РЕФЕРАТ

Выпускная квалификационная работа: 59 страниц, 19 рисунков, 5 таблиц, 20 источников.

Ключевые слова: ГИС СЦОС, интеграция с ГИС СЦОС, .NET, ASP.NET, .Entity Framework, Code First, Oracle.

Объектом исследования являлся Рубцовский институт (филиал) АлтГУ.

Предметом исследования являлся процесс интеграции корпоративной информационной системы института с ГИС СЦОС.

Целью выпускной квалификационной работы являлась разработка модуля интеграции с Государственной информационной системой «Современная цифровая образовательная среда» (на примере Рубцовского института (филиала) АлтГУ).

Для достижения поставленной цели были решены следующие задачи:

– проведён анализ предметной области;

– разработана архитектура и выработаны проектные решения по обеспечивающим подсистемам;

– реализованы проектные решения разрабатываемой модуля интеграции с ГИС СЦОС;

– выполнена оценка эффективности внедрения модуля интеграции с ГИС СЦОС.

Исходными данными для выполнения работы является учебная и научная литература, а также интернет-источники по проектированию и разработке информационных систем, локальная нормативно-справочная документация Рубцовского института (филиала) АлтГУ.

Для выполнения работы использовались следующие методы и средства: описание систем с использованием графических нотаций, оригинальное проектирование, прототипирование и разработка веб-приложений с использованием .NET, ASP.NET, C#, Oracle, Entity Framework.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современная образовательная система России активно трансформируется в рамках цифровой повестки, где ключевым инструментом контроля и управления становится Государственная информационная система «Современная цифровая образовательная среда» (ГИС СЦОС). Согласно Федеральному закону № 273-ФЗ «Об образовании в Российской Федерации» и приказам Минобрнауки, все вузы обязаны в установленные сроки предоставлять в ГИС СЦОС актуальные данные об учебных процессах, успеваемости студентов, кадровом составе и материально-технической базе. Это требование направлено на повышение прозрачности, качества образования и формирование единого образовательного пространства.

Однако для многих вузов, особенно использующих собственные уникальные информационные системы, интеграция с ГИС СЦОС становится серьёзным вызовом. Отсутствие автоматизированных решений вынуждает сотрудников вручную консолидировать данные из разрозненных источников, преобразовывать их в требуемые форматы и отправлять через API – процесс, сопряжённый с высокими трудозатратами, рисками ошибок и нарушением сроков. Несвоевременная или некорректная передача информации может привести к административным санкциям, снижению рейтинга вуза и потере доверия со стороны регуляторов.

Таким образом, разработка специализированной системы интеграции внутренних IT-ресурсов вуза с ГИС СЦОС – не просто техническая задача, а стратегическая необходимость. Она позволяет не только соответствовать законодательным требованиям, но и перейти от рутинных операций к управлению данными как стратегическим активом, повышая эффективность работы вуза в условиях цифровой трансформации образования.

В связи с этим, объектом исследования является Рубцовский институт (филиал) АлтГУ.

Предметом исследования является процесс интеграции корпоративной информационной системы института с ГИС СЦОС.

Цель исследования – разработка модуля интеграции с Государственной информационной системой «Современная цифровая образовательная среда».

Для достижения поставленной цели необходимо выполнить следующие задачи:

– провести анализ предметной области;

– разработать архитектуру и выработать проектные решения по обеспечивающим подсистемам;

– реализовать проектные решения проектируемого модуля интеграции с Государственной информационной системой «Современная цифровая образовательная среда»;

– выполнить оценку эффективности внедрения проектируемого модуля интеграции с ГИС СЦОС.

1 Аналитическая часть

1.1 Технико-экономическая характеристика предметной области

Рубцовский институт (филиал) АлтГУ – обособленное структурное подразделение Алтайского Государственного университета, расположенное в городе Рубцовске Алтайского края и осуществляющее постоянно его функции в рамках выданной Филиалу лицензии на образовательную деятельность.

Приказом Государственного комитета РФ по высшему образованию от 22 февраля 1996 г. № 324 г. был создан филиал Алтайского государственного университета в г. Рубцовске.

Приказом Федерального агентства по образованию от 03 марта 2006г. № 116 филиал Алтайского государственного университета в г. Рубцовске был переименован в Рубцовский институт (филиал) государственного образовательного учреждения высшего профессионального образования «Алтайский государственный университет». Приказом Министерства образования и науки Российской Федерации от 28 апреля 2011 г. № 1546.

Предприятие является государственной, не коммерческой организацией, имеет право на ведение образовательной деятельности и на льготы, установленные законодательством РФ, в соответствии с лицензией, выданной Министерством образования РФ [1].

Институт реализует образовательные программы полного и неполного высшего профессионального образования.

Институт обладает правом реализации образовательных программ по хозрасчетным договорам в рамках установленного перечня специальностей.

Институт имеет определенную Уставом, а также Положением о Филиале степень самоуправления, которая необходима Филиалу для эффективного принятия решения в отношении его деятельности.

Руководство образовательным учреждением осуществляют директор, заместитель директора по учебной работе и заместитель директора по научной работе.

Организационная структура института показана на рисунке 1.1.

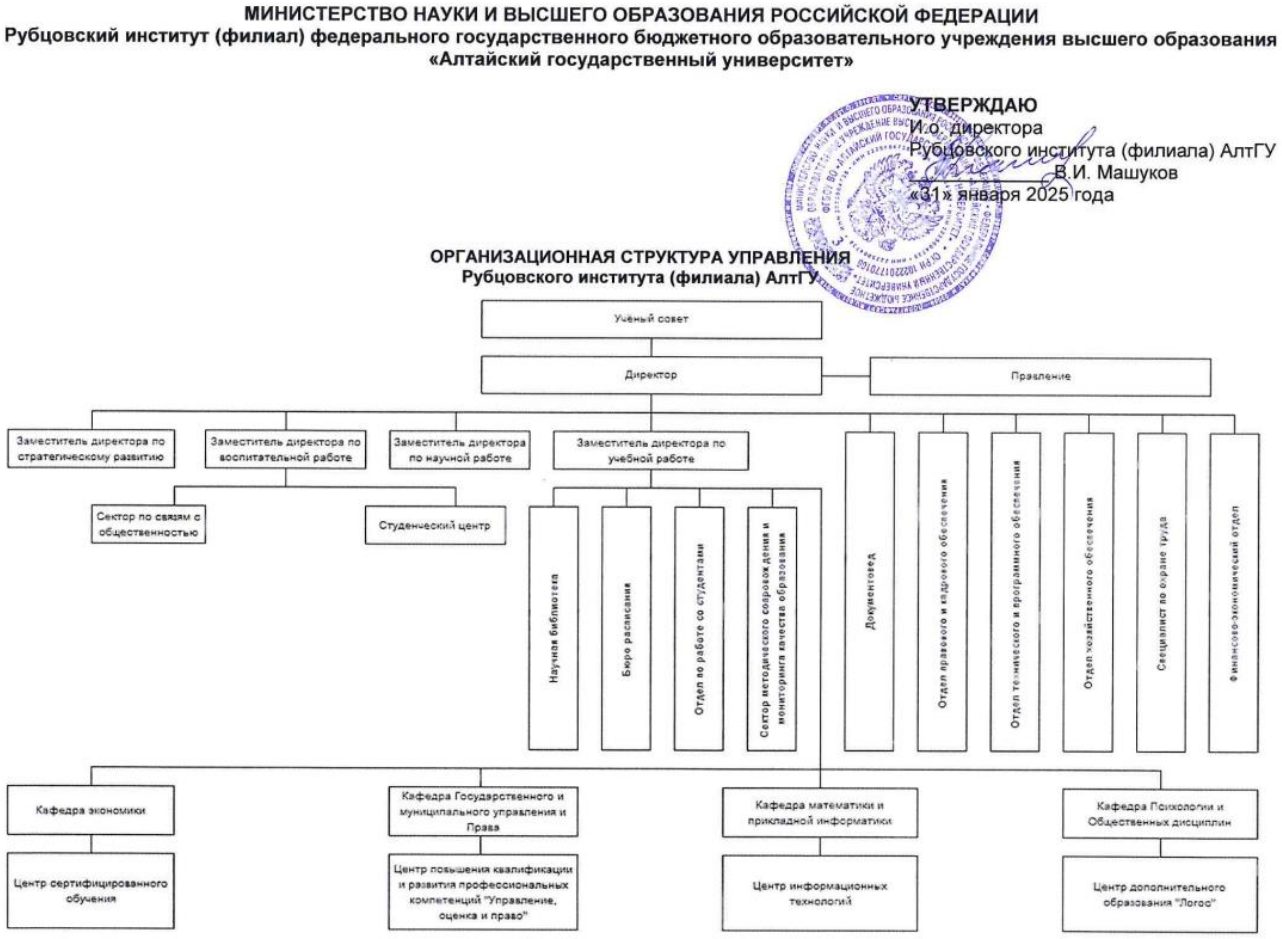

Рисунок 1.1 – Организационная структура управления Рубцовского института (филиала) АлтГУ

Общее количество сотрудников составляет 138 человек, из них 90 человек – это преподавательский состав. В отделе технического и программного обеспечения работает 9 человек.

Институт осуществляет обучение по 7 образовательным программам высшего образования(бакалавриата):

– 44.03.01 Педагогическое образование;

– 40.03.01 Юриспруденция;

– 38.03.04 Государственное и муниципальное управление;

– 38.03.02 Менеджмент;

– 38.03.01 Экономика;

– 09.03.03 Прикладная информатика.

Ещё институт осуществляет обучение по 7 образовательным программам среднего профессионального образования:

– 40.02.04 Юриспруденция;

– 40.02.02 Правоохранительная деятельность;

– 38.02.06 Финансы;

– 38.02.01 Экономика и бухгалтерский учет (по отраслям) (в промышленности, бюджетных отраслях);

– 21.02.05 Земельно-имущественные отношения;

– 09.02.07 Информационные системы и программирование (разработчик веб и мультимедийных приложений);

– 09.02.07 Информационные системы и программирование (специалист по информационным системам).

Также институт реализует программы по переподготовке и оказывает услуги по дополнительному образованию в следующих своих подразделениях:

– Центр повышения квалификации и развития профессиональных компетенций «Управление, оценка и право»;

– Центр дополнительного образования «Логос»;

– Центр информационных технологий;

– Центр сертифицированного обучения [1].

Общая численность студентов, обучающихся по программам среднего и высшего образования составляет 1422 человека.

Институт имеет 2 учебных корпуса по адресам: Алтайский край, г. Рубцовск, пр-кт Ленина, д. 200б; Алтайский край, г. Рубцовск, пр-кт Ленина, д. 243.

В институте имеется 16 компьютерных аудиторий с общим количеством ПК составляющим 250 шт, научная библиотека, большинство лекционных аудиторий оснащено проекционным оборудованием.

Основными источниками финансирования являются: государственное финансирование, платное обучение, дополнительное обучение.

В учебном процессе активно используется система Moodle. Также сопровождение учебного процесса осуществляется при помощи программ, разработанных специально для института, а именно:

– Учебные планы;
– Абитуриенты;
– Студенты;
– График учебного процесса;
– Поручения;
– Редактор расписания;
– Корпоративная информационная система (электронный журнал, РПД и ФОС, ведомости, консультации).

Всего 3 образовательные организации в Рубцовске предоставляют обучение по программам среднего профессионального образования, а также всего 2 организации в городе предоставляют услуги обучения по программам высшего образования. Ежегодно институт принимает около 550 студентов. За всё время своего существования институт выпустил 8975 студентов.

Отдел технического и программного обеспечения – занимается обеспечением работоспособности, ремонтом и наладкой оборудования в Филиале, созданием современного программного обеспечения. Кроме того, сопровождает разработанные и используемые в Филиале информационные системы и базы данных, занимается информатизацией Филиала и реализацией новейших информационных технологий.

Структуру и штаты отдела учреждает директор в соответствии с нормативами численности специалистов и служащих с учетом объема работ и особенностей деятельности отдела.

Задачами отдела является:

- анализ, внедрение и сопровождение новых локальных и глобальных вычислительных сетей;
- определение возможности использования готового ПО;
- обеспечение контроля над правильным использованием оборудования, разработка инструкций и обучение персонала работе со средствами вычислительной техники;
- своевременное обеспечение электронно-вычислительной техники запасными частями;

надзор за техническим состоянием средств охранной сигнализации

1.2 Анализ функционирования объекта исследования

Из-за высокого уровня автоматизации практически все процессы, связанные с учебной деятельностью, выполняются в программах собственной разработки института. Это позволяем проследить все бизнес-процессы через взаимодействие с этими программами [2].

Всё начинается с создания учебного плана. Перед началом приёмной компании в программе «Учебные планы» создаются и заполняются все необходимые учебные планы. Каждый учебный план хранит следующую информацию: уровень образования (бакалавриат, СПО), вид образования (очное, заочное, очно-заочное), продолжительность обучения, общее количество часов и зачётных единиц, циклы дисциплин, дисциплины, учебная нагрузка по различным типам занятий (лекции, лабораторные работ и т.д.) для каждой дисциплины в каждом семестре. После составления в программе можно создать различные отчёты, их список показан на рисунке 1.2.

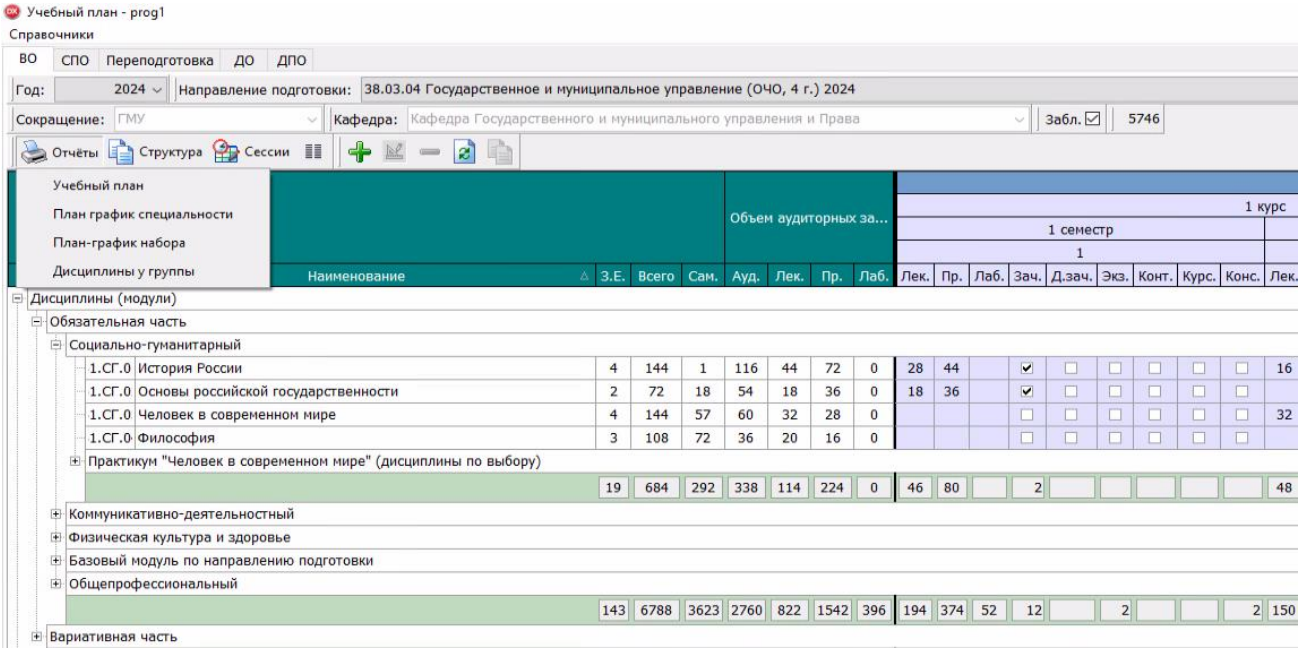

Рисунок 1.2 – Список отчётов в программе «Учебные планы»

После составления учебных планов перед началом приёмной компании в программе абитуриенты заполняется количество мест на каждый учебный план по каждому виду набора (ГН, ДН и т.д.). Во время приёмной компании в программе «Абитуриенты» сотрудники института заполняют информацию об абитуриенте, а также о том на какие учебные планы он претендует поступить. Также в программе абитуриенты можно автоматически создать следующие документы: договор об оказании образовательных услуг, дополнительное соглашение на получение материнского капитала, заявление о согласии на зачисление, опись документов в личном деле, согласие на обработку персональных данных. После окончания приёмной компании сотрудники отмечают студентов, которых необходимо зачислить, и они автоматически зачисляются и появляются уже в программе «Студенты».

В программе «Студенты» содержится вся информация о студенте: личные данные, успеваемость студента, индивидуальный план обучения студента, приказы (в которых фигурирует студент), договоры. После зачисления специалисты учебно-методической работе создают студенческие группы и распределяют по ним студентов, а также именно в этой программе

осуществляется перевод студента на другую специальность, уход в академический отпуск, выход из академического отпуска, отчисление, восстановление, создаются и хранятся договора и дополнительные соглашения со студентом.

В программе «Поручения» распределяется нагрузка на преподавателей. В этой программе можно точечно составить группы студентов для каждого занятия, а также закрепить за каждым занятием преподавателя. В программе можно создать отчёт по индивидуальной нагрузке преподавателя, кафедры.

В программе «График учебного процесса» заполняется периоды, отведённые для обычных занятий, практик, каникул. Также в этой программе можно создать одноименный отчёт. Данный отчёт в печатной форме раздаётся каждому студенту в начале обучения.

В программе «Редактор расписания» выставляются занятия в расписание. Программа берёт данные о группах для занятий из программы «Поручения», а также информацию из программы «График учебного процесса». Программа при каждом выставлении занятия проверяет: есть ли у преподавателя другое занятие в это время, есть ли у студентов занятие в это время, можно ли в этот день выставить занятие (по графику учебного процесса). Информация о расписании доступна в соответствующем разделе на сайте института.

В ИС «Электронный журнал» Корпоративной информационной системы преподаватели выставляют оценки студентам. В подразделе «Ведомости» преподаватели выставляют оценки студентам за контрольные точки. В этом подразделе преподаватели также могут посмотреть pdf файл экзаменационной ведомости.

На рисунке 1.3 представлена DFD диаграмма «AS-IS» информатизации Рубцовского института.

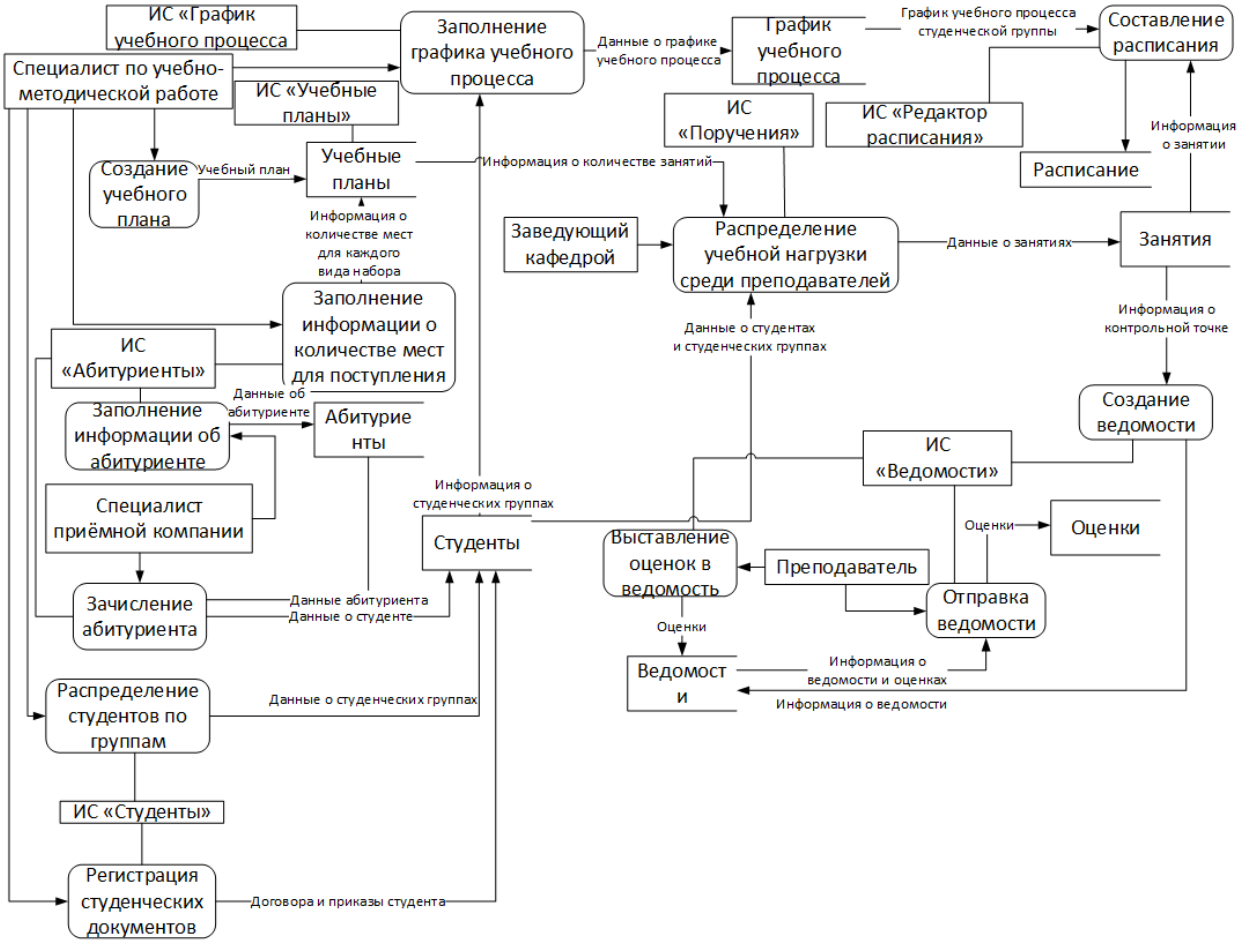

Рисунок 1.3 – DFD диаграмма «AS-IS» Рубцовского института

Из-за того, что информатизация осуществлена при помощи собственных программных разработок института не позволяет использовать «типовые» решения для интеграции своих систем с ГИС СЦОС. Единственный возможный вариант решения – это разрабатывать оригинальную систему для интеграции существующих систем института с ГИС СЦОС [5].

1.3 Определение цели и задач проектирования ИС

Исходя из вышеперечисленного цель исследования – разработка модуля интеграции с Государственной информационной системой «Современная цифровая образовательная среда».

Модуль ИС должен собирать данные из различных баз данных института, преобразовывать их к формату, который принимает ГИС СЦОС, и отправлять на сервера ГИС СЦОС согласно расписанию.

Модуль ИС должен будет собирать, преобразовывать и отправлять данные о следующих объектах:

1) Сведения о студентах.
2) Сведения об образовательных программах.
3) Сведения об учебных планах.
4) Сведения о дисциплинах.
5) Связь учебных планов и дисциплин.
6) Связь учебных планов и студентов.
7) Сведения об оценках.
8) Движение контингента студентов.

Вышеперечисленный перечень данных является обязательным для отправки на сервера ГИС СЦОС, но также есть необязательный блок данных – сведения о сотрудниках ППС, эти данные система не будет отправлять [6].

Модуль ИС позволит институту выполнить положения и цели федерального закона «Об образовании в Российской Федерации» от 29.12.2012 N 273-ФЗ, касающиеся цифровизации образования и создания единой образовательной среды. Постановление правительства Российской Федерации от 16 ноября 2020г. №1836 «О государственной информационной системе «Современная образовательная среда» регламентирует общие положения, участников системы, полномочия участников системы, структуру и порядок доступа к ГИС СЦОС.

Разрабатываемый модуль представляет собой фактографическую диалоговую интеграционную систему, реализованную на базе архитектуры клиент-сервер с использованием веб-технологий (ASP.NET, Swagger UI) [7].

1.4 Обзор и анализ существующих разработок, выбор технологии проектирования

1.4.1 Обзор и анализ существующих разработок

На данный момент существует несколько готовых решений для интеграции с ГИС СЦОС:

1. Технологии автоматизации: Интеграция с ГИС СЦОС. Расширение для 1С:Университет ПРОФ.

Данное расширение предназначено для 1С:Университет ПРОФ. Для передачи данных в ГИС СЦОС к ней подключают информационную систему университета. 1С:Университет ПРОФ интегрируется с ГИС СЦОС через отдельное расширение – «Технологии автоматизации: Интеграция с ГИС СЦОС. Расширение для 1С:Университет ПРОФ».

2. Модуль интеграции с ГИС «Современная цифровая образовательная среда» от лаборатории ММИС.

Модуль предназначен для использования с другими программами от ММИС, а именно: «Планы», «Деканат», «Электронные ведомости».

Применение указанных решений сопряжено с рядом ограничений, ключевым из которых является зависимость от экосистемы разработчиков [3].

Оба продукта функционируют исключительно в связке с проприетарным ПО (1С:Университет ПРОФ и модули ММИС), что делает их несовместимыми с уникальной информационной инфраструктурой института, включающей системы собственной разработки. Таким образом, использование готовых интеграционных решений потребовало бы полной замены существующих информационных платформ, что экономически и технологически нецелесообразно [4].

1.4.2 Выбор технологий проектирования

Решение о выборе ASP.NET в качестве основы для серверной части системы было продиктовано стремлением к унификации разработки. Поскольку институт уже использует экосистему .NET для других проектов, использование C# для новой интеграционной системы позволил сохранить единую языковую среду. Это не только упрощает поддержку и интеграцию с существующими модулями, но и снижает риски, связанные с переходом между разными технологиями. Единый стек технологий гарантирует, что будущие доработки или масштабирование системы будут выполняться быстрее и с меньшими затратами, так как команда разработки сосредоточена на одном инструментарии.

Для работы с данными был выбран Entity Framework с подходом Code-First, что стало логичным продолжением выбора ASP.NET. Этот подход позволил сосредоточиться на проектировании бизнес-логики приложения, а не на ручном управлении структурой базы данных. Code-First дал ключевые преимущества:

– возможность описывать модели данных на C#, что согласуется с основным языком разработки;

– автоматическая генерация и обновление схемы БД через миграции, что критично при частых изменениях требований ГИС СЦОС;

– возможность быстрой развёртки структуры БД (миграции) не только в Oracle, а в любой другой поддерживаемой БД.

Кроме того, Entity Framework обеспечил бесшовную интеграцию с уже используемой в институте СУБД Oracle, что исключило необходимость перевода данных в новую систему.

Swagger UI стал промежуточным решением для взаимодействия с API системы. Его выбор обусловлен двумя факторами. Во-первых, он предоставляет прозрачную документацию для frontend-разработчиков, которые в будущем будут создавать пользовательский интерфейс. Это ускоряет взаимодействие между командами: backend-разработчики могут

сразу демонстрировать готовые методы API, а frontend – тестировать их без дополнительных согласований. Во-вторых, Swagger позволил быстро реализовать минимальный интерфейс для ручного запуска передачи данных, что стало временным решением на этапе внедрения системы. Однако в перспективе планируется разработать отдельное веб-приложение с интуитивно понятным интерфейсом, чтобы даже сотрудники без технических навыков могли работать с системой.

Таким образом, каждая технология была выбрана не только для решения текущих задач, но и с учетом долгосрочных целей: минимизация затрат на поддержку, обеспечение гибкости при изменениях и подготовка основы для будущего развития системы.

1.5 Выбор и обоснование проектных решений

Взаимодействие с ГИС СЦОС возможно исключительно через REST API, что исключает ручную отправку данных без использования специализированных приложений. Это обусловлено следующими факторами:

В отличие от систем, где пользователи работают с визуальным интерфейсом программ, ГИС СЦОС не предоставляет графического интерфейса для отправки данных, визуальный интерфейс существует только для просмотра уже отправленных данных. Вместо этого обмен данных осуществляется через Http-запросы (GET, POST, PUT, DELETE) [8], которые требуют точного формирования структуры данных, заголовков и параметров авторизации. Для рядового пользователя, не обладающего техническими навыками, такой подход непонятен и недоступен.

Даже при использовании инструментов вроде Postman или cURL ручная работа остаётся крайне трудоёмкой и подверженной ошибкам. Каждый запрос должен включать:

1) Аутентификационные данные (токены OAuth 2.0), которые необходимо регулярно обновлять.

2) Корректно составленное тело запроса в формате JSON, соответствующее моделям данных, утверждённым ГИС СЦОС.

3) Правильные HTTP-заголовки.

Малейшая ошибка в синтаксисе, структуре или авторизации приведёт к отклонению запроса сервером.

Также необходимо обрабатывать каждый ответ на запрос. Например, при ошибке 400 Bad Request нужно будет определить какое поле неправильно заполнено в теле запроса, а при ошибке 401 Unauthorized нужно будет обновить токен авторизации [9].

За каждую сессию синхронизации может передаваться тысячи различных объектов, ручная отправка таких объёмов данных через единичные запросы технически невозможна. Кроме того, отправлять нужно каждую неделю.

Также невозможно использовать типовое решение, т.к. институт использует информационные системы собственной разработки, поэтому необходимо разрабатывать собственный модуль для интеграции с ГИС СЦОС.

Входные документы системы – это неформатированные данные, собираемые автоматически из внутренних источников, а выходные включают данные в формате, который принимает ГИС СЦОС, логи операций и ответы API, подтверждающие успешность передачи [10].

Поскольку сбор информации автоматизирован, ручной ввод через экранные формы практически исключен. Исключение составляет интерфейс администратора, реализованный через Swagger UI, где можно вносить корректировки в случае обнаружения ошибок. Для вывода данных предусмотрен веб-интерфейс, отображающий статусы операций передачи (успех, ошибка) и детализацию проблем. Это позволяет сотрудникам оперативно анализировать результаты без необходимости работы с сырыми логами.

Информационная база системы организована в виде централизованной реляционной базы данных Oracle, что обеспечивает единую точку доступа и согласованность данных. Архитектура клиент-сервер позволяет эффективно распределять нагрузку: серверная часть обрабатывает запросы и взаимодействует с API ГИС СЦОС, а клиентские компоненты (например, веб-интерфейс) обеспечивают взаимодействие с пользователями. Данные структурированы вокруг ключевых сущностей: студенты, дисциплины, оценки, учебные планы. Связи между ними поддерживаются через внешние ключи и транзакции, что гарантирует целостность информации. Например, записи об успеваемости автоматически привязываются к студентам и дисциплинам, а маппинг-таблицы обеспечивают соответствие внутренних кодов института стандартам ГИС СЦОС.

Обновление данных происходит автоматически через ETL (Extract, Transform, Load)-процессы, которые извлекают информацию из внутренних систем, преобразуют её и загружают в промежуточное хранилище Oracle. Для ручных корректировок предусмотрены транзакции с валидацией, предотвращающие внесение некорректных значений. Изменения в структуре базы данных (например, добавление новых полей) выполняются через миграции Entity Framework, что исключает простои и потерю информации [10].

Защита данных реализована на нескольких уровнях. Целостность обеспечивается транзакциями и ежедневным резервным копированием [11]. Конфиденциальность поддерживается шифрованием данных на уровне СУБД и шифрованием данных при отправке http-запросов [12].

Таким образом, информационное обеспечение системы сочетает стандартизацию с гибкостью. Это создает надежную основу для выполнения требований регуляторов и дальнейшего развития системы в условиях меняющихся стандартов.

Проект будет развёрнут на специально купленном для работы с ГИС СЦОС сервере.

Для соблюдения требований по безопасности данных на этом сервере помимо ИС для интеграции также установлен Континент TLS-клиент, а также антивирус DrWeb. Все запросы передаются в зашифрованном виде, после установления защищённого соединения с TLS-сервером [13].

2 Проектная часть

2.1 Разработка функционального обеспечения

Для достижения поставленной цели была разработана архитектура ИС, которая включает следующие компоненты:

1. Компонент сбора данных из БД института. Этот компонент обеспечивает регулярный сбора и первичную обработку информации из корпоративных баз данных. Компонент должен каждый день по расписанию собирать информацию об учебной процессе.

2. Компонент преобразования моделей данных. Этот компонент преобразует данные, полученные из БД института преобразовать в вид, который принимает ГИС СЦОС. Этот компонент важен, т.к. для каждого объекта (например, «Дисциплина», «Образовательная программа», «Студент») в ГИС СЦОС предусмотрено по 3 модели данных: для обновления (PATCH), вставки (POST) и чтения (GET). ГИС СЦОС принимает различные модели для каждого типа Http-методов.

3. Компонент синхронизации с ГИС СЦОС. Этот компонент связывается с ГИС СЦОС и получает актуальные данные с серверов ГИС СЦОС. После этого данные сверяются с теми, которые содержатся в корпоративных системах института. На основании сравнения данных из ГИС СЦОС и данных из корпоративных систем института компонент определят какие данные необходимо вставить, изменить или удалить.

4. Компонент хранения пакетов данных. Этот компонент представляет собой базу данных, которая хранит в себе подготовленные данные до отправки в ГИС СЦОС. Благодаря этому можно посмотреть какие данные отправятся в ГИС СЦОС, а также это позволяет хранить историю о том какие данные уже были отправлены на сервера ГИС СЦОС.

5. Компонент отправки данных. Этот компонент отправляет данные на сервера ГИС СЦОС. Для успешной отправки данных этот компонент получает JWT-токен и использует его в заголовке авторизации, а также разбивает данные на отдельные запросы.

DFD диаграмма «TO-BE» информатизации института показана на рисунке 2.1.

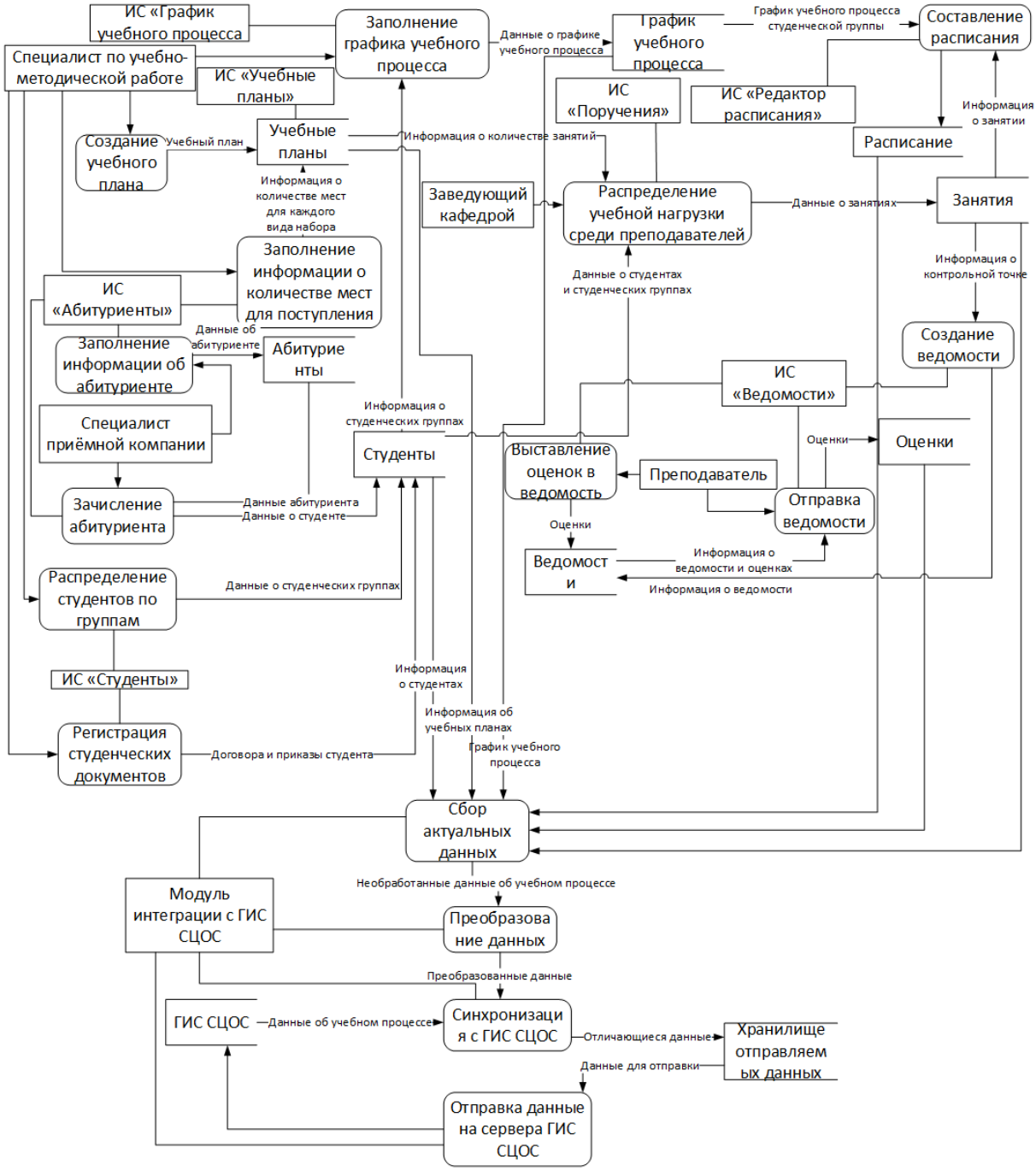

Рисунок 2.1 – Диаграмма «TO BE» информатизации Рубцовского института после внедрения модуля для интеграции с ГИС СЦОС

2.2 Разработка информационного обеспечения

2.2.1 Используемые классификаторы и системы кодирования

Для обеспечения совместимости и корректной работы с ГИС СЦОС, проектируемая система использует строго регламентированные системы кодирования и классификаторы, утверждённые оператором целевой системы. Эти классификаторы перечислены в документе «Описание программных интерфейсов государственной информационной системы «Современная цифровая образовательная среда», размещенной в информационно-телекоммуникационной сети «Интернет» по адресу online.edu.ru» версии 2.8.5.

В этом документе приведено несколько важных соглашений:

1. О именовании – В наименованиях ресурсов и параметров рекомендуется избегать имён, состоящих из нескольких слов. Если использования составных имён не избежать, для разделения слов следует использовать Snake Case. Пример: course_name, registration_number. Следует придерживаться следующих именований часто используемых параметров: a) user_id – идентификатор пользователя в СЦОС b) esia_id – идентификатор пользователя в ЕСИА.

2. О формате передачи данных – Принимать и передавать текстовые и числовые данные следует в формате JSON. Формат передачи файла определяется его типом: − Изображения (файлы .png, .jpeg/.jpg, .gif), а также файлы, размер которых не превышает 1 МБ, следует передавать в формате base64. − Остальные файлы следует передавать в виде multipart/form-data запроса. − Если логика сервиса допускает значительную вариативность размеров передаваемых файлов, формат передачи следует определять исходя из максимально допустимого этой логикой размера. То есть, если, к примеру, предполагается передавать файлы от 100 КБ до 10 МБ, то в таком сервисе следует использовать запросы multipart/formdata для всех файлов, независимо

от их размера. Для передачи дат без значимого времени следует использовать следующий формат: yyyyMM-dd. Для передачи даты с указанием точного времени следует использовать следующий формат: yyyy-MM-ddTHH:mm:ssZ. При передаче вещественных чисел следует в качестве разделителя следует использовать точку: 3.14156926. Также допускается использовать экспоненциальную форму представления вещественных чисел: 2.99792458e8 [14].

Также в документе «Методические рекомендации по выгрузке данных из информационных систем образовательных организаций высшего образования в подсистему виртуальной академической мобильности государственной информационной системы «Современная цифровая образовательная среда» приведены требования к процессу выгрузки данных:

Требования к порядку первичной загрузки, они обозначены в таблице 1.1.

Таблица 1.1 – требования к порядку первичной загрузки данных

Информационных объект	**Должен быть загружен до**
Сведения о студентах	Связь учебных планов и студентов, Движение контингента студентов
Сведения о сотрудниках ППС (если загрузка осуществляется)	Связь учебных планов и дисциплин, Статус сотрудников ППС
Сведения об образовательных программах	Сведения об учебных планах
Сведения об учебных планах	Связь учебных планов и дисциплин, Связь учебных планов и студентов
Сведения о дисциплинах	Связь учебных планов и дисциплин
Связь учебных планов и дисциплин	Сведения об оценках
Связь учебных планов и студентов	Сведения об оценках
Сведения об оценках, Движение контингента студентов	–

Требования ко времени загрузки: рекомендуется выполнять выгрузку сведений в ГИС СЦОС в любое время суток, кроме промежутка с 23:00 воскресенья до 03:00 понедельника (время по МСК).

Требования к размерам пакета: рекомендуется грузить 200 сущностей в пакете. Для массовых сущностей (например, «оценок») допускается грузить до 500 сущностей в пакете, если обновления производятся по всем сущностям с нуля [15].

2.2.2 Характеристика нормативно-справочной и входной оперативной информации

Вся информация, используемая ИС, берётся из уже существующей базы данных, а именно из следующих таблиц:

– CDB_DAT_STUDY_PROCESS.SPECIALITYEDUCATIONS – в этой таблице хранятся данные об учебных планах студентов;

– CDB_DAT_STUDY_PROCESS.SPECIALITYSESSIONS – в этой таблице хранятся наборы сессий и семестров для каждого учебного плана;

– CDB_DAT_STUDY_PROCESS.CYCLESPECIALITYEDUCATIONS – в этой таблице хранятся циклы учебных планов;

– CDB_DAT_STUDY_PROCESS.STUDYPLANS – в этой таблице хранятся дисциплины, реализуемые учебным планом;

– CDB_DAT_STUDY_PROCESS.LESSONTYPES – в этой таблице хранятся наборы часов по разным типам занятий для каждой дисциплины;

– CDB_DAT_STUDY_PROCESS.LESSONS – в этой таблице хранятся данные о занятиях;

– CDB_DAT_STUDY_PROCESS.SHEDULE – в этой таблице хранятся данные о расписании;

– CDB_DAT_STUDENTS.T_STUDENTS – в этой таблице хранятся основные данные о студенте;

– CDB_DAT_STUDENTS.T_PLANS – в этой таблице хранятся данные об индивидуальном плане обучения студента;

– CDB_DAT_STUDENTS.T_MARKS – в этой таблице хранятся оценки студента за контрольные точки;

– CDB_DAT_STUDENTS.T_ORDERS – в этой таблице хранится информация о приказах студентов;

– CDB_DAT_STUDENTS.T_CONTRACTS – в этой таблице хранится информация о договорах студента;

– CDB_DAT_STUDENTS.T_COURSEPAPERS – в этой таблице хранится информация о курсовых работах студента;

– CDB_DAT_STUDENTS.PRACTICS – в этой таблице хранится информация о практиках студента;

– CDB_DAT_PEOPLES.T_PEOPLES – в этой таблице хранится информация о физическом лице (ФИО, дата рождения, пол);

– CDB_DAT_PEOPLES.T_PHONES – в этой таблице хранится информация о всех номерах телефонов человека;

– CDB_DAT_PEOPLES.T_INNS – в этой таблице хранится информация об ИНН;

– CDB_DAT_PEOPLES.T_PASSPORTS – в этой таблице хранятся данные о паспортах.

Информацию из вышеперечисленных таблиц предстоит собирать при помощи SQL-запросов к базе данных. Это позволит сразу получать информацию с нужным набором столбцов [16].

2.2.3 Характеристика результатной информации

Результатная информация, формируемая в процессе функционирования ИС, представляет собой структурированные наборы данных, соответствующие регламентированным схемам взаимодействия с ГИС СЦОС. Данные приводятся к формально специализированному формату, совместимому с требованиями API ГИС СЦОС. Эти модели приведены в документе «Описание программных интерфейсов государственной информационной системы «Современная цифровая образовательная среда», размещенной в информационно-телекоммуникационной сети «Интернет» по адресу online.edu.ru» версии 2.8.5.

Перечень информации, передаваемой в ГИС СЦОС:

1) Сведения о студентах.
2) Сведения об образовательных программах.
3) Сведения об учебных планах.
4) Сведения о дисциплинах.
5) Связь учебных планов и дисциплин.
6) Связь учебных планов и студентов.
7) Сведения об оценках.
8) Движение контингента студентов.

2.2.4 Информационная модель и ее описание

Для обеспечения целостности и эффективности операций с данными, в проектируемой информационной системе реализована следующая структура хранения:

Все таблицы базы данных, обеспечивающие функционирование интеграционного контура расположены в специально выделенной схеме API_GIS. Также эти таблицы не имеют внешних ключей, это необходимо для того, чтобы не вмешиваться в работу уже существующих информационных систем [17]. ER-диаграмма этой схемы данных представлена на рисунке 2.2.

Рисунок 2.2 – ER-диаграмма схемы данных API_GIS

Поскольку выгрузка данных должна происходить регулярно, была создана таблица под названием SESSIONS. В этой таблице хранится информация и сессиях интеграции с ГИС СЦОС. Каждая запись в таблицах с данными для отправки будет иметь идентификатор сессии интеграции и тип

действия, которое необходимо сделать (обновить, добавить, удалить), таким образом становится возможных различать, когда данные были отправлены и когда они изменились.

Таблица _EFMigrationsHistory – является системной, в ней хранится информация о миграциях [18].

Все остальные таблицы напрямую соответствуют одноимённым объектам в API ГИС СЦОС.

Данные сохраняются в агрегированном состоянии, не дублирующем конечные модели целевой системы. Это обусловлено необходимостью поддержки множественных вариантов трансформации:

1. Преобразование в модели чтения (GET).
2. Генерация моделей создания (POST).
3. Формирование моделей обновления (PATCH).

Данное архитектурное решение устраняет избыточность хранения, снижает нагрузку на источники данных института и централизует логику преобразования. Соответствие паттерну Staging Area подтверждает эффективность подхода для ETL-систем

Список таблиц:

– CONTINGENT_FLOWS – в этой таблице хранится информация о движениях контингента студентов (переводов внутри института и переводов извне института);

– DISCIPLINES – в этой таблице хранится информация о дисциплине, её тип и название;

– EDUCATIONAL_PROGRAMS – в этой таблице хранится информация об образовательных программах;

– MARKS – в этой таблице хранится информация об оценках студентов за контрольные точки;

– SESSIONS – таблица для отслеживания сессий интеграции с ГИС СЦОС;

– STUDY_PLANS – в этой таблице хранится информация об учебных планах (в каждой образовательной программе может быть несколько учебных планов);

– STUDY_PLANS_DISCIPLINES – это таблица связи, в ней связываются учебные планы и дисциплины, которые эти учебные планы реализуют;

– STUDY_PLANS_STUDENTS – это таблица связи, в ней связываются учебные планы и студенты, обучающиеся по этому учебному плану;

– _EFMigrationsHistory – системная таблица.

2.3 Разработка программного обеспечения

2.3.1 Структурная схема функций управления и обработки данных

Основные функции информационной системы следующие:

Сбор данных из БД института. ИС собирает данные из базы данных из различных таблиц, а также эти данные приводятся в «промежуточное» состояние, это необходимо для дальнейшей возможности преобразования из «промежуточного» состояния к модели данных, соответствующей модели в ГИС СЦОС.

Преобразование данных в формат ГИС СЦОС. Для каждой таблицы был написаны по 3 механизма, которые преобразуют запись из таблицы в модели для GET, POST и PATCH запросов. Данный функционал позволяет не только преобразовывать данные при отправке их на сервера ГИС СЦОС, но и после сбора текущих данных из БД института, преобразовать их к единому формату и напрямую сравнивать с данными, полученными из ГИС СЦОС.

На рисунке 2.3 изображено дерево функций.

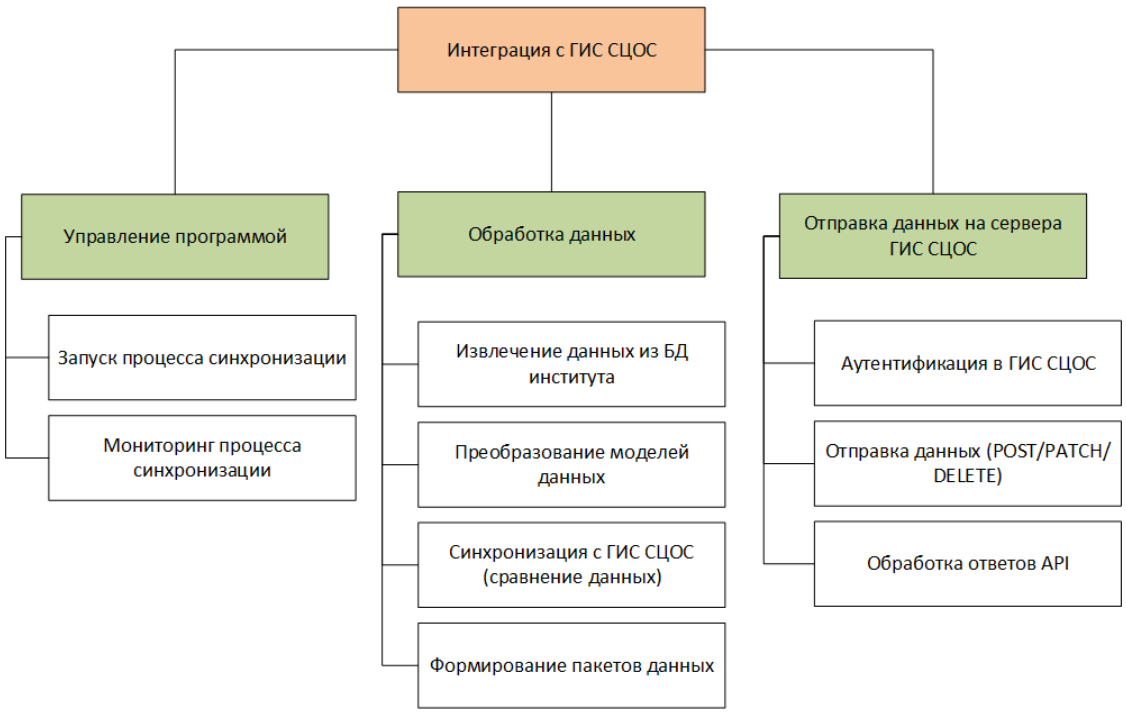

Рисунок 2.3 – Дерево функций

Синхронизация данных. Программа отправляет запросы на получение информации на сервера ГИС СЦОС и после их получения сравнивает их с актуальными данными из БД института. Сравнение данных необходимо для того, чтобы определить какие данные уже есть на серверах и ГИС СЦОС, какие нужно загрузить, а какие есть, но их нужно обновить.

Хранение пакетов данных. После того как программа определила какие данные необходимо загрузить на сервера ГИС СЦОС, она сохраняет их в базу данных. Это необходимо для того, чтобы сохранять историю отправки данных на сервера ГИС СЦОС. Пока данные не отправлены, то они при ежедневной синхронизации удаляются и на их место перезаписываются более актуальная информация.

Отправка данных в ГИС СЦОС. Данные отправляются при помощи Http-запросов. Система получает данные, которые необходимо отправить из БД и формирует пакеты, которое оно будет отправлять. К примеру, при добавлении данных нужно отправлять пакеты в которых будет не более 200 сущностей,

также для каждого запроса необходимо добавлять токен доступа, который необходимо получить на сервере авторизации.
Управление процессом. Имеется возможность запустить синхронизацию и отправку данных как по расписанию, так и вручную.

2.3.2 Описание программных модулей

Поскольку был выбран подход Code-First, то приложении описаны все модели для таблиц в базе данных [19]. Все классы таблиц унаследованы от базового класса сущности. В этой сущности содержится всего 1 свойство – идентификатор. Это базовая сущность необходима для реализации базового сервиса для работы с базой данных. Программный код с классом базовой сущности показан на рисунке 2.4.

```
using System.ComponentModel.DataAnnotations;
using WebCore.ApiGis.Models.BaseModels;

namespace WebCore.ApiGis.Entities
{
    public class BaseEntity
    {
        [Key]
        public int id { get; set; }
    }
}
```

Рисунок 2.4 – Программный код базовой сущности

От данной сущности наследуются следующие классы: SessionEntity (сессии синхронизации) и BaseSessionEntity – это базовая сущность для моделей, которые в дальнейшем и будет отправлять программа. В ней есть идентификатор сессии, к которой данные относятся, а также тип действия, которое необходимо сделать (обновить, удалить или добавить). Программный код BaseSessionEntity показан на рисунке 2.5.

```
using WebCore.ApiGis.Enums;

namespace WebCore.ApiGis.Entities
{
    public class BaseSessionEntity: BaseEntity
    {
        public int idSession { get; set; }
        public ActionsEnum action { get; set; }
    }
}
```

Рисунок 2.5 – Класс BaseSessionEntity

В качестве примера можно рассмотреть модель дисциплин, она показана на рисунке 2.6.

Помимо свойств модели в классе также описана логика преобразования этой модели к другим моделям для GET и PATCH запросов. По аналогии сделаны и остальные классы:

– ContingentFlowsEntity;
– EducationalProgramsEntity;
– MarksEntity;
– StudentsEntity;
– StudyPlansDisciplinesEntity;
– StudyPlansStudentsEntity.

Для взаимодействия с базой данных был написан базовый класс сервиса: CrudService. Этот класс реализует базовый функционал для создания, чтения, обновления и удаления для всех классов, унаследованный от BaseEntity. Фрагмент программного кода CrudService показан на рисунке 2.7.

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Reflection;
using System.Text.Json.Serialization;
using System.Xml.Linq;
using WebCore.ApiGis.Entities;
using WebCore.ApiGis.Enums;
using WebCore.ApiGis.Models.Students;
using static Dapper.SqlMapper;

namespace WebCore.ApiGis.Models.Disciplines
{
    /// <summary>
    /// Сущность дисциплин
    /// </summary>
    public class DisciplinesEntity: BaseIdEntity, IConvertible
    {
        /// <summary>
        /// Название дисциплины
        /// </summary>
        [StringLength(256)]
        public string title { get; set; }
        /// <summary>
        /// Типы дисциплин
        /// </summary>
        //[ JsonConverter(typeof(JsonStringEnumConverter)) ]
        public DisciplineTypesEnum? type { get; set; }
        ///// <summary>
        ///// Навигационное свойство
        ///// </summary>
        //[ForeignKey("idDiscipline")]
        //public List<StudyPlansDisciplinesEntity> StudyPlansDisciplines { get; set; }

        // Реализация интерфейса IConvertible
        public TypeCode GetTypeCode()
        {
            return TypeCode.Object;
        }

        public bool ToBoolean(IFormatProvider provider) => throw new InvalidCastException();
        public byte ToByte(IFormatProvider provider) => throw new InvalidCastException();
        public char ToChar(IFormatProvider provider) => throw new InvalidCastException();
        public DateTime ToDateTime(IFormatProvider provider) => throw new InvalidCastException();
        public decimal ToDecimal(IFormatProvider provider) => throw new InvalidCastException();
        public double ToDouble(IFormatProvider provider) => throw new InvalidCastException();
        public short ToInt16(IFormatProvider provider) => throw new InvalidCastException();
        public int ToInt32(IFormatProvider provider) => throw new InvalidCastException();
        public long ToInt64(IFormatProvider provider) => throw new InvalidCastException();
        public sbyte ToSByte(IFormatProvider provider) => throw new InvalidCastException();
        public float ToSingle(IFormatProvider provider) => throw new InvalidCastException();
        public string ToString(IFormatProvider provider) => throw new InvalidCastException();
        public ushort ToUInt16(IFormatProvider provider) => throw new InvalidCastException();
        public uint ToUInt32(IFormatProvider provider) => throw new InvalidCastException();
        public ulong ToUInt64(IFormatProvider provider) => throw new InvalidCastException();

        public object ToType(Type conversionType, IFormatProvider provider)
        {
            if (conversionType == typeof(DisciplinesEntity))
            {
                return this;
            }
            if (conversionType == typeof(DisciplinesInsertModel))
            {
                return new DisciplinesInsertModel
                {
                    external_id = external_id,
                    title = title,
                    type = type
                };
            }
            if (conversionType == typeof(DisciplinesUpdateModel))
            {
                return new DisciplinesUpdateModel
                {
                    external_id = external_id,
                    title = title,
                    type = type
                };
            }
            throw new InvalidCastException($"Cannot convert to {conversionType.Name}.");
        }
    }
}
```

Рисунок 2.6 – Модель дисциплин

```
using System.Linq.Expressions;
using System.Reflection;
using Microsoft.EntityFrameworkCore;
using WebCore.ApiGis.Entities;
using WebCore.ApiGis.Extensions;
namespace WebCore.ApiGis.Services
{
    public class CrudService<T> where T : BaseEntity, new()
    {
        protected ApplicationDbContext DbContext { get; set; }
        public CrudService(ApplicationDbContext dbContext)
        {
            DbContext = dbContext;
        }
        public async Task Save() {...}
        public virtual async Task<T> Insert(T o) {...}
        public virtual async Task<List<T>> Insert(List<T> o) {...}
        public virtual async Task<T> Update(T o)
        {
            if (o.id == 0)
            {
                throw new Exception("Нельзя обновить объект с идентификатором 0");
            }

            T oldItem;
            oldItem = await Get(o.id);
            oldItem.CopyProperties(o);
            await Save();
            o.CopyProperties(oldItem);
            return o;
        }
        public virtual void Delete(T o)
        {
            try
            {
                DbContext.Set<T>().Remove(o);
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
                throw;
            }
        }
        public virtual async Task<bool> Delete(IEnumerable<T> o) {...}
        public virtual async Task<bool> Delete(int id) {...}
        public virtual IQueryable<T> Where(Expression<Func<T, bool>> predicate)
        {
            return DbContext.Set<T>().Where(predicate);
        }
        public virtual async Task<T> Get(int id)
        {
            return await DbContext.Set<T>().FirstOrDefaultAsync(x => x.id == id);
        }

        public virtual IQueryable<T> GetAll()
        {
            return DbContext.Set<T>();
        }
    }
}
```

Рисунок 2.7 – Фрагмент кода CrudService

Базовый сервис позволят не прописывать в каждом сервисе функционал для работы с БД, поэтому для каждого сервиса остаётся создать функцию, которая бы делала SQL-запрос к БД. Сервис дисциплин показан на рисунке 2.8.

```
using System.Data;
using WebCore.ApiGis.Models.Students;
using WebCore.ApiTest.Services;
using Dapper;
using WebCore.ApiGis.Models.Disciplines;

namespace WebCore.ApiGis.Services
{
    public class DisciplinesService : CrudService<DisciplinesEntity>
    {
        public ConnectionFactory _factory { get; set; }
        public DisciplinesService(ApplicationDbContext dbContext, ConnectionFactory factory) : base(dbContext)
        {
            _factory = factory;
        }

        public async Task<IEnumerable<DisciplinesEntity>> GetFromDB()
        {
            using IDbConnection dbcon = _factory.CreateConnection();
            var qry = @"
           select sp.IDDISCIPLINENAME external_id, dn.DISCIPLINENAME title, dt.TYPE--,
           row_number() over (partition by sp.IDDISCIPLINENAME, dn.DISCIPLINENAME order by dt.TYPE)
           from CDB_DAT_STUDENTS.T_STUDENTS ts
           join cdb_dat_students.t_orders ord on ord.STUDENT_ID = ts.id
           join CDB_DAT_STUDy_process.SPECIALITYEDUCATIONS se
             on se.ID_SPECIALITYEDUCATION = ord.IDSPECIALITYEDUCATION or se.ID_SPECIALITYEDUCATION = ts.ADMISSION_ID
           join cdb_dat_study_process.CYCLESPECIALITYEDUCATIONS cse
                on cse.IDSPECIALITYEDUCATION = se.ID_SPECIALITYEDUCATION
           join CDB_DAT_STUDY_PROCESS.STUDYPLANS sp on sp.IDCYCLESPECIALITYEDUCATION = cse.ID_CYCLESPECIALITYEDUCATION
           join CDB_DAT_STUDY_PROCESS.LESSONTYPES lt on sp.ID_STUDYPLAN = lt.IDSTUDYPLAN
           join CDB_COM_DIRECTORIES.DISCIPLINETYPES dt on dt.ID_DISCIPLINETYPE = lt.IDTYPE
           join CDB_COM_DIRECTORIES.DISCIPLINENAMES dn on dn.ID_DISCIPLINENAME = sp.IDDISCIPLINENAME
  where ts.STATUS in (1,5)
    and ts.PEOPLE_ID <> 0       --все обучающиеся студенты(без виртуальных)
    and se.IDEDUCATIONLEVEL in (1,2) --бакалавриат
    and dt.TYPE is not null
  group by sp.IDDISCIPLINENAME, dn.DISCIPLINENAME, dt.TYPE";
            var result = new List<StudentsEntity>() as IEnumerable<DisciplinesEntity>;
            try
            {
                result = await dbcon.QueryAsync<DisciplinesEntity>(qry);
            }
            catch (Exception ex)
            {
                throw;
            }
            return result;
        }

    }
}
```

Рисунок 2.8 – Программный код DisciplinesService

В сервисе HttpRequestsService реализован функционал авторизации (рисунок 2.9) и работы с запросами. Этот сервис позволят отправлять Http-запросы на удаление (DELETE), обновление (PATCH), добавление (POST) и

получение (GET) данных из ГИС СЦОС. На рисунке 2.10 продемонстрирован фрагмент кода, который отвечает за обновление данных.

В классе DataPrepare реализован функционал для преобразования одной модели данных в другую. Код класса DataPrepare продемонстрирован на рисунке 2.11.

```
public HttpRequestsService()
{
    var httpClient = new HttpClient();
    httpClient.Timeout = TimeSpan.FromMinutes(5);
    httpClient.DefaultRequestHeaders.Add("X-CN-UUID", "**********************");
    var response = httpClient.GetAsync
    ("https://tls.online.edu.ru/api/v2/connections/check").Result;
    response.EnsureSuccessStatusCode();
    // Извлечение значений куки из заголовка Set-Cookie
    if (response.Headers.TryGetValues("Set-Cookie", out IEnumerable<string> cookieHeaders))
    {
       foreach (var header in cookieHeaders)
       {
           Console.WriteLine($"Set-Cookie: {header}");

           // Для конкретных кук, например JSESSIONID и SERVERID, можно извлечь их вручную
           var cookies = header.Split(';').Select(c => c.Trim());

           // Найти значения JSESSIONID и SERVERID
           string jsessionId = cookies.FirstOrDefault(c =>
                                       c.StartsWith("JSESSIONID="))?.Split('=')[1];
           string serverId = cookies.FirstOrDefault(c =>
                                         c.StartsWith("SERVERID="))?.Split('=')[1];
           httpClient.DefaultRequestHeaders.Add("Cookie", $"JSESSIONID={jsessionId}");
           if (jsessionId != null)
               Console.WriteLine($"JSESSIONID = {jsessionId}");
           if (serverId != null)
               Console.WriteLine($"SERVERID = {serverId}");
       }
    }
    httpClient.DefaultRequestHeaders.Add("User-Agent", "Twenty past four");
    httpClient.DefaultRequestHeaders.Add("Accept", "*/*");
    httpClient.DefaultRequestHeaders.Add("Accept-Encoding", "gzip, deflate, br");
    httpClient.DefaultRequestHeaders.Add("Connection", "keep-alive");
    _httpClient = httpClient;
}
```

Рисунок 2.9 – Фрагмент кода с авторизацией запросов

```
public async Task<List<string>> UpdateObjects<T>(Dictionary<string, T> objects, string path)
{
    var tasks = new List<Task>();
    var list = new List<string>();
    foreach (var obj in objects)
    {
        var objCopy = obj;//локальная копия
        tasks.Add(Task.Run(async () =>
        {
            var response = await SendToServer(obj.Value, path + "/" + obj.Key, HttpMethod.Put);
            lock (list)
            {
                list.Add(response);
            }
        }));
        //var response = await SendToServer(obj.Value, path + "/" + obj.Key, HttpMethod.Put);
        //list.Add(response);
    }
    await Task.WhenAll(tasks);
    return list;
}
public async Task<string> SendToServer(object? obj, string path, HttpMethod method)
{
    var content = new object() as JsonContent;
    var options = new JsonSerializerOptions
    {
        DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull,
        WriteIndented = true // Для удобного форматирования
    };
    if (obj != null)
    {
        content = JsonContent.Create(obj, options: options);
        await content.LoadIntoBufferAsync();
    }
    HttpResponseMessage response = new HttpResponseMessage();
    if (method == HttpMethod.Post)
    {
        try
        {
            response = await _httpClient.PostAsync(domain + path, content);
        } catch (Exception ex)
        {
            Console.WriteLine($"{ex.Message}");
            Console.WriteLine(ex.InnerException.Message);
        }

    } else if(method == HttpMethod.Put)
    {
        response = await _httpClient.PutAsync(domain + path, content);
    } else if (method == HttpMethod.Delete)
    {
        response = await _httpClient.DeleteAsync(domain + path);
    }
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
    return body;
}
```

Рисунок 2.10 – Функция для обновления данных на серверах ГИС СЦОС

```
using System.Reflection;
using WebCore.ApiGis.Entities;
using WebCore.ApiGis.Models.BaseModels;
using Newtonsoft.Json;
using WebCore.ApiGis.Services;

namespace WebCore.ApiGis.Extensions
{
    public static class DataPrepare
    {
        private static HttpRequestsService _httpService;
        public static void SetHttpService(IServiceProvider serviceProvider)
        {
            _httpService = serviceProvider.GetService<HttpRequestsService>();
        }
        public static List<Model> PrepareInsertData<Entity,Model>(List<Entity> entities)
            where Entity : BaseSessionEntity
            where Model : BaseModel, new()
        {
            var insertModels = new List<Model>();
            var insert = entities.Where(x => x.action == Enums.ActionsEnum.Insert).ToList();
            insert.ForEach(x => insertModels.Add((Model)Convert.ChangeType(x, typeof(Model))));
            return insertModels;
        }

        public static async Task<Dictionary<string,UpdateModel>> PrepareUpdateData<Entity,UpdateModel,GetModel>(List<Entity> entities)
            where Entity : BaseSessionEntity
            where UpdateModel : BaseModel, new()
            where GetModel : BaseGetModel, new()
        {
            var info = typeof(GetModel).GetCustomAttribute<Info>();
            var update = entities.Where(x => x.action == Enums.ActionsEnum.Update).ToList();
            var updateModels = new List<UpdateModel>();
            update.ForEach(x => updateModels.Add((UpdateModel)Convert.ChangeType(x, typeof(UpdateModel))));
            var result = new Dictionary<string, UpdateModel>();
            var tasks = new List<Task>();
            foreach (var model in updateModels)
            {
                tasks.Add(Task.Run(async () =>
                {
                    var conditions = new Dictionary<string, string>()
                    {
                        { "organization_id", GlobalInfo.organization_id },
                        { "external_id", model.external_id }
                    };
                    var response = (await _httpService.GetFromServer<GetModel>(info.Path, conditions)).First();
                    lock (result){
                        result.Add(response.id, model);
                    }

                }));

            }
            await Task.WhenAll(tasks);
            return result;
        }

        public static async Task<List<string>> PrepareDeleteData<Entity, GetModel>(List<Entity> entities)
            where Entity : BaseIdEntity
            where GetModel : BaseGetModel, new()
        {
            var info = typeof(GetModel).GetCustomAttribute<Info>();
            var delete = entities.Where(x => x.action == Enums.ActionsEnum.Delete).ToList();
            var result = new List<string>();
            foreach (var model in delete)
            {
                var conditions = new Dictionary<string, string>()
                    {
                        { "organization_id", GlobalInfo.organization_id },
                        { "external_id", model.external_id}
                    };
                var response = (await _httpService.GetFromServer<GetModel>(info.Path, conditions)).First();
                result.Add(response.id);
            }
            return result;
        }
    }
}
```

Рисунок 2.11 – Класс DataPrepare

Остальной функционал реализован непосредственно для каждой конечной точки (далее эндпоинт) в классе контроллера. На рисунке 2.12 показан программный код эндпоинта синхронизации дисциплин.

```
/// <summary>
/// Синхронизировать дисциплины
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("disciplines/")]
[ProducesResponseType((int)HttpStatusCode.NotFound)]
[ProducesResponseType((int)HttpStatusCode.BadRequest)]
[ProducesResponseType(typeof(IEnumerable<DisciplinesInsertModel>), (int)HttpStatusCode.OK)]
public async Task<IActionResult> SyncDisciplines()
{
    try
    {
        string path = (typeof(DisciplinesGetModel).GetCustomAttribute<Info>()).Path;
        var currentSession = await _sessionsService.Get();
        try
        {
            await _disciplineService.Delete(_disciplineService.Where(x => x.idSession == currentSession.id));
        }
        catch { }
        var entitiesFromDb = (await _disciplineService.GetFromDB()).ToList();
        var currentEntities = await _httpRequestsService.GetFromServer<DisciplinesGetModel>(path, GlobalInfo.organization_id_dictionary);
        var differentObjects =
            (from db in entitiesFromDb
            where !currentEntities.Any(x => x.external_id == db.external_id && x.title == db.title)
            select db).ToList(); //Проверяем есть ли отличающиеся объекты по полям выше

        foreach (var entity in differentObjects)
        {
            entity.idSession = currentSession.id;
            if (currentEntities.Where(x => x.external_id == entity.external_id).Any()) //Определяем нужно обновить или создать
            {
                entity.action = Enums.ActionsEnum.Update;
            }
            else
            {
                entity.action = Enums.ActionsEnum.Insert;
            }
        }

        var res = await _disciplineService.Insert(differentObjects);
        await _disciplineService.Save();
        if (res.Any())
        {
            return new JsonResult(res);
        }
        else
        {
            return NotFound();
        }
    }
    catch (Exception e)
    {
        return BadRequest(error: e);
    }
}
```

Рисунок 2.12 – Программный код эндпоинта синхронизации дисциплин

2.3.3 Компоненты пользовательского интерфейса

Поскольку ИС реализована при помощи фреймворка ASP.NET, то всё взаимодействие возможно только при помощи отправки Http-запросов на различные конечные точки. В стандартной поставке ASP.NET включает

Swagger, что позволяет создать визуализацию для эндпоинтов. Swagger позволяет визуализировать документацию эндпоинтов. В ASP.NET документация создаётся автоматически, но можно добивать дополнительную информации при помощи XML-комментариев в коде программы.

На рисунке 2.13 изображён визуальный интерфейс Swagger.

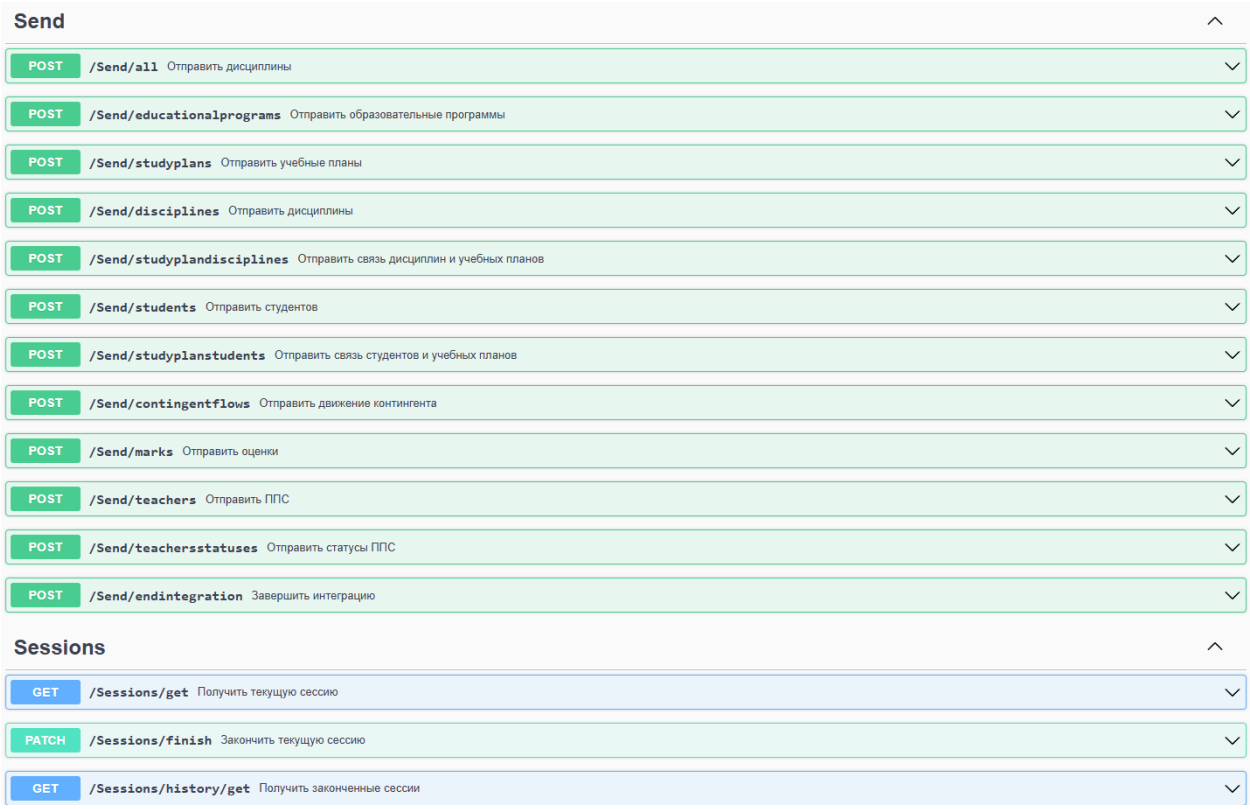

Рисунок 2.13 – Интерфейс Swagger

После успешной выгрузки данных каждый студент ВО может зайти на сайт ГИС СЦОС, авторизоваться при помощи Госуслуг и посмотреть в графическом интерфейсе следующую информацию о себе: данные о предыдущем образовании (рисунок 2.14), о текущем учебном плане (рисунок 2.15), зачётную книжку (рисунок 2.16).

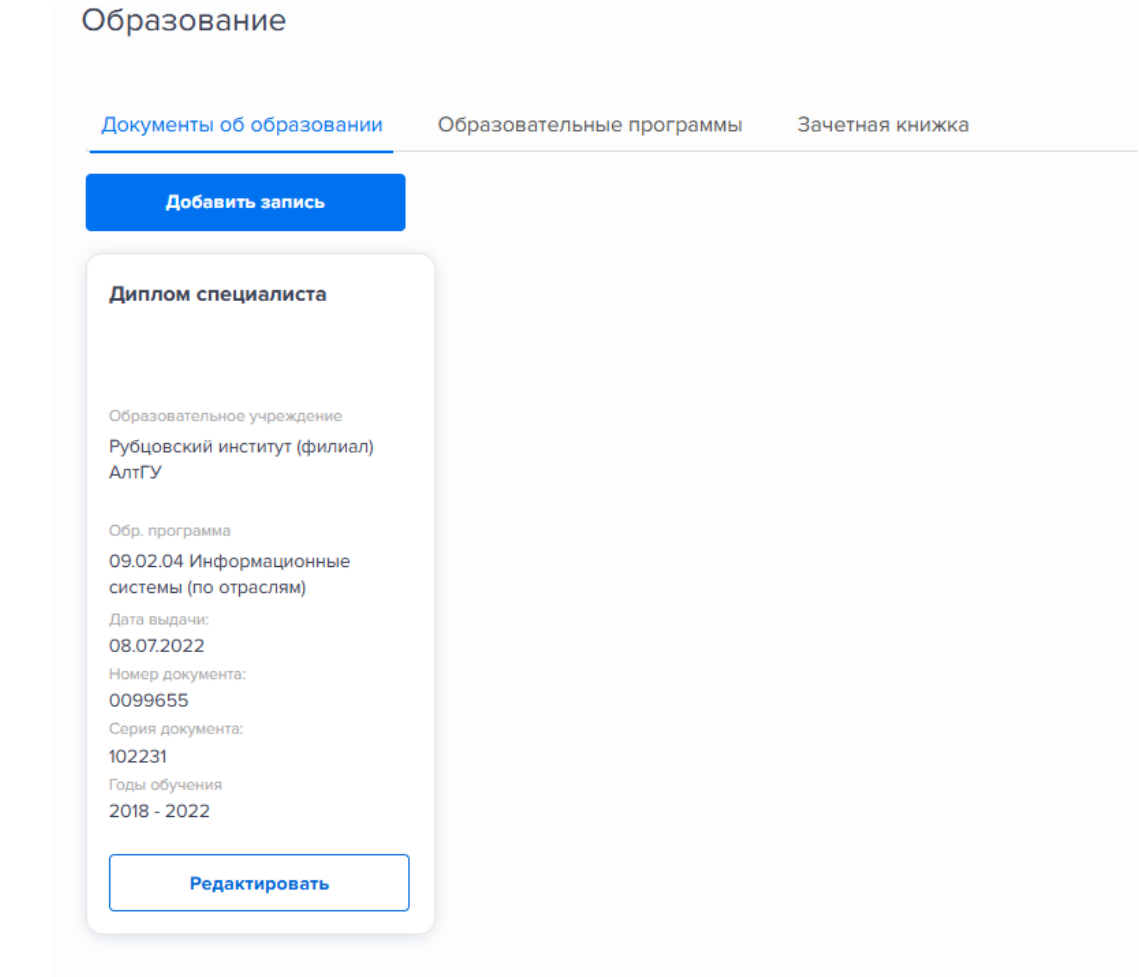

Рисунок 2.14 – Раздел «Документы об образовании» на сайте ГИС СЦОС

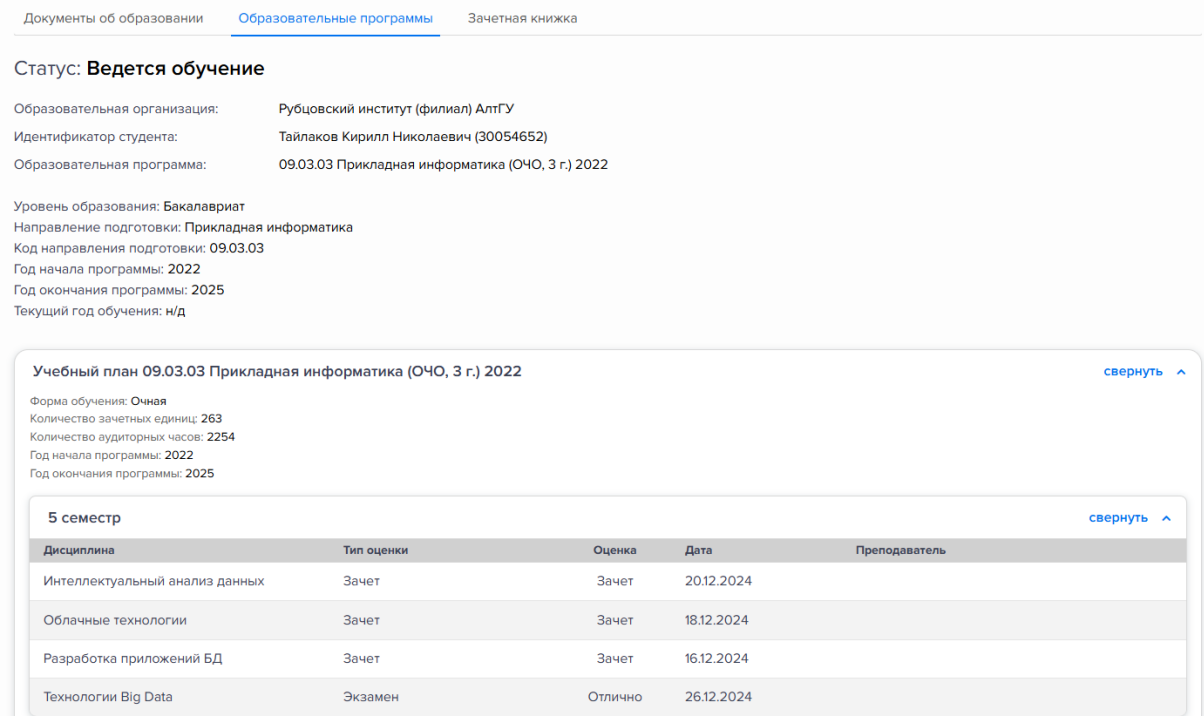

Рисунок 2.15 – Раздел «Образовательные программы» на сайте ГИС СЦОС

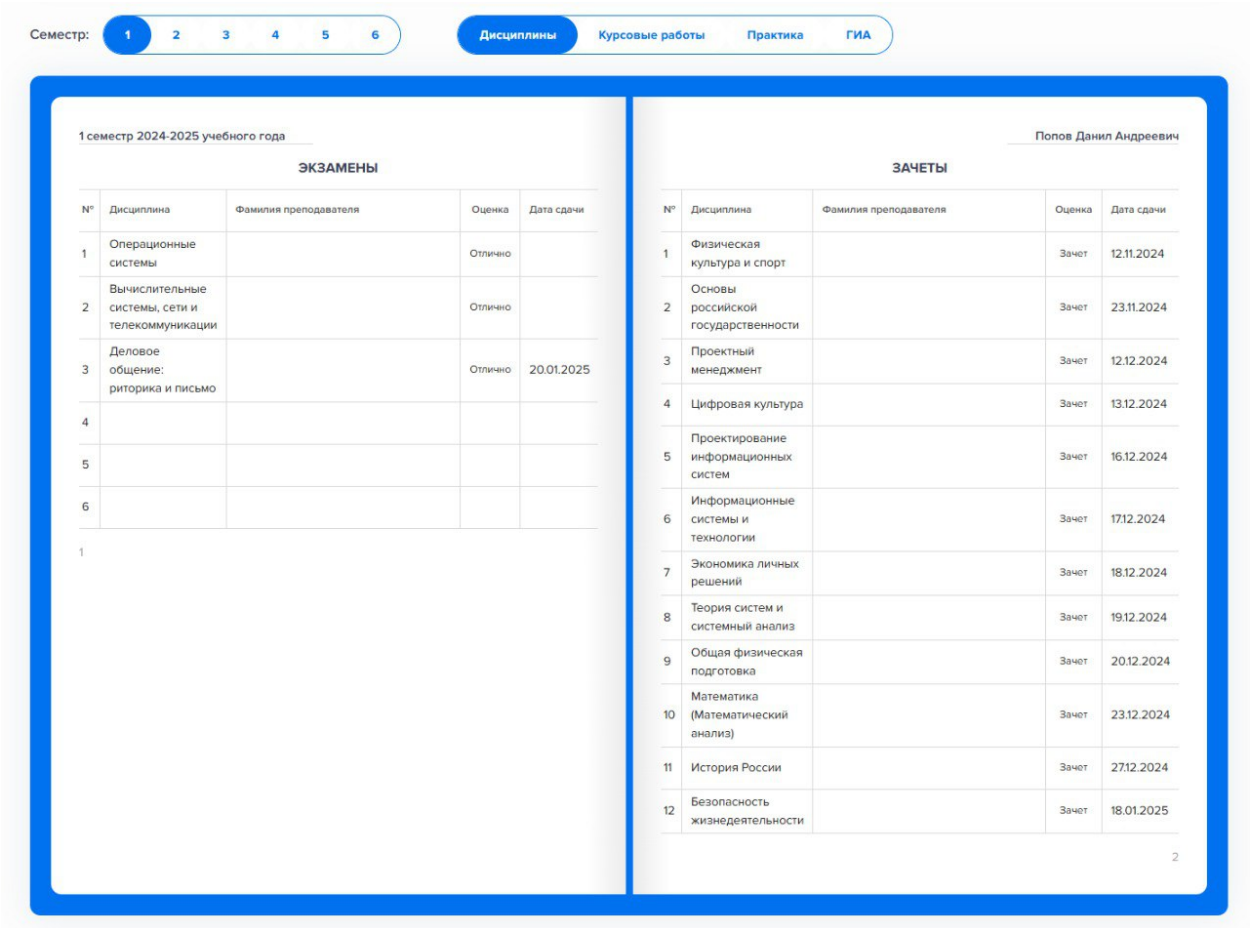

Рисунок 2.16 – Раздел «Зачётная книжка» на сайте ГИС СЦОС

2.4 Компьютерно-сетевое оборудование

2.4.1 Выбор размера сети и ее структуры

Для проведения работы была выбрана локальная сеть института. Её конфигурация полностью соответствует требованиям ИС к сети, поэтому она не подвергалась изменениям.

2.4.2 Выбор сетевого оборудования

Приложение развёрнуто на новом сервере, его характеристики:

1. Платформа Atermiter X79.
2. Процессор 12x Intel Xeon E5-2020 v2.
3. Оперативная память 16Гб DDR3.
4. 2x SSD ADATA SU800 128 Гб.

2.5 Обеспечение информационной безопасности

2.5.1 Область физической безопасности

Для обеспечения безопасности серверного оборудования и систем хранения данных применяется следующие меры:

1. Данные хранятся на защищённом сервера с регулярным резервным копированием для возможности восстановления данных.
2. Используются источник бесперебойного питания.
3. Доступ в серверную строго контролируется и разрешён только авторизованному персоналу.

Физическая безопасность обеспечивается за счёт:

4. Круглосуточного видеонаблюдения у входа в серверное помещение.
5. Охраны территории института частным охранным предприятием.
6. Система контроля доступа с использованием карт доступа.
7. Дверные замки.

2.5.2 Область безопасности оборудования

Сервер подключён к электричеству через ИБП, что позволяет обеспечить беспрерывную работу или возможность штатного выключения сервера при отключении электроэнергии.

Физический доступ к оборудованию имеет только квалифицированный персонал из Отдела технического и программного обеспечения. Подключение же к серверу осуществляется при помощи протокола SSH. Доступ осуществляется по шифрованному каналу и для входа требует ввода пароля.

2.5.3 Область безопасности программного обеспечения

Безопасность программного обеспечения обеспечивается за счёт ежедневного резервного копирования БД, также доступ возможно получить только из локальной сети института, а также только для разрешённых IP-адресов.

2.5.4 Правовая область безопасности

Меры защиты информации осуществляются в соответствии с ФЗ №149 «Об информации, информационных технологиях и о защите информации». Закон включает следующие меры защиты информации:

1. Предотвращение несанкционированного доступа, защита от уничтожения и модификации информации, недопущение незаконного копирования и распространения данных.
2. Обеспечение конфиденциальности сведений ограниченного доступа.
3. Реализация права на доступ к информации [20].

2.5.5 Защита персональных данных

В рамках функционирования информационных систем, осуществляется сбор, хранение, обработка и передача персональных данных пользователей.

В целях обеспечения их конфиденциальности реализуется комплекс организационных и технических мер защиты.

Документы, регламентирующие общие требования по защите информации в организации:

1. Положение о пропускном режиме.
2. Приказ о назначении ответственного за защиту информации и распределении обязанностей.
3. Инструкция о работе в сети интернет и использовании электронной почты организации.
4. Инструкция пользователя.

3 Оценка эффективности внедрения информационной системы

3.1 Общие положения

Внедрение модуля интеграции с ГИС СЦОС требует предварительного анализа его эффективности, определяющего обоснованность и целесообразность разработки.

Эффективность информационной системы представляет собой способность системы достигать поставленных целей, с заданным качеством при минимальных затратах ресурсов.

Экономическая эффективность рассчитывается и оценивается путем сопоставления результирующих показателей использования информационной системы с эксплуатацией данной системы после её внедрения.

Эффективность информационной системы отражает следующие характеристики:

– экономическая целесообразность внедрения информационной системы;
– техническое совершенство информационной системы;
– простота и технологичность разработки и создания системы;
– удобство использования и обслуживания системы;
– улучшение и облегчение условий труда;
– действенность системы.

При разработке модуля информационной системы основной задачей является обеспечение требуемого качества и минимальные затраты ресурсов.

Свойства системы, в которых обуславливается возможность её использования, для удовлетворения определенных потребностей пользователей в соответствии с её назначением, называются качеством.

К основным показателям качества информационной системы обычно относят:

– надёжность – это свойство системы сохранять во времени в установленных пределах значения всех параметров;

– безопасность – это свойство, заключающееся в способности системы обеспечить конфиденциальность и целостность информации;

– достоверность – это свойство системы, обусловливающее безошибочность производимых ею преобразований информации.

В соответствии с «ГОСТ Р ИСО 9000-2015», эффективность определяется как соотношение между достигнутым результатом и использованными ресурсами. Соответственно, разработка и внедрение информационной системы требует комплексной оценки всех характеристик и продуманного подхода к снижению издержек при сохранении высокого качества.

3.2 Показатели эффективности

Показатели эффективности характеризуют уровень приспособленности разработанной системы, с целью выполнения поставленных перед нею задач и являются обобщающими показателями оптимальности функционирования информационной системы.

Обобщающими показателями являются показатели экономической эффективности, которые характеризует снижение трудозатрат, уменьшение издержек за счёт отказа от устаревших систем.

В качестве прагматической эффективности выступают следующие показатели:

– приведение разрозненной информации;

– снижение количества ошибок при обмене данных;

– исполнение законодательных требований по передаче данных в государственные информационные системы.

Показатели технической эффективности оценивают техническое совершенство информационной системы при работе ее в различных режимах и оценивают научно-технический уровень организации и функционирования этой системы, что выражается в возможности работы системы в автоматическом режиме.

К социальной эффективности относится возможность каждого студента просмотреть актуальную информацию об образовательной программе, на которой он обучается, а также студенты могут записаться на онлайн-курсы при помощи федерального портала «Моё образование».

Экономический эффект от внедрения разработанного модуля интеграции с ГИС СЦОС выражается в автоматизации процесса отправки данных на сервера ГИС СЦОС. Основными источниками экономии являются:

– снижение трудоёмкости за счёт полной автоматизации процесса отправки данных в ГИС СЦОС;

– повышение надёжности и своевременности передачи данных, минимизирующее потенциальные издержки от несоответствия требованиям регулятора.

3.3 Расчет экономической эффективности

3.3.1 Смета затрат на разработку

Затраты на проектирование и разработку модуля интеграции с ГИС СЦОС состоят из затрат на покупные материалы, основной заработной платы, дополнительный заработной планы, страховых взносов, накладных расходов, затрат на электроэнергию.

Затраты на покупные изделия рассчитываются по формуле (3.1).

$$З_{мi} = Ц_{едi} * N_{мi} \quad (3.1)$$

где $З_{мi}$ – затраты на i-й материал, $Ц_{едi}$ – цена за единицу i-го материала, $N_{мi}$ – количество i-го материала.

Общие затраты на материалы определяются по формуле (3.2).

$$З_{м} = \sum З_{мi} \quad (3.2)$$

где $З_{м}$ – общие затраты на материалы.

В таблице 3.1 приведен перечень затрат на материалы и покупные изделия.

Таблица 3.1 – Затраты на материалы и покупные изделия

№ п/п	Наименование материала	Кол-во, шт.	Цена за единицу, руб.	Стоимость, руб.
1	Лицензия Secret Net Studio	1	5000	5000
2	Лицензия Dr. WEB	1	8000	8000
3	Итого			13000

Таким образом, общие затраты на материалы составляют 13000 рублей.

Далее произведём расчёт фонда заработной платы (основной и дополнительный заработной платы программиста по формуле (3.3).

$$З_{нач} = З_{осн} * З_{доп} \quad (3.3)$$

где $З_{осн}$ – размер основной заработной планы, $З_{доп}$ – размер дополнительной заработной платы, в которую входят выплаты, предусмотренные трудовым договором, определяется по формуле (3.4).

$$З_{доп} = З_{осн} * 0{,}1 \quad (3.4)$$

Результаты расчета представлены в таблице 3.2.

Таблица 3.2 – Расчет фонда заработной платы

№ п/п	Должность: программист	Кол-во рабочих дней	Кол-во проработанных дней	Размер дневной оплаты, руб.	Заработная плата, руб.
1	Основная заработная плата	31	31	910	28 210
2	Дополнительная заработная плата				2821
3	Итоговый фонд заработной платы				31 031

Таким образом, исходя из таблицы 3.2, фонд заработной платы программиста составляет 28210 рублей.

К отчислениям на социальные нужды относят страховые взносы в ПФР, ФСС, ФОМС и взносы на страхование от несчастных случаев на производстве и профессиональных заболеваний.

Отчисления в пенсионный фонд определяются по формуле (3.5).

$$З_{пф} = З_{нач} * 0{,}22 = 31\,031 * 0{,}22 = 6\,826{,}82 \text{ руб.} \quad (3.5)$$

Отчисления на социальное страхование рассчитываются по формуле (3.6).

$$З_{сс} = З_{нач} * 0{,}029 = 31\,031 * 0{,}029 = 899{,}89 \text{ руб.} \quad (3.6)$$

где $З_{сс}$ – размер отчислений на социальное страхование.

Отчисления в фонд обязательного медицинского страхования определяются по формуле (3.7).

$$З_{мс} = З_{нач} * 0{,}051 = 31\,031 * 0{,}051 = 1\,582{,}58 \text{ руб.} \quad (3.7)$$

где $З_{мс}$ – размер отчислений в фонд обязательного медицинского страхования.

Отчисления на обязательное социальное страхование от несчастных случаев на производстве рассчитываются по формуле (3.8).

$$З_{нс} = З_{нач} * 0{,}002 = 31\,031 * 0{,}002 = 62{,}06 \text{ руб.} \quad (3.8)$$

где $З_{нс}$ – размер отчислений на обязательное страхование от несчастных случаев на производстве.

В таблице 3.3 представлены численные значения отчислений на социальные нужды.

Таблица 3.3 – Расчет отчислений на социальные нужны (страховые взносы)

№ п/п	Отчисления на социальные взносы (страховые нужды)	Тарифы страховых взносов, в %	Суммы страховых взносов, руб.
1	ПФР	22,00	6 826,82
2	ФСС	2,90	899,89
3	ФОМС	5,10	1 582,58
4	На обязательное социальное страхование от несчастных случаев на	0,20	62,06

	производстве и профессиональных заболеваний		
5	Итого	30,20	9 371,35

Стоимость машинного времени зависит от себестоимости машино-часа работы машины, времени работы и амортизации машины и оборудования, а также затраты на электроэнергию ((3.9) – (3.11)).

$$А_М = \frac{О_ф * Н_{ам}}{365 * 100} * Т_м \quad (3.9)$$

Стоимость сервера составляет 50000 рублей, норма амортизации принята равной 25%. Таким образом, $А_М$ = (850000/36500) * 70 = 1630,14 рублей

Рассчитаем дополнительные расходы к основным затратам на процессы производства и обращения. Накладные расходы $З_н$ составляют 20 процентов от суммы основной и дополнительной заработной платы:

$$З_н = З_{нач} * 0,2 = 31031 * 0,2 = 6\,206,2 \text{ руб.} \quad (3.10)$$

Рассчитаем затраты на машинное время. Как следует из данных таблицы 3.2, на разработку и тестирование личного кабинета студента потребовался 31 рабочий день ($Д_н$). В среднем, с учётом перерывов программист работает за компьютеров 7 часов в день. Себестоимость одного кВт/ч электроэнергии ($С_{1квт/ч}$) для организаций составляет 7 рублей 58 копеек. Складывая мощность энергопотребителей для программиста из мощности, потребляемой системным блоком компьютера, монитором и другими периферийными оборудованиями, то она составит 1,2 кВт. Следовательно, за 7 часов работы программиста, суммарное энергопотребление за день составит: Р = 1,2 * 7 =

8,4 кВт/ч. Таким образом, стоимость машинного времени ($З_{маш}$), необходимого для разработки личного кабинета студента составит:

$$З_{маш} = P * Д_{н} * С_{\frac{1квт}{ч}} = \frac{8{,}4кВт}{ч} * 31 * 7,58 \text{ руб.} \frac{\frac{руб}{кВт}}{ч} = 1\,973{,}83 \text{ руб.} \quad (3.11)$$

Затраты на машинное время учитываются как затраты на электроэнергию. В результате произведённых выше расчётов, были получены итоговые затраты на разработку, указанные в таблице 3.4.

Таблица 3.4 – Итоговая смета затрат

№ п/п	Наименование статей расхода	Сумма, руб.
1	Стоимость материалов и покупных изделий	13000
2	Основная заработная плата	28210
3	Дополнительная заработная плата	2821
4	Отчисления на социальные нужды	9371,35
5	Амортизация ЭВМ и оборудования	1630,14
6	Накладные расходы	6206,2
7	Затраты на машинное время	1973,83
8	Итого	63212,52

Цена программного продукта определяется по формуле (3.12).

$$Ц = З_{итог} + П = З_{итог} + З_{нач} * 0{,}3. = 63212{,}52 + 31031 * 0{,}3 = 72521{,}82 \quad (3.12)$$

где Ц – итоговая цена программного продукта, $З_{итог}$ – итоговые затраты, П – прибыль, которая составляет 30 процентов от фонда заработной платы.

ЗАКЛЮЧЕНИЕ

Целью выпускной квалификационной работы являлась разработка модуля интеграции с Государственной информационной системой «Современная цифровая образовательная среда» (на примере Рубцовского института (филиала) АлтГУ).

Для достижения поставленной цели были решены следующие задачи:

– проведён анализ предметной области;

– разработана архитектура и выработаны проектные решения по обеспечивающим подсистемам;

– реализованы проектные решения разрабатываемой модуля интеграции с ГИС СЦОС;

– выполнена оценка эффективности внедрения модуля интеграции с ГИС СЦОС.

Разработанная информационная система «Модуль интеграции с ГИС СЦОС» позволит институту своевременно передавать данные об учебной процессе в ГИС СЦОС. Созданное решение не только решает конкретную техническую задачу, но и вносит существенный вклад в цифровую трансформацию образовательного процесса Рубцовского института, повышая его управляемость, прозрачность и соответствие современным стандартам.

Практическая значимость проекта подтверждена расчётом ряда экономических показателей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рубцовский институт (филиал) АлтГУ [Электронный ресурс] / Режим доступа: https://rb.asu.ru/ - Загл. с экрана.

2. Астапчук, В. А. Корпоративные информационные системы: требования при проектировании: учебник для вузов / В. А. Астапчук, П. В. Терещенко. – 3-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2025. – 175 с. – (Высшее образование). – ISBN 978-5-534-16715-3. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/562833/ – Загл. с экрана

3. Богатырев, В. А. Информационные системы и технологии. Теория надежности: учебное пособие для вузов / В. А. Богатырев. – 2-е изд. – Москва: Издательство Юрайт, 2024. – 366 с. – (Высшее образование). – ISBN 978-5-534- 15951-6. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/510320 – Загл. с экрана.

4. Богатырев, В. А. Информационные системы и технологии. Теория надежности: учебное пособие для вузов / В. А. Богатырев. – 2-е изд. – Москва: Издательство Юрайт, 2024. – 366 с. – (Высшее образование). – ISBN 978-5-534- 15951-6. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/510320 – Загл. с экрана.

5. Зараменских, Е. П. Информационные системы: управление жизненным циклом: учебник и практикум для вузов / Е. П. Зараменских. – 2-е изд. – Москва: Издательство Юрайт, 2025. – 486 с. – (Высшее образование). – ISBN 978-5-534-21415-4. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/571328 – Загл. с экрана.

6. Грекул, В. И. Проектирование информационных систем: учебник и практикум для вузов / В. И. Грекул, Н. Л. Коровкина, Г. А. Левочкина. – 2-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2025. – 404 с. – (Высшее образование). – ISBN 978-5-534-19505-7. – Текст: электронный //

Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/560976 – Загл. с экрана.

7. Портал разработчиков ГИС СЦОС [Электронный ресурс] / Режим доступа: https://tech.online.edu.ru/ - Загл. с экрана.

8. Тузовский, А. Ф. Проектирование и разработка web-приложений : учебник для вузов / А. Ф. Тузовский. – Москва : Издательство Юрайт, 2025. – 219 с. – (Высшее образование). – ISBN 978-5-534-16300-1. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/561176 – Загл. с экрана

9. Web-разработки в asp. Net web forms : учебник для вузов / С. Т. Гуляева, В. В. Миронов, Н. О. Котелина, И. И. Лавреш. – Москва : Издательство Юрайт, 2025. – 134 с. – (Высшее образование). – ISBN 978-5-534-19885-0. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/569218 – Загл. с экрана

10. Тузовский, А. Ф. Объектно-ориентированное программирование : учебник для вузов / А. Ф. Тузовский. – Москва : Издательство Юрайт, 2025. – 213 с. – (Высшее образование). – ISBN 978-5-534-16316-2. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/561394 – Загл. с экрана

11. Внуков, А. А. Защита информации: учебник для вузов / А. А. Внуков. – 3-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2025. – 161 с. – (Высшее образование). – ISBN 978-5-534-07248-8. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/561313 – Загл. с экрана.

12. Зенков, А. В. Информационная безопасность и защита информации: учебник для вузов / А. В. Зенков. – 2-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2025. – 107 с. – (Высшее образование). – ISBN 978-5-534-16388-9. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/567915 – Загл. с экрана.

13. Казарин, О. В. Надежность и безопасность программного обеспечения: учебник для вузов / О. В. Казарин, И. Б. Шубинский. – 2-е изд. – Москва: Издательство Юрайт, 2025. – 352 с. – (Высшее образование). – ISBN 978-5-534-19386-2. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/580669 – Загл. с экрана.

14. Описание программных интерфейсов государственной информационной системы «Современная цифровая образовательная среда», размещенной в информационнотелекоммуникационной сети «Интернет» по адресу online.edu.ru [Электронный ресурс] / Режим доступа: https://tech.online.edu.ru/files/api.pdf – Загл. с экрана.

15. Методические рекомендации по выгрузке данных из информационных систем образовательных организаций высшего образования в подсистему виртуальной академической мобильности государственной информационной системы «Современная цифровая образовательная среда» [Электронный ресурс] / Режим доступа: https://tech.online.edu.ru/files/upload_guidelines.pdf – Загл. с экрана.

16. Маркин, А. В. Программирование на SQL : учебник и практикум для вузов / А. В. Маркин. – 3-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2025. – 805 с. – (Высшее образование). – ISBN 978-5-534-18371-9. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/568900 – Загл. с экрана

17. Гордеев, С. И. Организация баз данных : учебник для вузов / С. И. Гордеев, В. Н. Волошина. – 2-е изд., испр. и доп. – Москва : Издательство Юрайт, 2025. – 691 с. – (Высшее образование). – ISBN 978-5-534-21115-3. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/559377 – Загл. с экрана

18. Подбельский, В. В. Программирование. Базовый курс C# : учебник для вузов / В. В. Подбельский. – Москва : Издательство Юрайт, 2025. – 369 с. – (Высшее образование). – ISBN 978-5-534-10616-9. – Текст :

электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/560848 – Загл. с экрана

19. Советов, Б. Я. Базы данных : учебник для вузов / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 4-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2025. – 403 с. – (Высшее образование). – ISBN 978-5-534-18479-2. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/559898 – Загл. с экрана

20. Организационное и правовое обеспечение информационной безопасности: учебник для вузов / Т. А. Полякова, А. А. Стрельцов, С. Г. Чубукова, В. А. Ниесов; под редакцией Т. А. Поляковой, А. А. Стрельцова. – 2-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2025. – 357 с. – (Высшее образование). – ISBN 978-5-534-19108-0. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/560516 – Загл. с экрана.