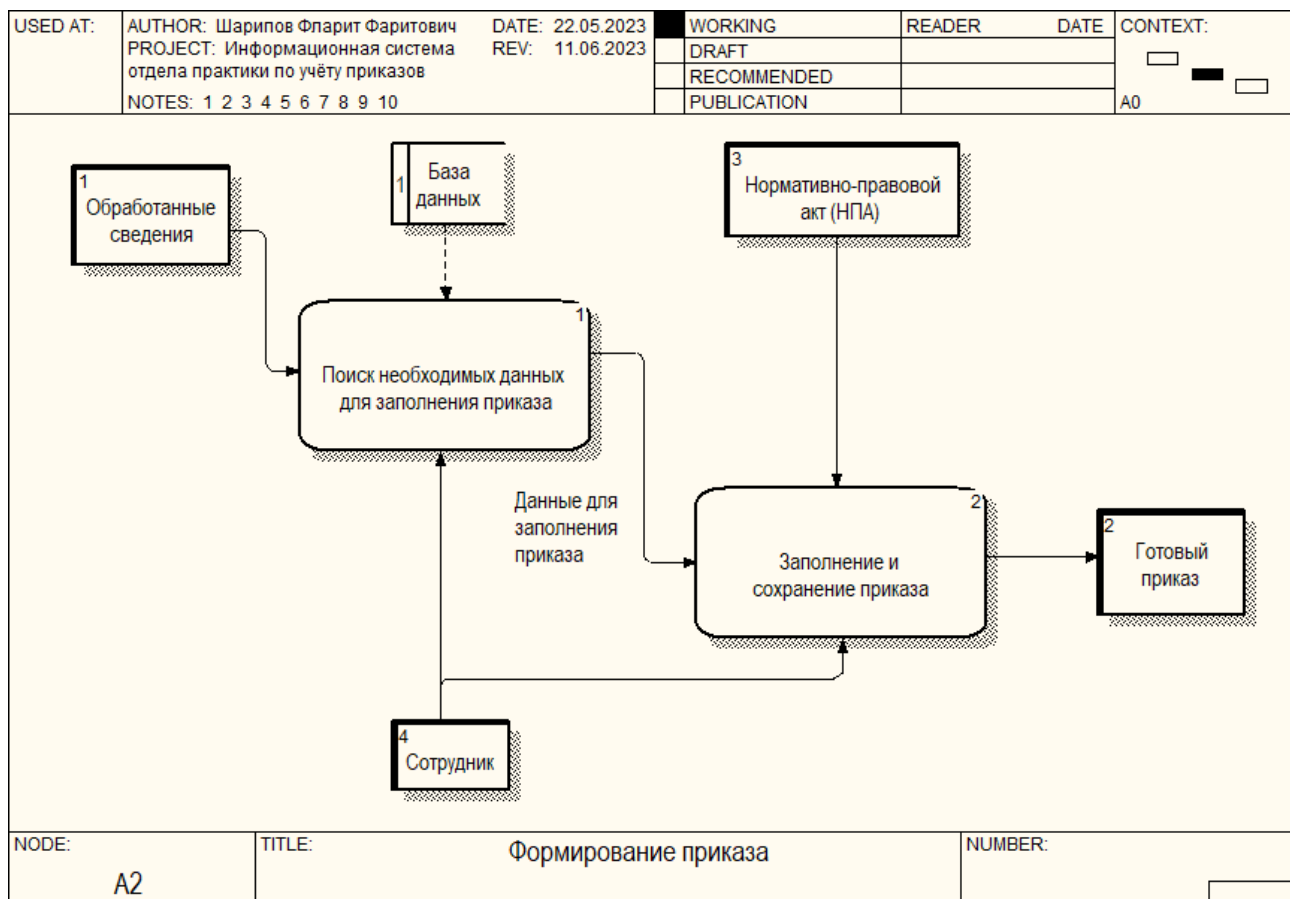


Рисунок 1.4. Декомпозиция блока «Формирование приказа» в нотации DFD.

1.4. Обзор программных средств и технологий

Выбор языка программирования

Для разработки информационной системы существует большой выбор технологий. Самыми популярными языками программирования для разработки desktop-приложений являются: C, C++, C#, Go, Python, Java.

Язык программирования C – один из самых популярных и распространенных языков. Он представляет компилируемый язык программирования общего назначения со статической типизацией, разработанный в 1969—1973 годах в компании Bell Labs программистом Деннисом Ритчи (Dennis Ritchie) [5]. C предоставляет мощные возможности для низкоуровневого программирования, такие как работа с памятью и указателями. Однако он требует более подробного управления памятью, что может быть сложно для новичков. C обычно используется

для создания системного программного обеспечения и высокопроизводительных приложений.

Язык программирования C++ – высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день C++ является одним из самых популярных и распространенных языков [6]. C++ является расширением языка C, добавляющим объектно-ориентированные возможности. Он предоставляет более высокий уровень абстракции, чем C, и широкий набор библиотек для различных задач. C++ имеет сильную поддержку многопоточности, обработки исключений и шаблонов. Он позволяет разработчикам создавать эффективные и масштабируемые приложения, включая desktop-приложения с графическим интерфейсом.

C# – современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript [7]. C# обладает простым и интуитивным синтаксисом, поддержкой объектно-ориентированного программирования и многопоточности. Он предоставляет множество инструментов и библиотек для создания desktop-приложений под Windows, включая приложения с графическим интерфейсом с использованием технологии Windows Forms или WPF.

Go – это компилируемый статически типизированный язык программирования от компании Google. Язык Go развивается как open source, то есть представляет проект с открытым исходным кодом. Этот язык является кроссплатформенным, он позволяет создавать программы под различные операционные системы - Windows, Mac OS, Linux, FreeBSD. Код обладает переносимостью: программы, написанные для одной из этих операционных систем, могут быть легко с перекомпиляцией перенесены на другую ОС. [8]. Go известен своей простотой и эффективностью в разработке. Он имеет сборщик мусора и поддерживает параллельные вычисления. Он предоставляет стандартную библиотеку с множеством функций, включая

возможности создания desktop-приложений. Go часто используется для разработки высокопроизводительных и масштабируемых серверных приложений, но также может быть использован для создания desktop-приложений.

Python – популярный высокоуровневый язык программирования, который предназначен для создания приложений различных типов. Это и веб-приложения, и игры, и настольные программы, и работа с базами данных. Довольно большое распространение этот язык получил в области машинного обучения и исследований искусственного интеллекта [9]. Python известен своей простотой, читаемостью и разнообразием библиотек. Он имеет обширное сообщество разработчиков и активную экосистему с инструментами для создания desktop-приложений, таких как PyQt или Tkinter. Python широко используется в научных исследованиях, анализе данных, автоматизации задач и разработке прототипов.

Java – один из самых распространённых и популярных языков программирования. Он задумывался как универсальный язык программирования, который можно применять для различного рода задач. К настоящему времени Java превратилась из просто универсального языка в целую платформу и экосистему, которая объединяет различные технологии, используемые для целого ряда задач: от создания десктопных приложений до написания крупных веб-порталов и сервисов [10]. Java известен своей платформенезависимостью и возможностью создания кросс-платформенных приложений. Он имеет обширную стандартную библиотеку и множество фреймворков, которые упрощают создание desktop-приложений, таких как JavaFX или Swing. Java широко используется в корпоративной среде и разработке масштабных приложений.

В данной работе для разработки использован язык программирования C#. Он интегрирован с платформой .NET, поддерживает ООП, обладает простым синтаксисом и хорошей читаемостью. C# обеспечивает поддержку многопоточности, а также имеет широкую поддержку инструментов и фреймворков.

Среда разработки

В случае выбора языка программирования C#, выбор среды разработки сужается и состоит из следующих вариантов: Visual Studio, Visual Studio Code, JetBrains Rider.

Visual Studio – интегрированная среда разработки программного обеспечения для платформы .NET, разрабатываемая компанией Microsoft. Имеет огромное количество инструментов для работы, которые оптимизированы для C#. Присутствует бесплатная версия[11].

Visual Studio Code – своего рода «лёгкий» редактор кода для кроссплатформенной разработки «веб-» и облачных приложений. Является полностью бесплатным[12]. Имеет поддержку огромного количества языков программирования и многочисленные плагины для них. Минусом является то, что направление больше подходит для веб-разработки на JavaScript из-за обильного количества плагинов для него[13].

Rider – кроссплатформенная интегрированная среда разработки программного обеспечения для платформы .NET, разрабатываемая компанией JetBrains [14]. Является довольно мощной средой, но не распространяется бесплатно.

В данной работе для разработки использован Visual Studio Community Edition, так как имеет все необходимые инструменты для разработки.

Платформа пользовательского интерфейса

Windows Forms – интерфейс программирования приложений, отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде [15].

Windows Presentation Foundation – аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework, использующая язык XAML [16].

В данной работе для разработки использован Windows Forms.

База данных

В качестве базы данных, выбор пал на использование SQLite – встраиваемая кроссплатформенная БД, которая поддерживает достаточно полный набор команд SQLи доступна с открытым исходным кодом [17].

Таким образом, в качестве протокола обмена используются вызовы функций API библиотеки SQLite [18]. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных в единственном стандартном файле на том компьютере, на котором выполняется программа.

Выводы по главе 1

В первой главе произведено исследование предметной области.

В первой части главы описана краткая характеристика организации.

Во второй части главы произведен анализ предметной области, определена необходимость разработки информационной системы отдела практики по учёту приказов.

В третьей части главы была рассмотрена структура прототипа информационной системы с использованием нотаций бизнес - моделирования IDEF0, IDEF3.

В четвёртой части главы произведен обзор программных средств, предоставляющих возможность создания прототипа информационной системы. Для разработки выбрана интегрированная среда Microsoft Visual Studio 2010 (язык программирования C#).

ГЛАВА 2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1. Структурно-функциональная диаграмма «Как должно быть»

В ходе разработки информационной системы значительно переработана описанная ранее диаграмма “Как есть” и в результате получена структурно функциональная диаграмма “Как должно быть” (рис. 2.1) [27,28].

Управление, входные и выходные данные не изменены. К механизмам добавлена информационная система (приложение), участвующая в большинстве функций.

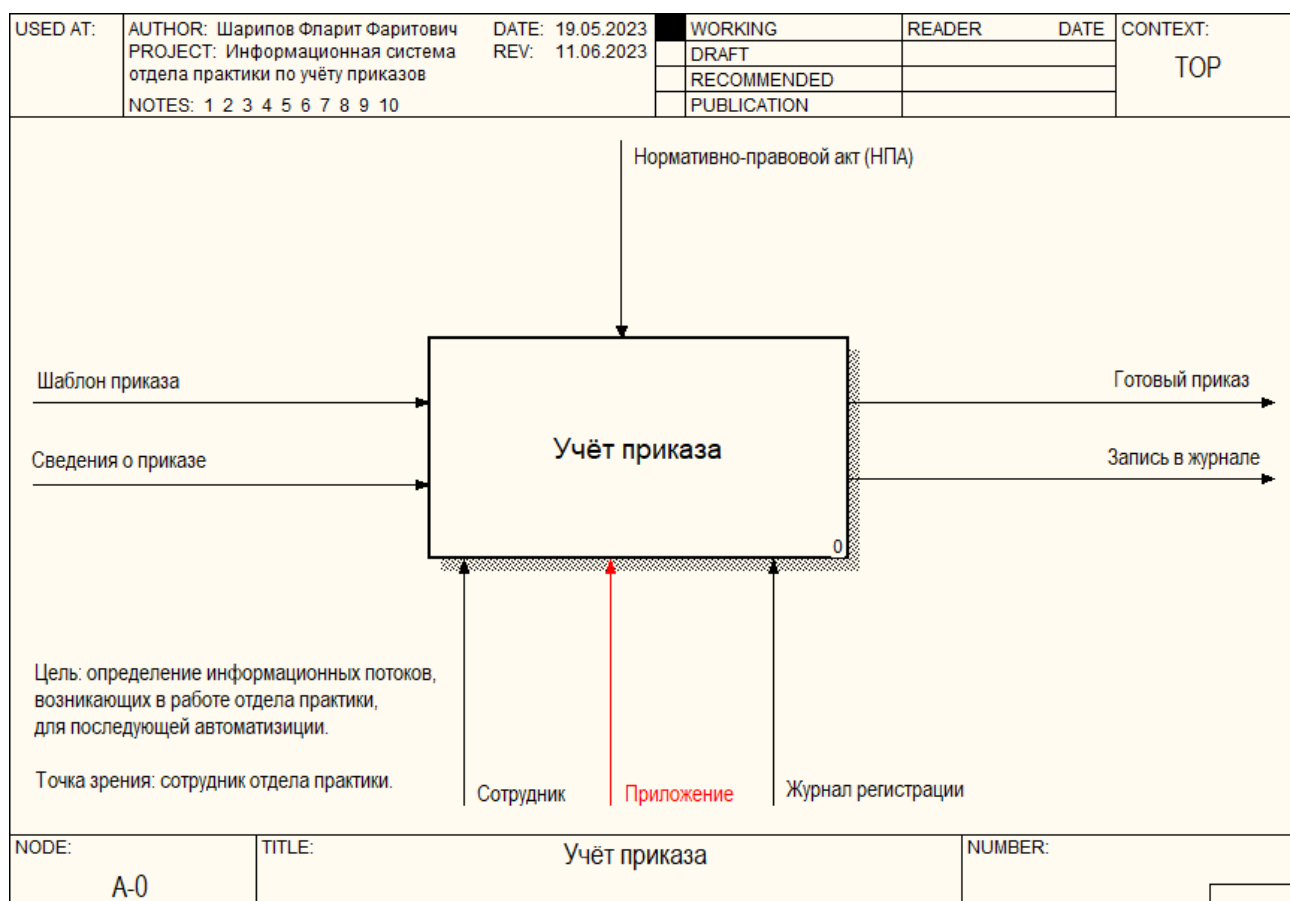


Рисунок 2.1. Контекстная диаграмма «Как должно быть».

Декомпозиция контекстной диаграммы (рис. 2.2) включает в себя следующий новый механизм: «Приложение», которое будет и обрабатывать сведения о приказе,

и формировать приказ о практике, и записывать в электронный журнал информацию о готовом приказе.

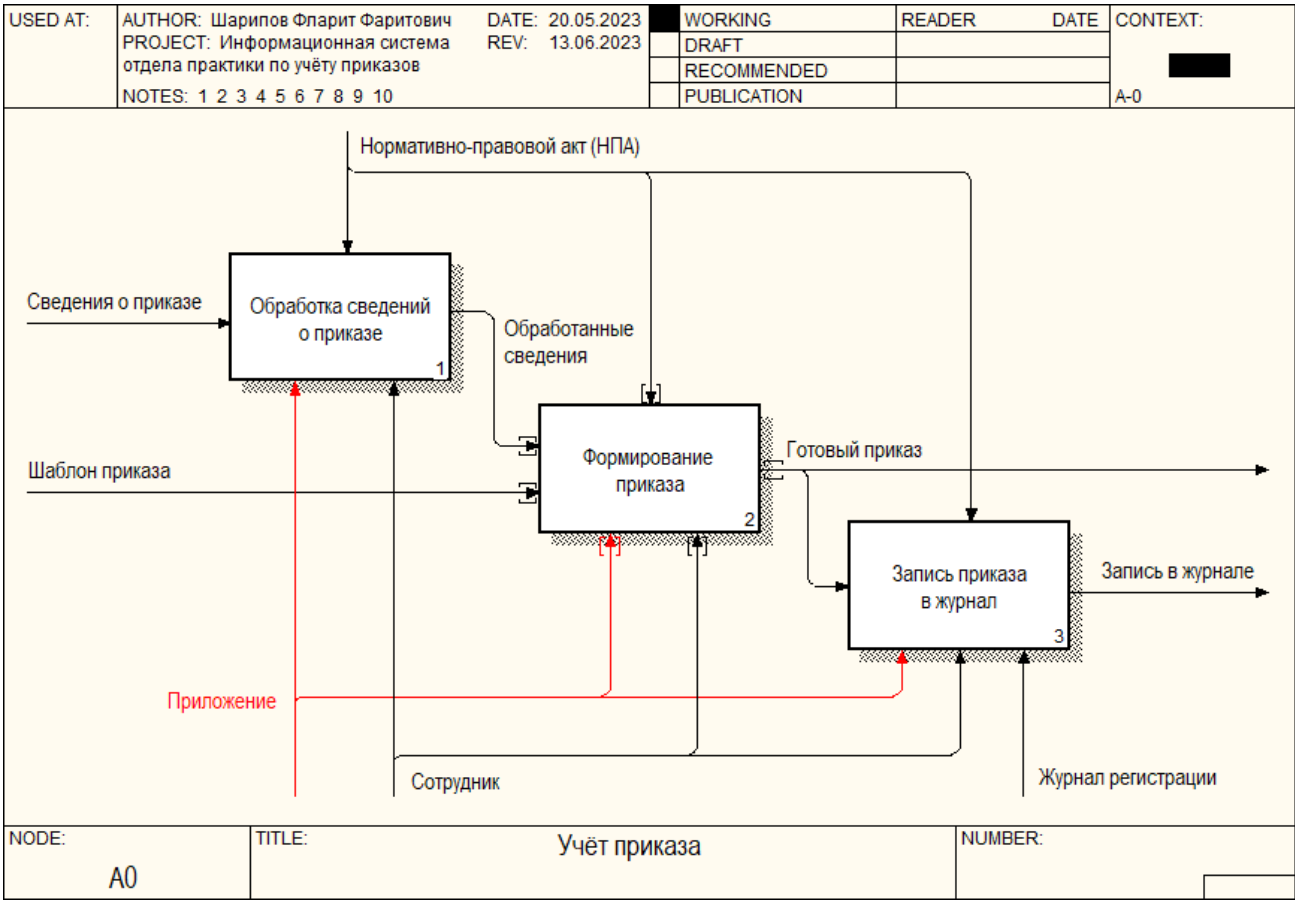


Рисунок 2.2.Декомпозиция контекстной диаграммы «Как должно быть».

С учётом спроектированного процесса учёта приказов, после процесса внедрения информационной системы, данная операция будет происходить в следующей последовательности:

- Система обрабатывает поступившие сведения о приказе;
- Исходя из обработанных сведений и выбранного шаблона приказа сотрудником, система формирует готовый приказ;
- Заключающим действием система записывает информацию о приказе в журнал регистрации.

2.2. Анализ информационных потребностей пользователей

Диаграмма прецедентов или диаграмма вариантов использования – это диаграмма, на которой отражаются отношения между актерами и прецедентами системы, позволяющая описывать систему на концептуальном уровне.

Прецедент – это возможность системы, часть ее функциональности, с помощью которой актер может получить нужный ему результат. Прецедент является соответствием для отдельного сервиса системы и определяет варианты ее использования.

При выявлении потребностей, нужно учитывать, что на данном этапе разработки информационной системы пользователь будет выступать только один. Следовательно, требования необходимо учитывать под данного пользователя. Рассмотрим диаграмму прецедентов, показывающую способы взаимодействия пользователя с системой (рис. 2.3).

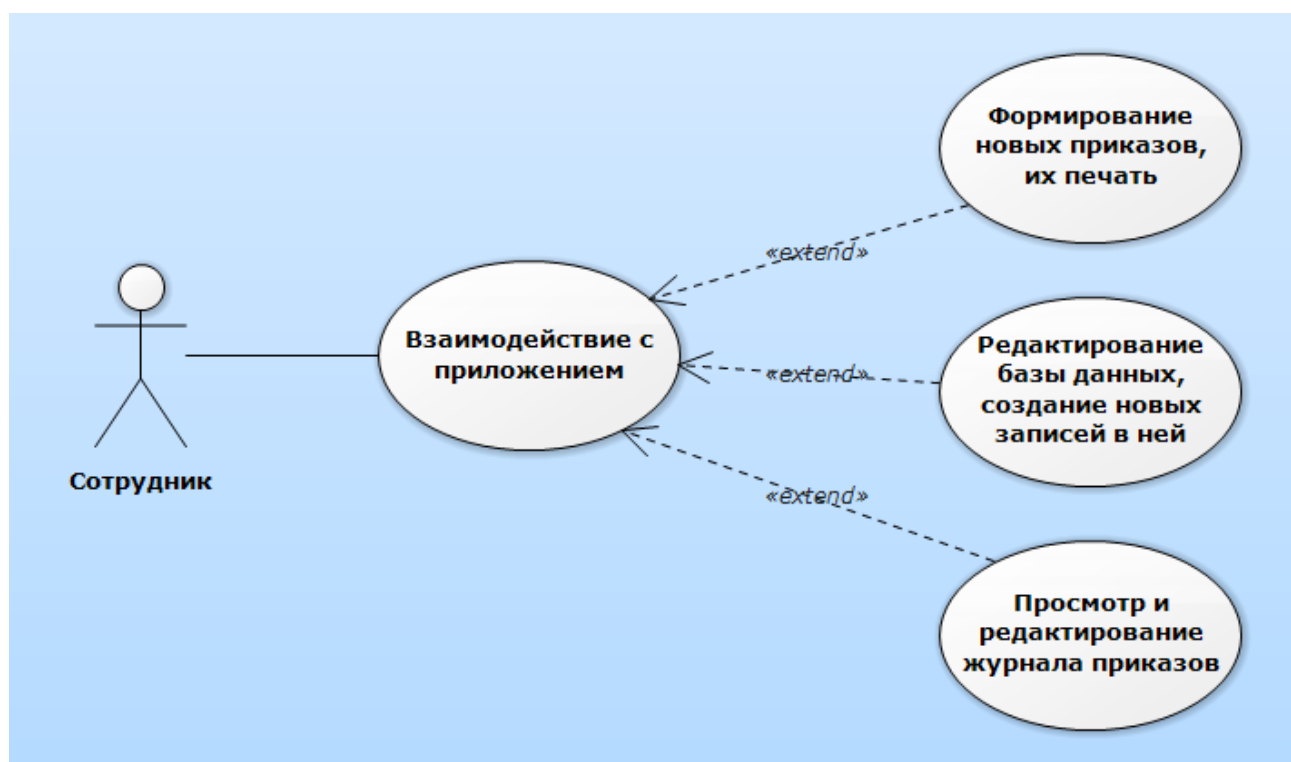


Рисунок 2.3. Диаграмма прецедентов.

Информационная система отдела практики выполняет следующие функции:

1. «Формирование новых приказов, их печать» - система позволяет создавать новые приказы посредством ввода входных данных, таких как тип практики, группа студентов, которые будут проходить практику, даты проведения практики и т.д., и её печать в бумажном виде;
2. «Редактирование базы данных, создание новых записей в ней» - система позволяет изменять записи, например, при отчислении студента добавлять его в список отчисленных; или создавать записи, например, при приёме нового сотрудника на работу;
3. «Просмотр и редактирование журнала приказов» - система позволяет просматривать журнал регистрации приказов в электронном виде и редактировать его в случае необходимости.

2.3. Диаграмма состояний

Диаграмма состояний – это тип диаграммы, используемый в UML для описания всех состояний системы и переходов между ними. Она отображает разрешенные состояния и переходы, а также события, которые влияют на эти переходы. Кроме этого помогает визуализировать весь жизненный цикл объектов и, таким образом, помогает лучше понять системы, основанные на состояниях[30].

На диаграмме состояний моделируемая система или объект представлена в виде набора состояний, которые представляют собой конкретные условия или ситуации, в которых система может существовать. Эти состояния связаны переходами, которые отображают возможные изменения в состоянии системы. Кроме того, диаграммы состояний могут включать события, действия и условия, которые запускают переходы между состояниями.

Основная цель диаграммы состояний состоит в том, чтобы фиксировать и передавать поведение системы или объекта с течением времени. Это обеспечивает четкую и интуитивно понятную визуализацию того, как система ведет себя в ответ на внешние и внутренние события и как она переходит из одного состояния в другое. Диаграммы состояний помогают понять поведение системы, определить

возможные последовательности состояний и проанализировать сложные взаимодействия между состояниями и событиями.

Кроме того, диаграммы состояний облегчают проектирование и внедрение программных систем. Они служат основой для разработчиков, предоставляя визуальное представление о поведении системы, облегчая понимание требований и передачу информации о них. Диаграммы состояний также помогают выявлять потенциальные ошибки или несоответствия в поведении системы, позволяя своевременно обнаруживать и устранять неполадки.

В целом, диаграммы состояний являются важными инструментами в разработке программного обеспечения. Они способствуют лучшему пониманию, коммуникации и анализу поведения системы, что приводит к созданию более надежных программных решений.

Ниже представлена диаграмма состояний, на которой показаны основные состояния программы с точки зрения пользователя (рис. 2.4).

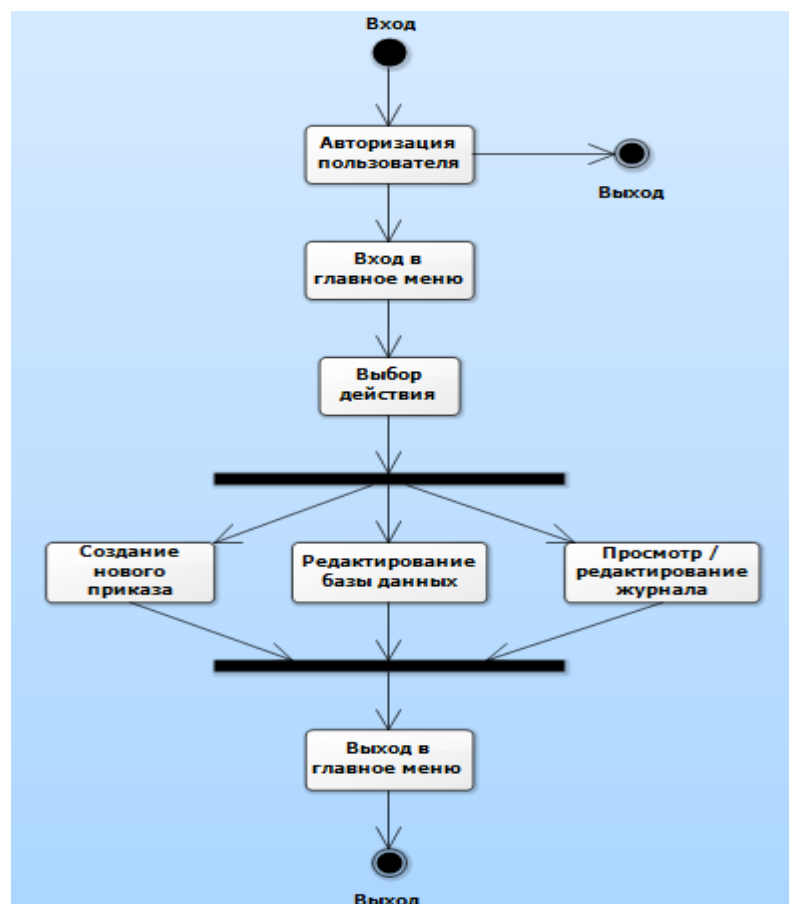


Рисунок 2.4. Диаграмма состояний.

2.4. Инфологическая модель базы данных

Инфологическая модель представляет собой описание предметной области, основанное на анализе семантики объектов и явлений, выполненное без ориентации на использование в дальнейшем программных или технологических компьютерных средств [19].

Нотация Питера Чена, необходимая для представления инфологической модели, использует различные символы и диаграммы для иллюстрации сущностей (объектов или концепций), отношений между сущностями и атрибуты, связанные с сущностями. Ключевые компоненты этой нотации включают сущности, атрибуты и отношения.

Сущности представляют собой различные объекты или концепции в базе данных, такие как клиенты, продукты или заказы. Атрибуты - это характеристики или свойства сущностей, предоставляющие дополнительную информацию о них. Отношения изображают ассоциации или связи между объектами, показывая, как они взаимодействуют или соотносятся друг с другом.

Основная цель инфологической модели – обеспечить высокоуровневое представление структуры базы данных, позволяющее заинтересованным сторонам визуализировать и понимать взаимосвязи и зависимости между различными объектами. Он служит основой для проектирования и внедрения системы баз данных, помогая в организации, хранении и извлечении данных.

Используя инфологическую модель, разработчики могут идентифицировать и определять сущности, атрибуты и взаимосвязи, которые необходимо представить в базе данных. Это помогает прояснить требования и ограничения системы, способствуя эффективной коммуникации между проектировщиками, разработчиками и заинтересованными сторонами.

Кроме того, инфологическая модель способствует обеспечению целостности и непротиворечивости данных. Это позволяет определить ключевые ограничения и правила, которые управляют базой данных, гарантируя, что данные остаются

точными и надежными. Он служит ориентиром для поддержания целостности базы данных в течение ее жизненного цикла, включая обновления данных, модификации и запросы.

Таким образом, инфологическая модель, основанная на нотации Питера Чена, представляет собой концептуальное представление базы данных, которое помогает в понимании, проектировании и обслуживании системы баз данных. Это позволяет заинтересованным сторонам визуализировать структуру и взаимосвязи внутри базы данных, что приводит к эффективной коммуникации, целостности данных и эффективному управлению данными.

В информационной системе отдела практики по учёту приказов определены следующие сущности:

- Факультет: ID факультета, название факультета;
- Кафедра: ID кафедры, название кафедры, ID руководителя кафедры, ID факультета;
- Сотрудник: ID сотрудника, фамилия сотрудника, имя сотрудника, отчество сотрудника, ID кафедры, ID должности сотрудника, ID статуса в деканате;
- Должность сотрудника: ID должности, название должности;
- Форма обучения: ID формы обучения; наименование формы обучения;
- Специальность: ID специальности, код специальности, наименование специальности, соответствующий номер группы, ID кафедры, ID формы обучения;
- Приказ: ID приказа, номер приказа, дата формирования приказа, ID типа практики, ID руководителя практики, ID специальности, ID курса обучения, путь к файлу приказа, ID пользователя, сформировавшего приказ;
- Пользователь: ID пользователя, логин пользователя, пароль пользователя, роль пользователя, ID сотрудника;
- Тип практики: ID типа практики, наименование типа практики.
- Студент: ID студента, фамилия студента, имя студента, отчество студента, ID курса обучения, ID специальности;

-

По диаграмме (рис. 2.2) видим, что сущность «Студент» имеет атрибут «Идентификатор студента», который также является первичным ключом, задаваемым базой данных. «Студент» учится по «Специальности», первичный ключ которой («Идентификатор специальности») является внешним ключом сущности «Студент»; связь осуществляется по отношению «один-ко-многим», т. к. по одной специальности могут обучаться много студентов.

- от сущности «Факультет» по отношению «один-ко-многим», т. к. в одном факультете может содержаться несколько специальностей;
- от сущности «Форма обучения» по отношению «многие-ко-многим», т. к. во множестве специальностей могут применяться множество форм обучения.

23

2.5. Даталогическая модель базы данных

Даталогическая модель – это модель логического уровня системы, представляющая собой отображение логических связей между элементами базы данных [20]. Эта модель разрабатывается с учетом конкретной реализации СУБД и на основе её инфологической модели.

Основная цель даталогической модели данных состоит в том, чтобы обеспечить концептуальную основу для проектирования и реализации системы баз данных. Это помогает разработчикам баз данных понять требования к данным приложения или бизнес-процесса и преобразовать их в логическую структуру. Определяя сущности и их атрибуты, он создает основу для хранения и извлечения данных.

Даталогическая модель данных также обеспечивает целостность и непротиворечивость данных за счет применения ограничений и правил к данным. Это позволяет разработчикам определять связи и зависимости между объектами, гарантируя точное представление и обслуживание данных. Это помогает предотвратить аномалии и несоответствия данных, что приводит к повышению качества данных.

Кроме того, даталогическая модель данных действует как инструмент коммуникации между заинтересованными сторонами, участвующими в процессе разработки базы данных. Это позволяет дизайнерам, разработчикам и бизнес-пользователям иметь общее представление о структуре данных и их семантике.

Таким образом, логическая модель данных необходима для проектирования, организации и управления данными в системе баз данных. Он обеспечивает структурированное представление требований к данным, обеспечивает целостность данных и способствует эффективной коммуникации между заинтересованными сторонами.

На диаграмме сущности (рис 2.3) представлены в виде прямоугольников с атрибутами внутри. Каждый атрибут обозначен соответствующим типом данных, каким он записан в базе данных [21]. Связи между сущностями реализуются с

помощью первичных и внешних ключей. Обозначение связей между сущностями на диаграмме показано в виде стрелок с надписью внешнего ключа, которое расположено сверху этой самой стрелки.

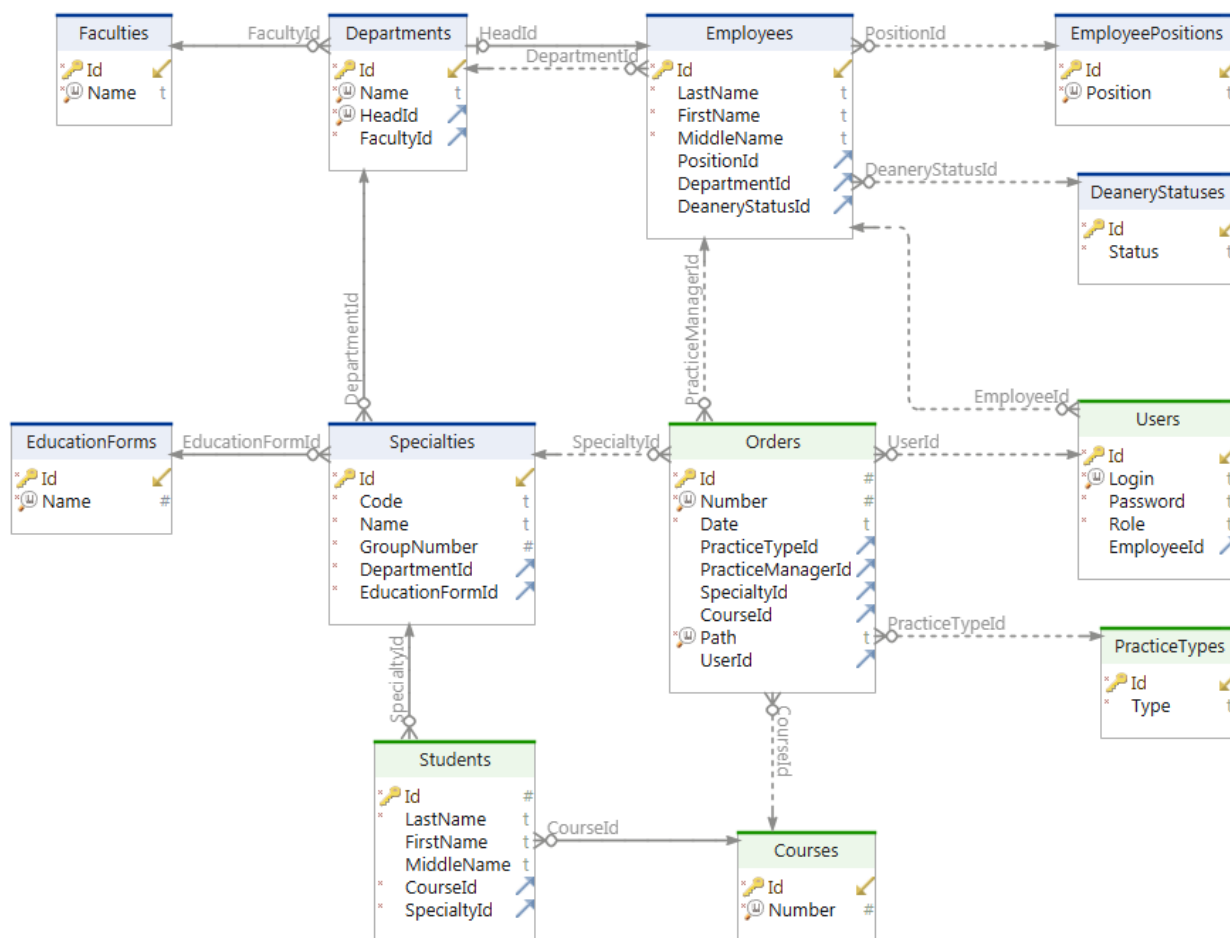


Рисунок 2.3. Даталогическая модель базы данных.

Для работы с базой данных была выбрана библиотека .NET Entity Framework 6. Entity Framework в сочетании с LINQ (Language-Integrated Query) представляет собой реализацию ORM для платформы .NET Framework от компании Microsoft. Entity Framework содержит механизмы создания и работы с сущностями базы данных через объектно-ориентированный код на языке, совместимым с CLR. LINQ представляет собой библиотеку, расширяющую возможности C#, и облегчающую создание запросов [21]. EF6 позволяет работать с базами данных, но представляет собой более высокий уровень абстракции: EF6 позволяет абстрагироваться от самой

базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Для информационной системы были определены следующие модели:

1. Course, определяющий сущность курса обучения:

```
public class Course
{
    [Key]
    public int Id { get; set; }
    public int Number { get; set; }
}
```

2. DeaneryStatus, определяющий сущность статуса в деканате:

```
public class DeaneryStatus
{
    [Key]
    public int Id { get; set; }
    public string Status { get; set; }
}
```

3. Department, определяющий сущность кафедры:

```
public class Department
{
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
    public int HeadId { get; set; }
    public int FacultyId { get; set; }
}
```

4. EducationForm, определяющий сущность формы обучения:

```
public class EducationForm
{
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
}
```

5. EmployeePosition, определяющий сущность должности сотрудника:

```
public class EmployeePosition
{
    [Key]
    public int Id { get; set; }
    public string Position { get; set; }
}
```

```
}
```

6. Employee, определяющий сущность сотрудника:

```
public class Employee
{
    [Key]
    public int Id { get; set; }
    public string LastName { get; set; }
    public string FirstName { get; set; }
    public string MiddleName { get; set; }
    public int? PositionId { get; set; }
    public int? DepartmentId { get; set; }
    public int? DeaneryStatusId { get; set; }
}
```

7. Faculty, определяющий сущность факультета:

```
public class Faculty
{
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
}
```

8. Order, определяющий сущность приказа:

```
public class Order
{
    [Key]
    public int Id { get; set; }
    public int Number { get; set; }
    public string Date { get; set; }
    public int? PracticeTypeId { get; set; }
    public int? PracticeManagerId { get; set; }
    public int? SpecialtyId { get; set; }
    public int? CourseId { get; set; }
    public string Path { get; set; }
    public int? UserId { get; set; }
}
```

9. PracticeType, определяющий сущность типа практики:

```
public class PracticeType
{
    [Key]
    public int Id { get; set; }
    public string Type { get; set; }
}
```

10. Specialty, определяющий сущность специальности:

```
public class Specialty
{
    [Key]
    public int Id { get; set; }
    public string Code { get; set; }
    public string Name { get; set; }
    public int GroupNumber { get; set; }
    public int DepartmentId { get; set; }
    public int EducationFormId { get; set; }
}
```

11. Student, определяющий сущность студента:

```
public class Student
{
    [Key]
    public int Id { get; set; }
    public string LastName { get; set; }
    public string FirstName { get; set; }
    public string MiddleName { get; set; }
    public int CourseId { get; set; }
    public int SpecialtyId { get; set; }
}
```

12. User, определяющий сущность пользователя:

```
public class User
{
    [Key]
    public int Id { get; set; }
    public string Login { get; set; }
    public string Password { get; set; }
    public string Role { get; set; }
    public int? EmployeeId { get; set; }
}
```

Вывод по главе 2

Во второй главе произведено проектирование информационной системы.

В первой части представлена структурно-функциональная диаграмма «Как должно быть» и её описание.

Во второй части проведен анализ информационных потребностей пользователей и представлена модель прецедентов (вариантов использования).

В третьей части представлена диаграмма состояний.

В четвёртой части описана инфологическая модель, отражающая необходимые сущности для разработки приложения.

В пятой части представлена схема даталогической модели приложения и основные модели необходимые для разработки базы данных.

ГЛАВА 3. РЕАЛИЗАЦИЯ КОМПОНЕНТОВ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1. Реализация интерфейса

Интерфейс информационной системы реализован с помощью платформы пользовательского интерфейса для создания классических приложений Windows под названием Windows Forms[26].

Весь пользовательский интерфейс состоит из форм – в Windows Forms это визуальная поверхность, на которой выводится информация для пользователя[26].

В данной информационной системе используются следующие формы:

1. AddDepartmentForm – форма добавления кафедры;
2. AddEmployeeForm – форма добавления сотрудника;
3. AddingStudentForm – форма быстрого добавления студента;
4. AddPracticeTypeForm – форма добавления типа практики;
5. AddStudentForm – форма добавления студента;
6. AddStudentsForm – форма добавления нескольких студентов;
7. CrudDepartmentsForm – форма для просмотра информации о кафедрах;
8. CrudEmployeesForm – форма для просмотра информации о сотрудниках;
9. LoadingForm – форма отображения загрузки при переключения между окнами;
- 10.LoginForm – форма авторизации для входа в систему и аутентификации пользователя;
- 11.MainForm – главная форма, где расположены основные элементы взаимодействия с системой;
- 12.OrderCreatingForm – форма для формирования нового приказа;
- 13.OrdersJournalForm – форма для просмотра и редактирования журнала регистрации приказов.
- 14.RegistrationForm – форма регистрации для создания нового аккаунта пользователя;

- 15.SettingsForm – форма для настроек системы и базы данных;
- 16.UpdateDepartmentForm – форма для изменения руководителя кафедры.

В каждой из форм присутствуют элементы управления, например, в форме LoginForm (рис. 3.1) есть 6 элементов:

1. LoginLable – надпись «Логин»;
2. LoginTextBox – текстовое поле для ввода логина;
3. PasswordLable – надпись «Пароль»;
4. PasswordTextBox – текстовое поле для ввода пароля;
5. LoginButton – кнопка, при нажатии на которую открывается главная форма MainForm, если пройдена аутентификация пользователя;
6. RegistrationButton – кнопка, при нажатии на которую открывается форма регистрации RegistrationForm.

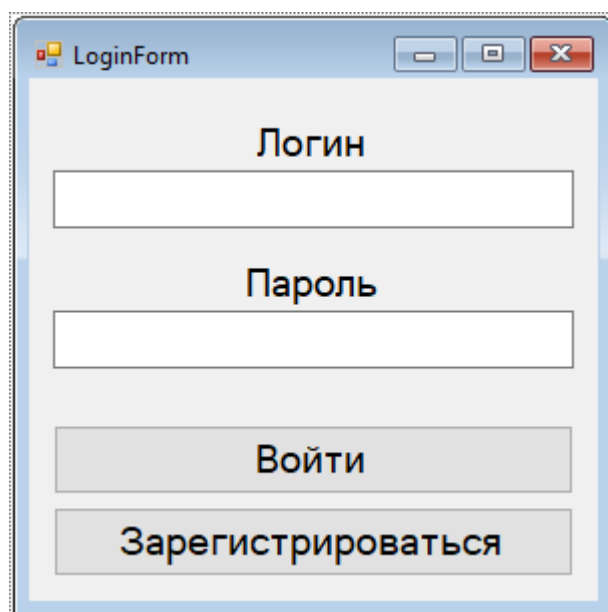
The image shows a screenshot of a Windows application window titled "LoginForm". The window contains a login form with the following elements: a label "Логин" above a text input field, a label "Пароль" above another text input field, and two buttons at the bottom labeled "Войти" and "Зарегистрироваться". The window has a standard Windows title bar with minimize, maximize, and close buttons.

Рисунок 3.1. Форма авторизации LoginForm.

3.2. Реализация подключения интерфейса к базе данных

Для подключения базы данных используется технология доступа от Microsoft Entity Framework, которая автоматически связывает обычные классы языка C# с

таблицами в базе данных. Entity Framework 6 нацелен в первую очередь на работу с СУБД MS SQL Server, однако поддерживает также и ряд других СУБД. В данной работе будем использовать SQLite [18].

Для взаимодействия с базой данных через EF используется контекст данных – класс, унаследованный от класса `System.Data.Entity.DbContext`. В проекте был создан контекст данных `Context`:

```
public class Context : DbContext
{
    public Context() : base("SQLiteConnection") { }

    public DbSet<EducationForm> EducationForms { get; set; }
    public DbSet<Employee> Employees { get; set; }
    public DbSet<Faculty> Faculties { get; set; }
    public DbSet<Order> Orders { get; set; }
    public DbSet<PracticeType> PracticeTypes { get; set; }
    public DbSet<Specialty> Specialties { get; set; }
    public DbSet<Student> Students { get; set; }
    public DbSet<User> Users { get; set; }
}
```

Чтобы подключиться к базе данных, в `app.config` прописывается конфигурация для подключения:

```
<connectionStrings>
    <addname="SQLiteConnection"
        connectionString="Data Source=..\..\data\usersdata.sqlite"
        providerName="System.Data.SQLite" />
</connectionStrings>
```

В методе `Main` класса `Program` производится инициализация класса `Context`, экземпляр которого будет доступен на протяжении работы программы, но в конце соединение с базой данных удалится с помощью метода `Dispose`:

```
static class Program
{
    public static string userRole = null;

    [STAThread]
    static void Main()
    {
        Context db = new Context();

        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new LoginForm(db));

        if (userRole != null)
```



```
Application.Run(newForm1(db));  
    db.Dispose();  
}  
}
```

3.3. Руководство пользователя

Окно входа. При запуске приложения открывается окно входа (рис. 3.2), в котором пользователь может войти в систему, указав данные своей учётной записи (логин и пароль) и нажав кнопку «Войти»; либо он может зарегистрироваться, если не имеет учётной записи в приложении, нажав соответствующую кнопку в данном окне.

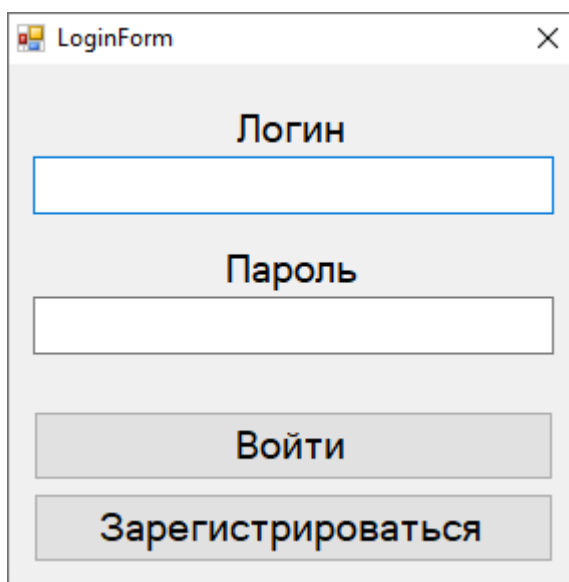


Рисунок 3.2. Окно входа.

Главное окно. После прохождения аутентификации и авторизации пользователя, открывается главное окно приложения (рис 3.3). Из этого окна можно перейти в другие, нажав на определённые кнопки.

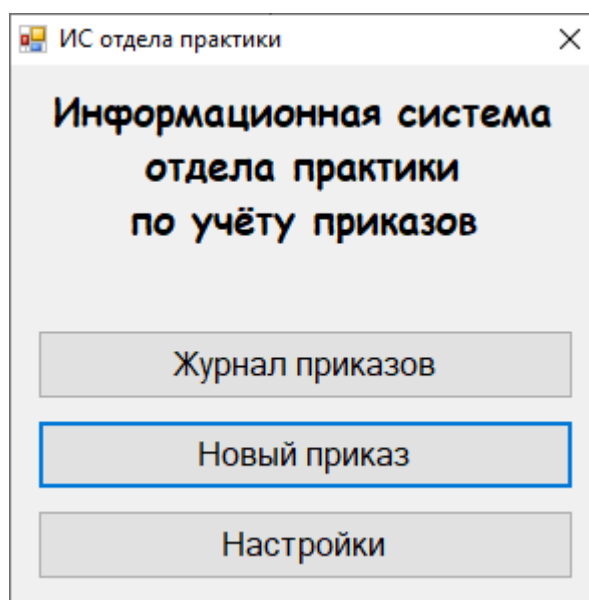


Рисунок 3.3. Главное окно.

Окно журнала приказов. Так, если в главном окне нажать первую кнопку, то откроется окно журнала приказов (рис 3.4). При нажатии на кнопку «Поиск» отображается журнал приказов, представляющий собой таблицу, содержащую информацию о номере приказа, о дате его формирования, о типе практики, о руководителе практики, о специальности и курсе студентов, которые будут проходить практику, об ответственном за формирование приказа. Также можно выполнить фильтрацию запросов по определенным критериям.

№ приказа	Дата форм. приказа	Тип практики	PracticeManager	Специальность	Курс	Ответственный за формирование приказа	Орен
6	06.06.2023	Учебная	Тазетдинов Б.И.	09.03.03 Прикл...	4	admin	Открыть
7	06.06.2023	Ознакомительн...	Чудинов В.В.	01.03.02 Прикл...	4	vopigaw	Открыть
8	06.06.2023	Производствен...	Пихтовников С.В.	09.03.03 Прикл...	4	admin	Открыть

Тип практики:

Факультет:

Пользователь:

Поиск

Очистить фильтр

Рисунок 3.4. Окно журнала приказов

Окно формирования нового приказа. В главном окне при нажатии на кнопку «Новый приказ» появится окно формирования нового приказа (рис 3.5). В этом окне заполняются все необходимые данные для формирования приказа. В сведениях о практике указываются такие данные, как: группа, которая будет проходить практику, вид практики, даты его начала и окончания, а также ФИО руководителя практики от кафедры. Далее, в данных приказа указываются дата приказа и дата его формирования. Последующим действием становится выбор, будет ли сохранён файл приказа в формате PDF, а также, в каком формате будет открыт этот файл после нажатия на кнопку «Сформировать новый приказ».

OrderCreatingForm

Данные практики

Форма обучения: Очная

Факультет / колледж: Физики и математики

Кафедра: Информатики и экономики

Курс: 4

Направление: 09.03.03 Прикладная информатика

Вид практики: Ознакомительная

Дата начала практики: 12.06.2023

Дата окончания практики: 03.07.2023

Рук. практики от кафедры: Тазетдинов Б.И.

Данные приказа

☒ Сохранить приказ в базу данных

Номер приказа: 19

Дата формирования: 12.06.2023

☒ Сохранить в формате PDF

Открыть:

☐ в формате DOCX

☒ в формате PDF

Сформировать новый приказ

Рисунок 3.5. Окно формирования нового приказа.

Окно настроек. Окно настроек (рис. 3.6) предоставляет работу с базой данных. Через него можно перейти в другие окна, которые позволяют добавлять, читать, изменять и удалять записи. Записи отображаются внутри таблицы, похожие на таблицу в окне журналов (рис. 3.4), фильтрация также имеется.

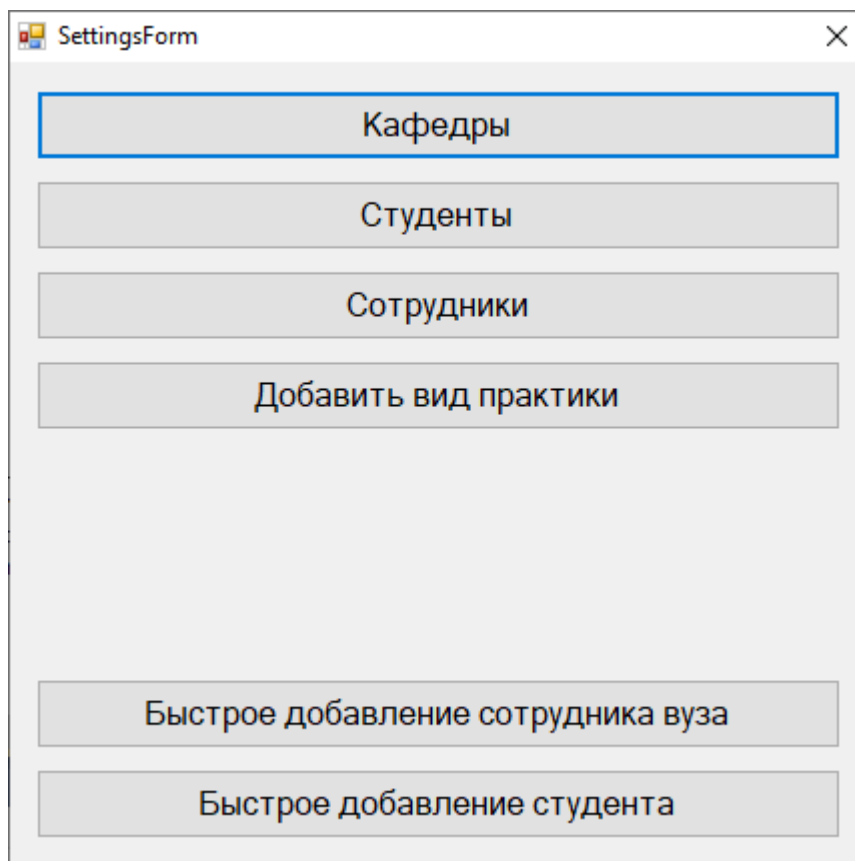


Рисунок 3.6. Окно настроек.

Вывод по главе 3

В третьей главе произведено реализация компонентов информационной системы.

В первой части представлена реализация интерфейса, а также приведён пример одной из форм интерфейса и описаны элементы управления в ней.

Во второй части проведена реализация подключения интерфейса к базе данных с использованием Entity Framework 6 и SQLite.

В третьей части представлено руководство пользователя.

ЗАКЛЮЧЕНИЕ

В данной работе рассматривалась разработка информационной системы отдела практики по учёту приказов.

В первой главе был проведен анализ предметной области, детально изучен объект исследования при помощи диаграмм в нотациях IDEF0, DFD и IDEF3.

Во второй главе описываются основные элементы технического задания, продемонстрирована диаграмма прецедентов и описаны модели представления баз данных.

В третьей главе описана разработка интерфейса информационной системы.

Итогом работы является реализованная информационная система отдела практики по учёту приказов. Система позволяет формировать новые приказы, вести их учёт, а также выполнять настройку этой системы и её базы данных.

Я подтверждаю, что настоящая работа написана мною лично, не нарушает интеллектуальные права третьих лиц и не содержит сведения, составляющие государственную тайну.

09.06.2023 Шарф- / Шарипов Фарит
Фаритович

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Метод поддержки принятия решений при разработке информационных систем на основе мультиагентного подхода: [Электронный ресурс] – https://elar.urfu.ru/bitstream/10995/65454/1/978-5-7186-1078-9_2018.pdf (дата обращения: 01.12.2022)
2. Назначение и состав методологии IDEF0 в бизнес моделировании: [Электронный ресурс] – <https://bpmn.pro/process/idef0> (дата обращения: 02.12.2022)
3. DFD методология. Нотация, принципы моделирования: [Электронный ресурс] – <https://goo.su/I2Joh> (дата обращения: 02.12.2022)
4. Методология IDEF3 в бизнес моделировании: [Электронный ресурс] – <https://itteach.ru/bpwin/metodologiya-idef> (дата обращения: 02.12.2022)
5. Язык программирования C: [Электронный ресурс] – <https://metanit.com/c/tutorial/1.1.php> (дата обращения: 02.12.2022)
6. Язык программирования C++: [Электронный ресурс] – <https://metanit.com/cpp/tutorial/1.1.php> (дата обращения: 02.12.2022)
7. Краткий обзор языка C#: [Электронный ресурс] – <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/> (дата обращения: 02.12.2022)
8. Что такое Go: [Электронный ресурс] – <https://metanit.com/go/tutorial/1.1.php> (дата обращения: 02.12.2022)
9. Язык программирования Python: [Электронный ресурс] – <https://metanit.com/python/tutorial/1.1.php> (дата обращения: 02.12.2022)
10. Язык программирования Java: [Электронный ресурс] – <https://metanit.com/java/tutorial/1.1.php> (дата обращения: 02.12.2022)
11. Языки программирования в информационных системах: [Электронный ресурс] – <https://inlnk.ru/RjNGnv> (дата обращения: 03.12.2022)
12. Visual Studio | Visual Studio Community Edition: [Электронный ресурс] – <https://visualstudio.microsoft.com/ru/vs/community/> (дата обращения: 03.12.2022)

13. Visual Studio Code – Википедия: [Электронный ресурс] – https://ru.wikipedia.org/wiki/Visual_Studio_Code (дата обращения: 03.12.2022)
14. Visual Studio Code: что это за редактор и для чего он нужен: [Электронный ресурс] – <https://blog.skillfactory.ru/glossary/visual-studio-code/> (дата обращения: 03.12.2022)
15. Rider: кросс-платформенная IDE для .NET - JetBrains: [Электронный ресурс] – <https://www.jetbrains.com/ru-ru/rider/> (дата обращения: 03.12.2022)
16. VisualStudio | WindowsForms: [Электронный ресурс] – <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-6.0> (дата обращения: 03.12.2022)
17. Windows Presentation Foundation : [Электронный ресурс] – https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation (дата обращения: 03.12.2022)
18. SQLite, типизация, надежность, практика: [Электронный ресурс] – <https://lecturesdb.readthedocs.io/databases/sqlite.html> (дата обращения: 04.12.2022)
19. SQLite: [Электронный ресурс] – <https://metanit.com/sql/sqlite/1.1.php> (дата обращения: 04.12.2022)
20. Инфологическое проектирование: [Электронный ресурс] – <https://ppt-online.org/636338> (дата обращения: 05.12.2022)
21. Даталогическое проектирование: [Электронный ресурс] – https://studopedia.ru/4_114437_datalogicheskoe-proektirovanie.html (дата обращения: 05.12.2022)
22. Entity Framework 6 | Развитие: [Электронный ресурс] – https://professorweb.ru/my/entity-framework/6/level1/1_1.php (дата обращения: 05.12.2022)
23. Обзор ORM для C#: что подойдет для проекта: [Электронный ресурс] – <https://habr.com/ru/company/simbirsoft/blog/659841/> (дата обращения: 06.12.2022)
24. Worker-ы и sharedworker-ы: [Электронный ресурс] – <https://habr.com/ru/post/261307/> (дата обращения: 06.12.2022)

25. ViewModel и LiveData: паттерны и антипаттерны: [Электронный ресурс] – <https://habr.com/ru/post/338590/> (дата обращения: 06.12.2022)

26. Команды в MVVM: [Электронный ресурс] – <https://metanit.com/sharp/wpf/22.3.php> (дата обращения: 06.12.2022)

27. Руководство по классическим приложениям (Windows Forms .NET): [Электронный ресурс] – <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-6.0> (дата обращения: 29.05.2023)

28. Репин В.В., Елиферов В.Г. Процессный подход к управлению. Моделирование бизнес-процессов. – М.: Манн, Иванов и Фербер, 2013. – 544 с.

29. Силич В.А., Силич М.П. Моделирование и анализ бизнес-процессов: учебное пособие. – Томск: Издательство Томск. гос. ун.-та систем управления и радиоэлектроники, 2011. – 212 с.

30. Работа с данными в EntityFramework: [Электронный ресурс] – <https://metanit.com/sharp/aspnet5/12.1.php> (дата обращения: 29.05.2023)

31. Карьера в IT. Системный аналитик, часть 3, диаграммы. UML: [Электронный ресурс] - https://pikabu.ru/story/karera_v_it_sistemnyiy_analitik_chast_3_diagrammyi_uml_9980766 (дата обращения: 29.05.2023)

Марф - / Маринков П. П.