# АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ СИСТЕМЫ УЧЕТА НАГРУЗКИ ПРЕПОДАВАТЕЛЕЙ В ШКОЛЕ ИНОСТРАННЫХ ЯЗЫКОВ

# 1.1 Описание предметной области

Программа учёта нагрузки преподавателей в школе иностранных языков предназначена для мониторинга и анализа рабочего времени преподавательского состава. Это важный инструмент для оптимизации учебного процесса и распределения рабочих часов среди преподавателей. Каждый преподаватель должен иметь чётко определённое количество часов, которое соответствует его квалификации и учебной нагрузке.

Организационная структура программы включает в себя список должностных лиц и преподавателей, связанных между собой, чтобы обеспечить эффективное управление учебным процессом. Директор школы является ключевой фигурой, так как он отвечает за общее руководство и координацию работы школы. В его обязанности входит управление, сортировка и отбор преподавательского состава.

Существует список параметров, которые влияют на функционирование и организационную структуру программы учёта нагрузки. Среди них количество преподавателей, их квалификация, количество учебных часов и расписание занятий.

Методист школы играет важную роль в организационной структуре, так как он отвечает за разработку учебных программ и контроль за их исполнением. Он также участвует в организации методических мероприятий и повышении квалификации преподавателей.

Внимательность и ответственность являются критериями эффективной работы директора школы, так как эти качества помогают избежать ошибок в управлении учебным процессом. Необходимо также упомянуть административный персонал, который поддерживает работу школы, помогая преподавателям и учащимся в организационных вопросах.

Другие работники школы, такие как секретари, бухгалтеры, уборщики и технический персонал, играют важную роль в поддержании порядка, чистоты и комфорта в учебном заведении. Они обеспечивают благоприятную атмосферу для обучения и работы преподавателей и учащихся.

# 1.2 Разработка функциональной модели предметной области в нотации IDEF0

Для глубокого понимания процесса учёта нагрузки преподавателей, важно осознать, что нотация IDEF0 представляет собой стандарт описания, включающий в себя графические элементы и правила моделирования, применяемые для визуализации и анализа рабочих процессов. Эта нотация служит графическим языком для демонстрации работы школы и взаимодействия между её отделами и сотрудниками. Нотация IDEF0 может использоваться как для процессного, так и для функционального моделирования, что делает её эффективным инструментом в анализе бизнес-процессов.

IDEF0, разработанная для автоматизации производственных процессов, позволяет формализовать и описывать процессы учёта нагрузки, фокусируясь на логических связях между задачами. Функциональная модель IDEF0 состоит из блоков, которые детализируются до необходимого уровня и соединяются стрелками, указывающими на взаимосвязи и потоки данных. Важно, что каждый блок и стрелка имеют чётко определённое значение, отражающее входные данные, результаты, управление и механизмы процесса.

Для наглядности и удобства анализа была создана функциональная модель учёта нагрузки в нотации IDEF0 с использованием программного средства BPwin (см. рисунок 1.1). Это позволяет детально рассмотреть каждый аспект процесса и обеспечить его оптимизацию.

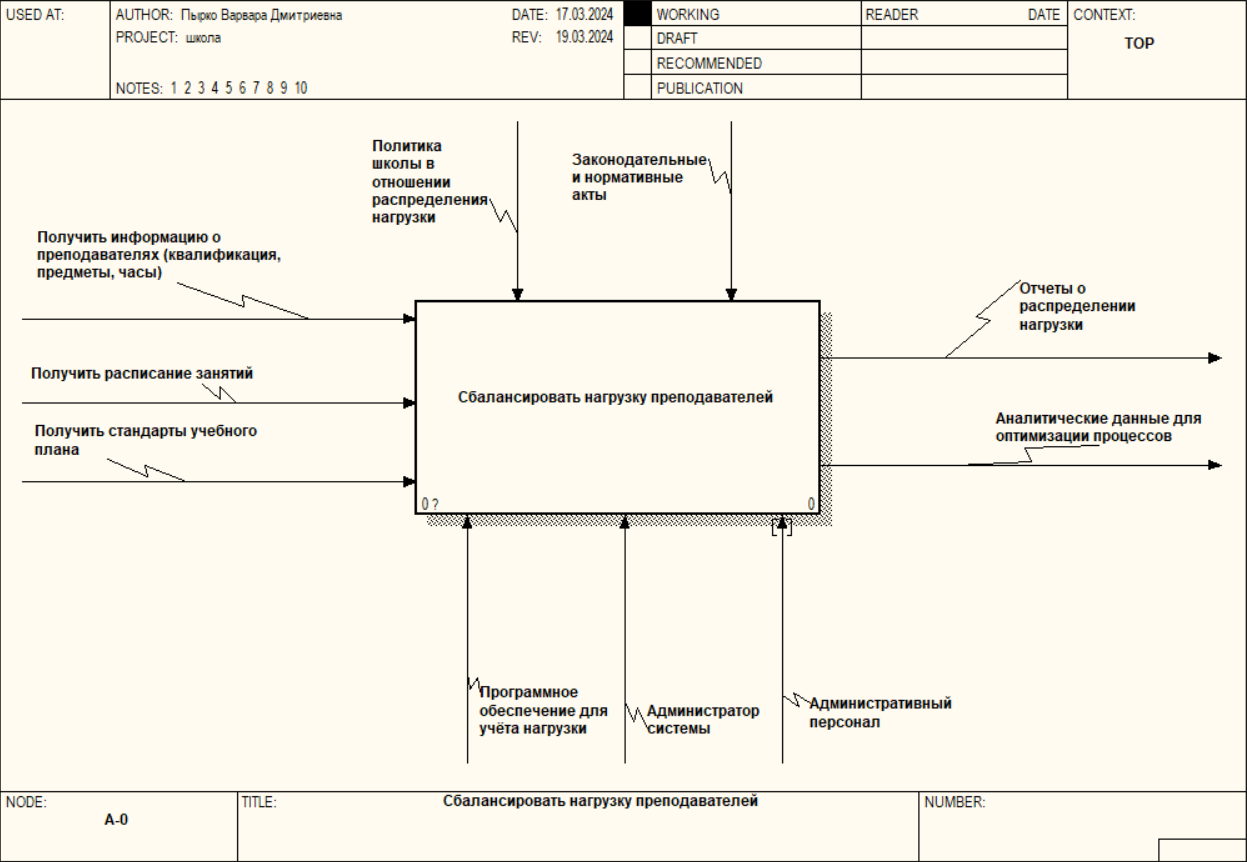


Рисунок 1.1 – Контекстная диаграмма первого уровня

Декомпозиция модели позволяет выявить ключевые функции и задачи, связанные с учётом нагрузки преподавателей, и обеспечить их эффективное выполнение. Такой подход способствует лучшему пониманию и управлению процессами в школе иностранных языков. (см. рисунок 1.2)

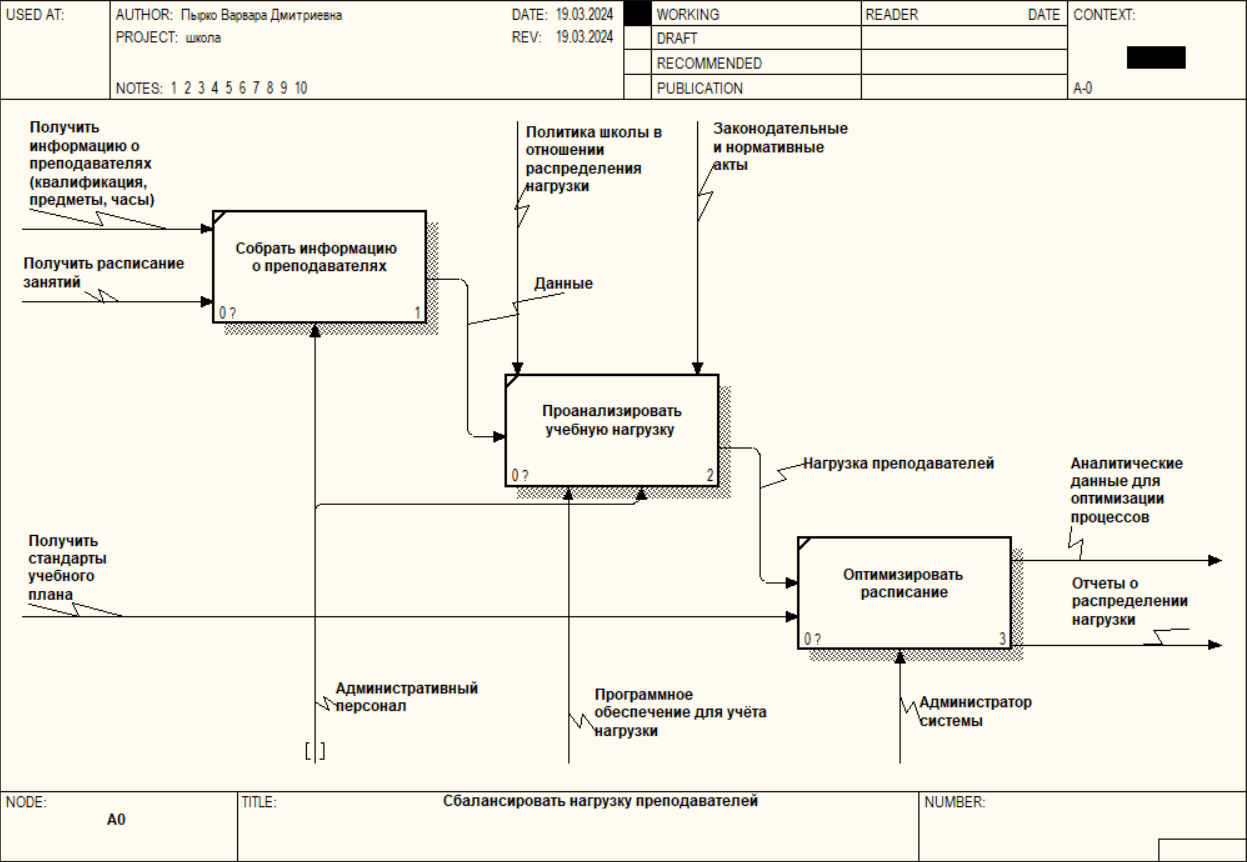


Рисунок 1.2 – Декомпозиция диаграммы

Таким образом, мы визуализировали организацию и задачи нашей системы, используя язык IDEF0.

# 1.3 Разработка BPMN-модели основного процесса предметной

# области

Как и в случае с IDEF0, для детального понимания процесса учёта нагрузки преподавателей важно ознакомиться с нотацией BPMN. Эта нотация является стандартом для визуализации бизнес-процессов и позволяет описывать последовательность действий, потоки данных и сообщений между участниками процесса. BPMN идеально подходит для моделирования поведенческих аспектов процесса, но не охватывает такие элементы, как организационная структура или бизнес-стратегия.

BPMN поможет визуализировать процесс учёта нагрузки, начиная с получения данных о преподавателях и заканчивая формированием отчётов. На рисунке 1.3 будет представлена BPMN-диаграмма, иллюстрирующая этот процесс.

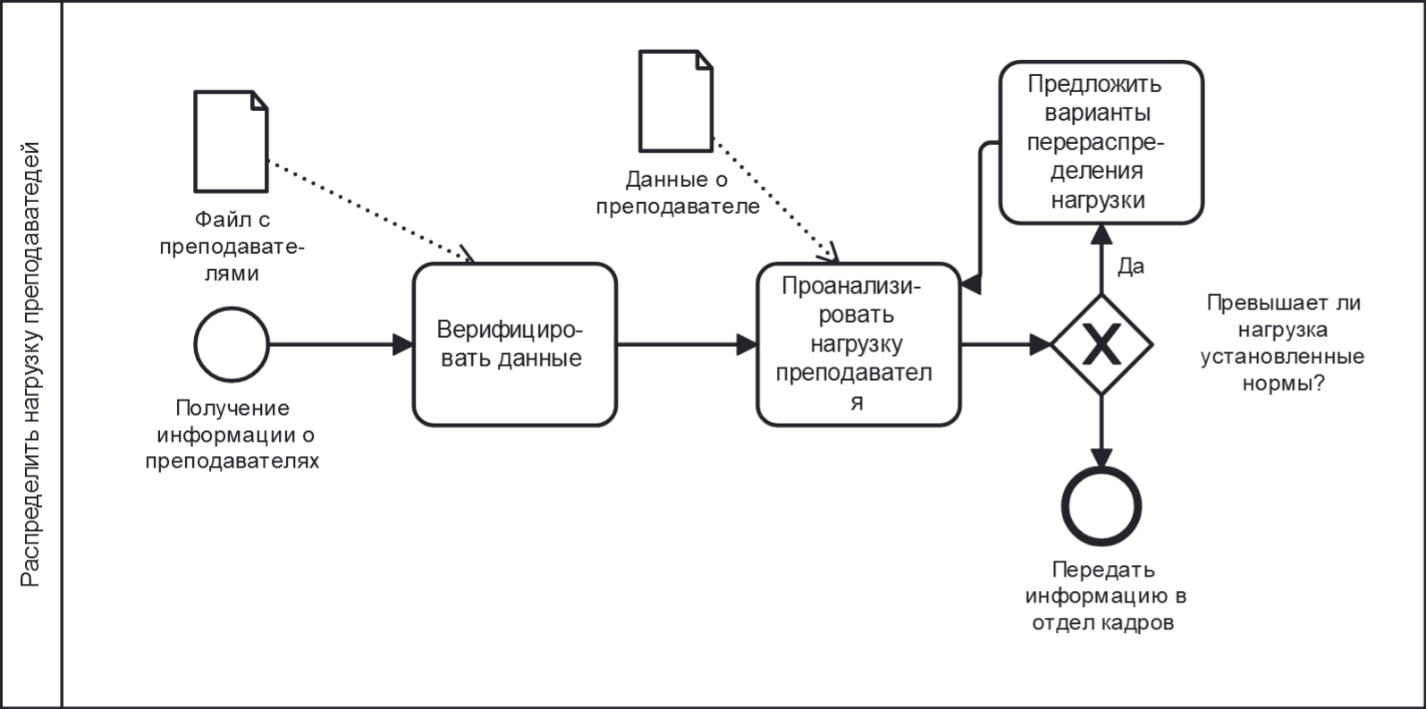


Рисунок 1.3 – Диаграмма в нотации BPMN

При детальном рассмотрении диаграммы можно увидеть, что первым шагом является получение информации о преподавателях и их расписании. Затем следует процесс верификации данных, включая квалификацию и количество часов. После верификации система анализирует нагрузку каждого преподавателя. Если нагрузка превышает установленные нормы, система предлагает варианты перераспределения часов.

В случае, если нагрузка находится в пределах нормы, информация передаётся в отдел кадров для подтверждения. После этого формируются отчёты, которые могут быть использованы для оптимизации учебного процесса. Только те преподаватели, чья нагрузка соответствует стандартам, получают окончательное расписание.

# 1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований

Для регистрации рабочей нагрузки преподавателей необходимо составить список, содержащий: полное имя преподавателя, предмет, количество часов, квалификацию и стаж работы. Приоритет распределения нагрузки отдается преподавателям с наибольшим стажем и квалификацией.

Индивидуальная задача программы заключается в выводе списка преподавателей с учётом их нагрузки и квалификации, при этом пользователь может задать параметры для сортировки и фильтрации списка.

Программа также должна включать систему авторизации с разграничением доступа для администратора и пользователя.

Исходные данные для выполнения курсовой работы:

⎯ язык программирования: C++;

⎯ среда разработки: Microsoft Visual Studio;

⎯ тип приложения: консольное;

⎯ парадигма программирования: объектно-ориентированная;

⎯ способ организации данных: классы (class);

⎯ способ хранения данных: файлы;

⎯ каждая логически завершенная задача программы должна быть реализована в виде метода или функции;

⎯ код программы должен соответствовать правилам, установленным в документе "С++ Code Convention";

⎯ текст пояснительной записки должен быть оформлен в соответствии со стандартом предприятия СТП 01-2017.

Функциональные требования к курсовой работе:

⎯ Авторизация пользователя для доступа к функциям программы, основанная на данных из файла с учетными записями.

⎯ Возможность входа в систему как администратор (роль = 1) или обычный пользователь (роль = 0).

⎯ Автоматическое создание файла учетных записей при его отсутствии.

⎯ Работа с данными после авторизации, хранение информации в файле.

Модуль администратора включает следующие подмодули:

1 Управление учетными записями: просмотр, добавление, редактирование, удаление.

2 Работа с файлом данных: создание, открытие, удаление.

3 Режим редактирования данных: просмотр, добавление, удаление, редактирование записей.

4 Режим обработки данных: выполнение задач, поиск.

Модуль пользователя включает подмодуль работы с данными со следующими возможностями:

⎯ просмотр всех данных;

⎯ выполнение индивидуального задания;

⎯ поиск данных.

Для реализации этих модулей/подмодулей необходимо создать меню с соответствующими пунктами.

Дополнительные функциональные требования:

1 Обработка исключительных ситуаций:

⎯ Проверка соответствия введенных данных формату поля;

⎯ Валидация корректности введенных данных;

⎯ Обработка отсутствия файла с данными для чтения;

⎯ Уведомление пользователя при отсутствии результатов поиска;

⎯ Проверка допустимости номера удаляемой записи;

⎯ Проверка уникальности логина при создании новой учетной записи.

1. Навигация:

⎯ Возможность возврата к предыдущим шагам программы.

1. Подтверждение необратимых действий:

⎯ Запрос подтверждения перед удалением файлов или записей.

1. Обратная связь:

⎯ Информирование пользователя об успешности выполнения операций.

Требования к программной реализации:

1 Именование:

⎯ Переменные и константы должны иметь осмысленные имена, отражающие их функцию в контексте программы;

⎯ Функции должны называться по принципу действие + объект (глагол + существительное).

2 Структура кода:

⎯ Использование оператора goto запрещено.

⎯ Избегание неименованных числовых и строковых констант; такие значения должны быть определены как именованные константы с атрибутом const.

⎯ Комментирование кода для повышения его читаемости и понимания;

⎯ Избегание дублирования кода; повторяющиеся операции должны быть вынесены в функции;

⎯ Каждая функция должна выполнять одну задачу.

3 Оптимизация работы с файлами:

⎯ Минимизация операций чтения/записи в файл для повышения производительности.

Таким образом, мы выявили основные требования к программной реализации и приняли их к сведению.

# 1.5 UML-модели представления программного средства и их описание

UML (Unified Modeling Language) — это универсальный язык моделирования, используемый для визуализации, спецификации, конструирования и документирования артефактов системы программного обеспечения. Как открытый стандарт, UML предоставляет набор графических нотаций, которые помогают создать абстрактное представление системы, известное как UML-модель. Этот язык не предназначен для программирования, но позволяет генерировать код на его основе.

Нотации UML являются ключевыми элементами для создания моделей, которые эффективно передают структуру и поведение системы. Неправильное использование нотаций может привести к созданию неэффективных и запутанных моделей.

В UML различают структурные, поведенческие, группирующие и аннотационные элементы, каждый из которых играет свою роль в моделировании системы. Структурные элементы, такие как классы, объекты, интерфейсы и компоненты, отражают статическую сторону модели и могут представлять как концептуальные, так и физические элементы.

Классы в UML используются для описания объектов с определенными свойствами и методами. Диаграмма класса обычно делится на три части: имя класса, атрибуты и операции. Объекты являются экземплярами классов и отображаются с подчеркнутым именем. Интерфейсы определяют набор операций без их реализации и изображаются в виде круга с названием. Сотрудничество указывает на взаимодействие между элементами для выполнения задачи.

Случаи использования описывают, как пользователи (актеры) взаимодействуют с системой, и представлены в виде овалов, связанных с актерами. Актеры могут быть как внутренними, так и внешними участниками процесса. Начальное и конечное состояния указывают на начало и конец процесса в системе. Компоненты представляют части системы и изображаются в виде прямоугольников с названием и ответственностью.

Активные классы отображают параллельные процессы и делятся на три части: имя класса, требования и операции. Узлы представляют физические компоненты системы, такие как серверы или оборудование, и изображаются в виде кубов с названием и ответственностью.

Переходя от теории к практике, мы создаем диаграмму вариантов использования, которая отражает взаимодействие пользователей с системой учёта нагрузки преподавателей, в соответствии с требованиями, описанными в разделе 1.3 (см. рисунок 1.4).

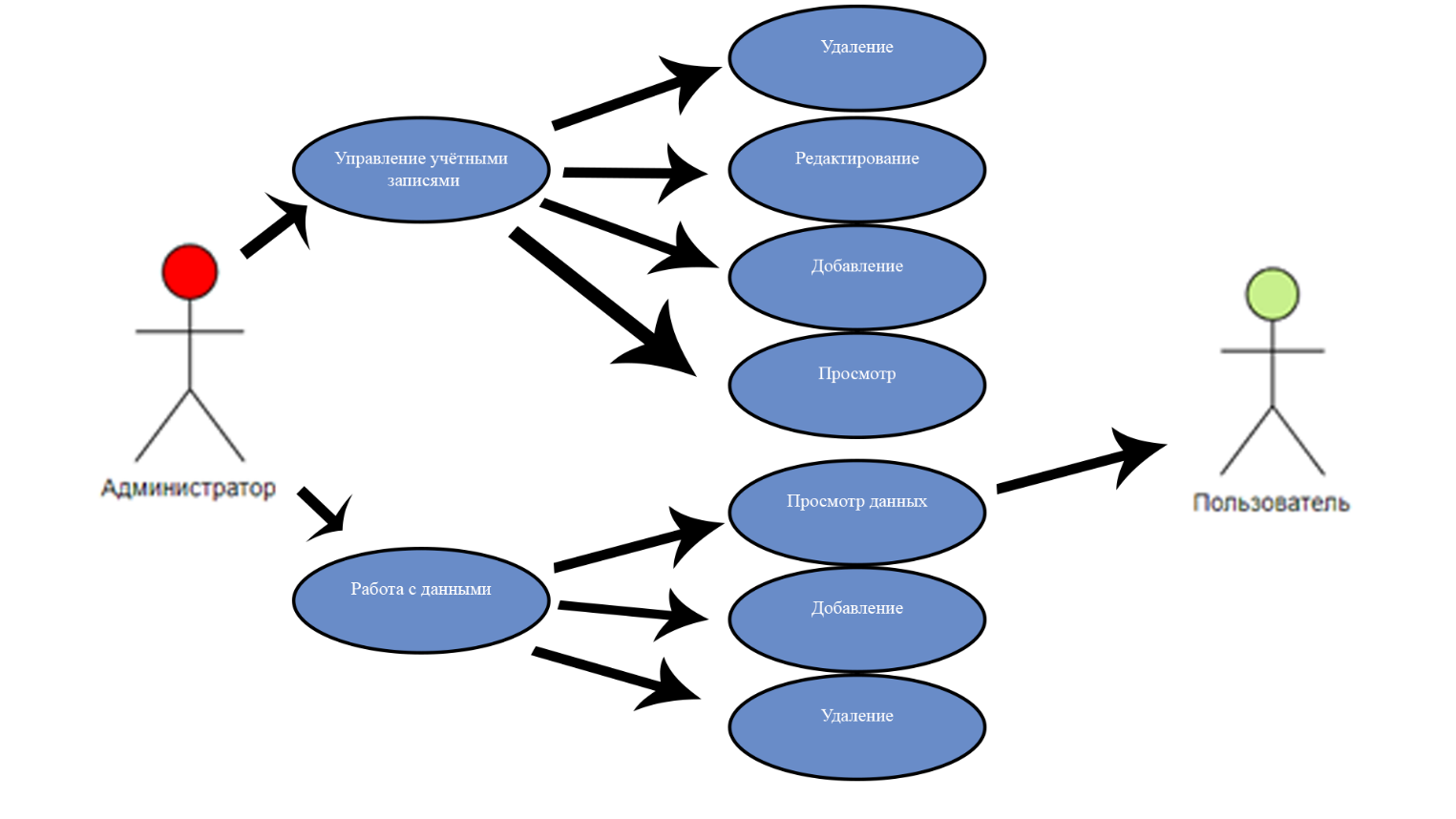


Рисунок 1.4 – Диаграмма вариантов использования

Диаграмма состояний в UML отображает различные состояния объекта в течение его жизненного цикла. На рисунке 1.5 представлена диаграмма состояний для процесса учёта нагрузки преподавателей. Она начинается с анализа доступных часов преподавания, последовательно переводит преподавателя через различные этапы проверки квалификации и стажа работы, и в зависимости от результатов, распределяет учебные часы.

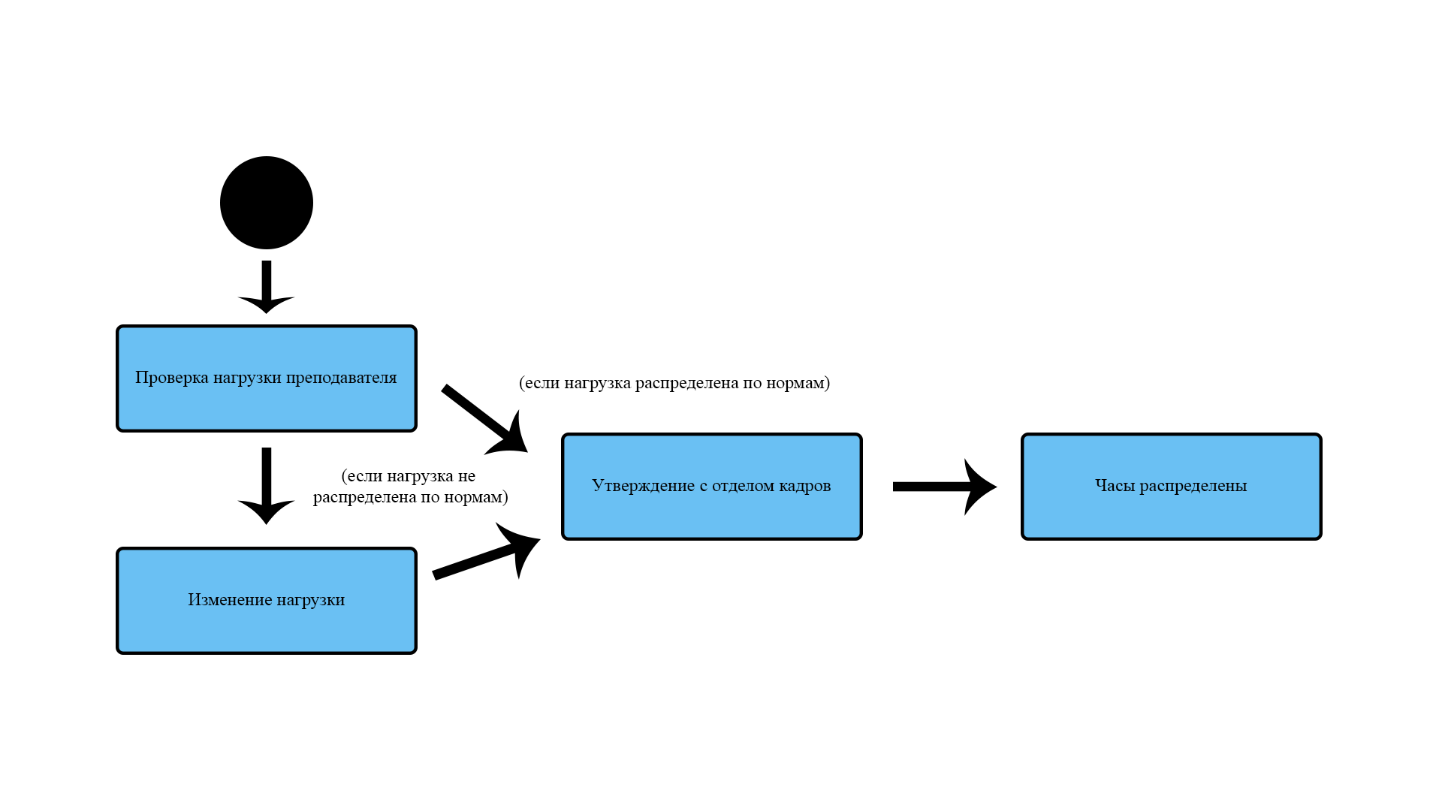


Рисунок 1.5 – Диаграмма состояний

Процесс регистрации новых пользователей начинается с того, что Администратор вводит личные данные пользователя, которые затем проходят валидацию. После этого данные пользователя сохраняются в файл и создаётся учётная запись пользователя. Этот процесс представлен на рисунке 1.6.

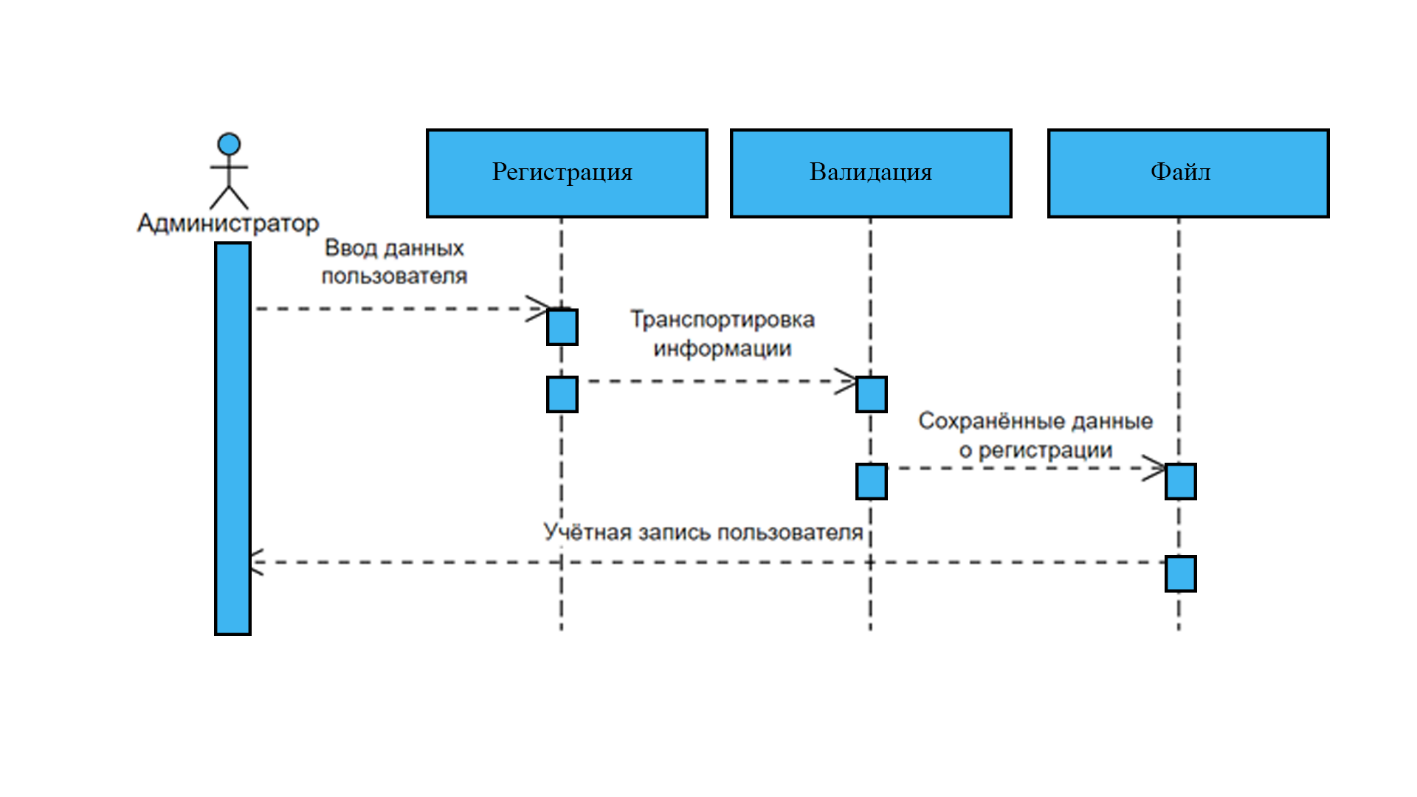


Рисунок 1.6 – Диаграмма последовательности состояния

Диаграмма развертывания показывает физическую архитектуру системы, включая аппаратные и программные среды исполнения. Она важна для понимания, как программное средство будет развернуто и функционировать на аппаратном обеспечении. На рисунке 1.7 представлена диаграмма развертывания, иллюстрирующая аппаратную топологию системы учёта нагрузки преподавателей.

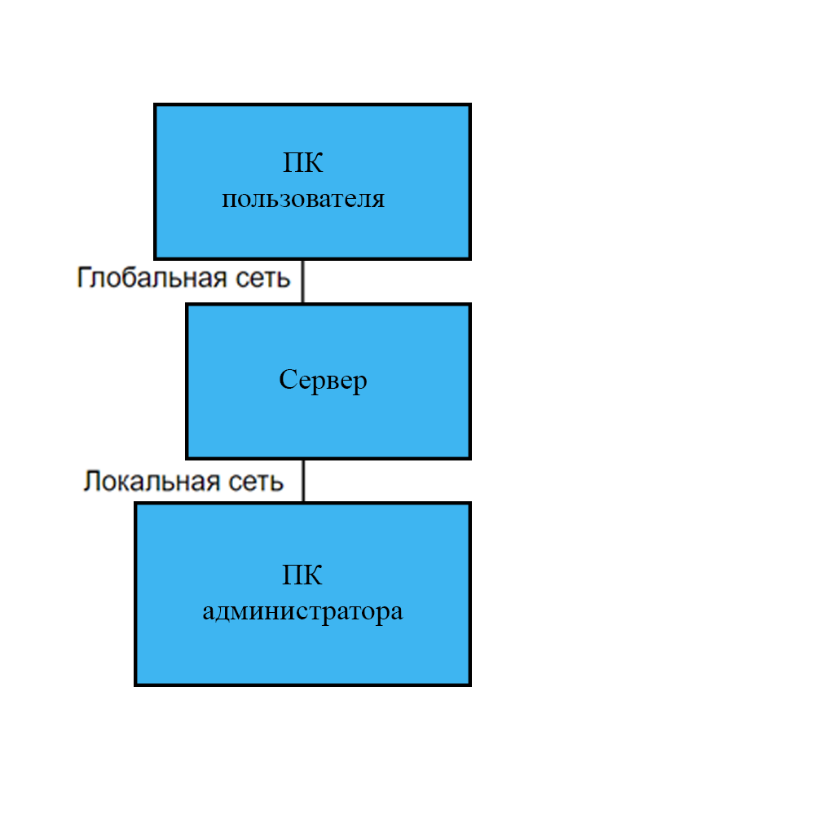


Рисунок 1.7 – Диаграмма развёртывания

Таким образом, с помощью UML мы визуализировали ключевые процессы программного средства, что позволяет лучше понять его структуру и поведение.