

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных систем и сетей

Кафедра Программного обеспечения информационных технологий

К защите допустить:

Заведующая кафедрой ПОИТ

_____ Н. В. Лапицкая

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему:

**ПРОГРАММНОЕ СРЕДСТВО АВТОМАТИЗАЦИИ РАБОТЫ
ПРЕПОДАВАТЕЛЯ**

БГУИР ДП 1-40 01 01 01 133 ПЗ

Студент

Е. С. Шульга

Руководитель

К. А. Сурков

Консультанты:

*от кафедры ПОИТ
по экономической части*

К. А. Сурков
В. А. Палицын
Г. В. Данилова

Нормоконтролёр

Рецензент

Минск 2017

РЕФЕРАТ

Пояснительная записка 119 с., 15 рис., 10 табл., 30 формул, 36 источников.
ПРОГРАММНОЕ СРЕДСТВО, ВЕБ-ПРИЛОЖЕНИЕ, УНИВЕРСИТЕТ,
ОРГАНИЗАЦИЯ УЧЕБНОГО ПРОЦЕССА, РАСПИСАНИЕ,
ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Цель настоящего дипломного проекта состоит в разработке программной системы, предназначенной для эффективной автоматизации задач участников учебного процесса: студентов и преподавателей.

В процессе анализа предметной области были выделены основные аспекты процесса образования в университетах, которые в настоящее время практически не охвачены автоматизацией. Было проведено их исследование и моделирование. Кроме того, рассмотрены существующие средства, разрозненно применяемые сотрудниками университетов и обучаемыми людьми (так называемые частичные аналоги). Выработаны функциональные и нефункциональные требования.

Была разработана архитектура программной системы, для каждой ее составной части было проведено разграничение реализуемых задач проектирование, уточнение используемых технологий и собственно разработка. Были выбраны наиболее современные средства разработки, широко применяемые в индустрии.

Полученные в ходе технико-экономического обоснования результаты о прибыли для разработчика, пользователя, уровень рентабельности, а также экономический эффект доказывают целесообразность разработки проекта.

СОДЕРЖАНИЕ

Введение	7
1 Анализ литературных источников, прототипов и формирование требований к проектируемому программному средству	9
1.1 Аналитический обзор литературных источников	9
1.2 Обзор существующих аналогов	15
1.3 Требования к проектируемому программному средству	23
2 Анализ требований к программному средству и разработка функциональных требований	30
2.1 Функциональная модель программного средства	30
2.2 Разработка спецификации функциональных требований	38
3 Проектирование и разработка программного средства	42
3.1 Разработка архитектуры программного средства	42
3.2 Разработка даталогической и физической моделей базы данных .	43
3.3 Проектирование и разработка серверной части программного средства	45
3.4 Проектирование и разработка клиентской части программного средства	51
3.5 Разворачивание программного средства	62
4 Тестирование и проверка работоспособности программного средства	64
5 Методика использования программного средства	74
6 Технико-экономическое обоснование разработки и внедрения программного средства	77
6.1 Характеристика программного средства	77
6.2 Определение объема и трудоемкости программного средства .	77
6.3 Расчет сметы затрат	80
6.4 Оценка экономической эффективности применения программного средства у пользователя	84
Заключение	87
Список использованных источников	88
Приложение А. Исходный код	91
Приложение Б. Описание схемы базы данных	109
Приложение В. Конфигурационные файлы проекта	117

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

Спецификация – документ, который желательно полно, точно и верифицируемо определяет требования, дизайн, поведение или другие характеристики компонента или системы, и, часто, инструкции для контроля выполнения этих требований [1].

Веб-приложение – клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер.

Кроссплатформенность – способность программного обеспечения работать более чем на одной аппаратной платформе и (или) операционной системе.

Нативное программное средство – программное средство, специфичное для какой-либо платформы [2].

Проприетарное программное обеспечение – программное обеспечение, являющееся частной собственностью авторов или правообладателей и не удовлетворяющее критериям свободного ПО: свобода запуска программы в любых целях, свобода адаптации программы для любых нужд, свобода распространения, свобода улучшений исходных кодов и публикации улучшений [3].

ВУЗ – высшее учебное заведение.

ПС – программное средство.

ПО – программное обеспечение.

БД – база данных.

СУБД – система управления базами данных.

ЯП – язык программирования.

API – application programming interface (сетевой программный интерфейс).

UI – user interface (пользовательский интерфейс).

ТЭО – технико-экономическое обоснование.

ВВЕДЕНИЕ

Люди давно поняли, что управление временем, задачами и контактами – одни из самых важных условий повышения личной эффективности. Наука управления временем объединяет всевозможные техники и приемы, позволяющие беречь драгоценное время, распределять его более рационально и благодаря этому быстрее достигать своих целей [4]. Управление задачами представляет собой их систематическое фиксирование, ранжирование по приоритету, выполнение, контроль статуса и истории их выполнения [5]. Построение сети персональных контактов способствует поискам работы и продвижению по карьерной лестнице.

Перечисленные задачи особенно актуальны для людей, участвующих в образовательном процессе в ВУЗах: для студентов, преподавателей, а также для работников деканатов и других структурных подразделений университетов. У студентов и преподавателей практически на каждый день в расписании запланировано несколько занятий, в любое время у них есть задачи, которые они обязаны выполнить, студенты постоянно взаимодействуют с большим числом преподавателей, у которых в свою очередь есть еще большее число студентов: даже просто запомнить такое количество имен и лиц практически невозможно. Но на данный момент не существует решения, которое бы объединяло всю вышеперечисленную функциональность. И людям для выполнения своих задач приходится комбинировать существующие и приспосабливать под свои нужды. Каждый человек вынужден тратить своё время на поиск средств, тратить время на их конфигурирование. Возникает разрозненность способов, которая приводит к тому, что другим людям также приходится тратить своё время на адаптацию своего набора средств к другим людям.

Для этих задач реализовано огромное число приложений для мобильных, настольных операционных систем, большое число веб-приложений; кроме того, люди по-прежнему продолжают использовать ежедневники, записные книжки и бумажные календари.

Целью настоящего дипломного проекта является разработка программного средства, которое в рамках ВУЗа смогло бы предложить участникам учебного процесса унифицированные способы управления календарём, задачами, а также предоставляет канал обмена информацией между студентами и преподавателями.

В пояснительной записке к дипломному проекту излагаются детали поэтапной разработки приложения повышения профессиональной эффективности в процессе обучения. В первом разделе приведены результаты анализа литературных источников по теме дипломного проекта, рассмотрены особенности существующих систем-аналогов, выдвинуты требования к проектируемому ПС. Во втором разделе приведено описание функциональности проектируемого ПС, представлена спецификация функциональных требований. В третьем

разделе приведены детали проектирования и конструирования ПС. Результатом этапа конструирования является функционирующее программное средство. В четвертом разделе представлены доказательства того, что спроектированное ПС работает в соответствии с выдвинутыми требованиями спецификации. В пятом разделе приведены сведения по развертыванию и запуску ПС, указаны требуемые аппаратные и программные средства. Обоснование целесообразность создания программного средства с технико-экономической точки зрения приведено в шестом разделе. Итоги проектирования, конструирования программного средства, а также соответствующие выводы приведены в заключении.

1 АНАЛИЗ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ, ПРОТОТИПОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

Конечный успех программного проекта во многом определяется до начала конструирования: на этапе подготовки, которая проводится с учетом всех особенностей проекта.

Первое предварительное условие, которое нужно выполнить перед конструированием, – ясное формулирование проблемы, которую система должна решать. Общая цель подготовки – снижение риска: адекватное планирование позволяет исключить главные аспекты риска на самых ранних стадиях работы, чтобы основную часть проекта можно было выполнить максимально эффективно.

Главный факторы риска в создании ПО – неудачная выработка требований. Требования подробно описывают, что должна делать программа система. Внимание к требованиям помогает свести к минимуму изменения системы после начала разработки [6].

Перед формулированием требований необходимо изучить ряд вопросов, которые напрямую влияют на все дальнейшие этапы разработки. В частности, необходимо рассмотреть вопросы выбора платформ, архитектуры. По результатам анализа можно будет составить техническое задание к проектируемому программному средству, которое станет основой для составления функциональных требований.

1.1 Аналитический обзор литературных источников

Далее приводится анализ сведений, которые влияют на формулирование требований, выбор архитектуры и дальнейшее проектирование и разработку программного средства.

1.1.1 Кроссплатформенность приложений

Как несколько десятилетий назад, так и в настоящее время выбор платформы является серьезным ограничением для всех последующих этапов разработки. Однако уже начали появляться технологии, которые позволяют использовать однажды написанный код на многих платформах. Сегодня все больше приложений создается сразу для нескольких платформ, а приложения, созданные изначально для одной платформы, активно адаптируются под другие [2]. Разработчик, изучающий какую-либо из таких технологий, получает конкурентное преимущество, поскольку за счет расширения количества платформ расширяется круг задач, над которыми он может работать. Поэтому кроссплатформенность – реальная или потенциальная – является одним из факторов, который необходимо учитывать при выборе технологий реализации проекта.

1.1.2 Обзор целевых платформ

Несмотря на планируемое использование кроссплатформенных технологий, поддержка всех платформ может затребовать значительно больших средств и времени, чем есть в наличии для выполнения дипломного проектирования. Поэтому, необходимо осуществить выбор одной основной платформы с расчетом на продолжение разработки и реализацию проекта для других платформ. Рассмотрим достоинства и недостатки основных.

Настольное приложение – программное средство, которое запускается локально на компьютере пользователя. При его создании появляется возможность использования всех преимуществ аппаратного обеспечения, которым оснащен компьютер, например: прямой доступ к видеокарте, внешним устройствам. Кроме того, появляется возможность взаимодействия с другими установленными приложениями.

Тем не менее у настольных приложений есть и ряд недостатков. Когда пользователь работает удаленно, возникают проблемы, связанные с сетью, соединениями, сетевыми экранами. Один из самых больших недостатков заключается в огромной сложности развертывания приложения на десятки или сотни машин, их конфигурирования, периодического обновления [7].

Веб-приложения, в отличие от настольных, работают на удаленном аппаратном обеспечении и поставляются пользователю через браузер [8]. При их использовании разработчик избавляется от необходимости поддерживать установку большого числа зависимостей, он всегда может быть уверен, что все пользователи используют самую последнюю версию приложения. Вычислительные операции могут производиться на мощном сервере, а результаты вычислений поставляются пользователю – так называемая концепция «тонкого клиента» [9].

Несмотря на то, что ресурсоемкие вычисления производятся на сервере, задержки при передаче, особенно при нестабильном соединении, сама потребность в постоянном интернет соединении могут значительно снизить удобство пользования приложением. Кроме того, размер загружаемого при каждом запуске кода и ресурсов может значительно увеличить траты пользователя, особенно если он использует дорогое мобильное подключение к интернету.

Для мобильных приложений актуальны ограничения платформ, на которых они запускаются, такие как меньшие размеры экранов, более медленные процессоры, ограниченное энергопотребление. Несмотря на это, мобильные устройства часто находятся рядом с пользователями, появляется возможность использования мгновенных оповещений [10]. Вместе с этим, большинство смартфонов оснащено модулями, такими как GPS, камера, NFC, что предоставляет разработчику новые возможности по их использованию.

На основании рассмотренных характеристик различных платформ можно осуществить выбор одной из них, которая и станет целевой для разработки.

1.1.3 Обзор архитектурных стилей

Далее необходимо рассмотреть применяющиеся на практике архитектурные стили, провести их анализ и по результатам осуществить выбор архитектуры, которая затем будет применяться при проектировании программного средства.

Под разработкой *архитектуры* понимают специфицирование структуры всей системы: глобальную организацию и структуру управления, протоколы коммуникации, синхронизации и доступа к данным, распределение функциональности между компонентами системы, физическое размещение, состав системы, масштабируемость и производительность [11]. Набор принципов, используемых в архитектуре, формирует *архитектурный стиль*. Применение архитектурных стилей упрощают решение целого класса абстрактных проблем [12].

При проектировании архитектуры программной системы почти никогда не ограничиваются единственным архитектурным стилем, поскольку они могут предлагать решение каких-либо проблем в различных областях. В таблице 1.1 приведен вариант категоризации архитектурных стилей [13].

Таблица 1.1 – Категоризация архитектурных стилей

Категория	Архитектурный стиль
Связь	SOA (Service-oriented architecture – архитектура, ориентированная на сервисы), Шина сообщений
Развертывание	Клиент-серверный, трехуровневый, N-уровневый
Предметная область	DDD (Domain-driven design – проблемно-ориентированное проектирование)
Структура	Компонентный, объектно-ориентированный, многоуровневый

Сервис-ориентированная архитектура позволяет приложениям предоставлять некоторую функциональность с помощью набора слабосвязанных автономных сервисов; связь между сервисами обеспечивается с помощью заранее определенных контрактов. Данный стиль предоставляет следующие преимущества [13]:

- повторное использование сервисов снижает стоимость разработки;
- автономность и использование формальных контрактов способствует слабой связанности и повышает уровень абстракции;
- сервисы могут использовать возможность автоматического обнаружения и определения интерфейса;
- сервисы и использующие их приложения могут быть развернуты на различных платформах.

Архитектура *шины сообщений* описывает принципы построения систем, которые используют обмен сообщениями как способ связи. Наиболее часто при реализации данной архитектуры используется модель маршрутизатора сообще-

ний или шаблон издатель-подписчик. Главные преимущества использования данного архитектурного стиля [13]:

- расширяемость, которая заключается в возможности добавлять и удалять приложения без влияния на другие;
- снижается сложность приложений, так как единственный интерфейс, который они должны поддерживать – интерфейс общей шины;
- гибкость, которая заключается в возможности подстраиваться под бизнес-требования или желания пользователей через изменения конфигурации или параметров маршрутизации сообщений;
- слабая связанность, поскольку единственное, чем связаны приложения – интерфейс общей шины;
- масштабируемость, которая заключается в возможности в случае необходимости присоединения к шине нескольких экземпляров одного и того же приложения.

Клиент-серверная архитектура описывает распределенную систему, которая включает независимые системы сервера, клиента и соединяющую их сеть. Иногда данную архитектуру называют двухзвенной. Из преимуществ выделяют следующие [12]:

- безопасность: все данные хранятся на сервере, обеспечивающем больший уровень безопасности, чем отдельные клиенты;
- централизованный доступ к данным, который предоставляет возможность более легкого управления, чем в других архитектурах;
- устойчивость и легкость поддержки: роль сервера могут выполнять несколько физических компьютеров, объединённых в сеть; благодаря этому клиент не замечает сбоев или замены отдельного серверного компьютера.

Многоуровневая архитектурный стиль заключается в группировании схожей функциональности приложения по уровням, которые выстроены в вертикальную структуру. Уровни связаны слабо, связь между ними осуществляется по явно установленным протоколам. Строгий вариант архитектуры предполагает, что компоненты какого-либо уровня могут взаимодействовать только с компонентами одного нижележащего уровня; ослабленный вариант разрешает взаимодействие с компонентами любого из нижележащих уровней. Использование данного архитектурного стиля предлагает следующие преимущества [13]:

- есть возможность осуществлять изменения на уровне абстракций;
- изолированность: изменения на каких-либо уровнях не влияют на другие, что снижает риск и минимизирует воздействие на всю систему;
- разделение функциональности помогает управлять зависимостями, что приводит к большей управляемости всего кода;
- независимые уровни предоставляют возможность повторного использования компонентов;

- строго-определенные интерфейсы способствуют повышению тестируемости компонентов.

Многозвенная архитектура предлагает схожее с многоуровневой архитектурой разбиение функциональности, отличие же заключается в предлагающем размещении звеньев на физически обособленной машине [12]. Предлагается использовать данный стиль в случаях, когда компоненты одного звена могут проводить дорогие в ресурсном отношении вычисления, так что это может оказаться на других уровнях, или когда некоторую чувствительную информацию переносят со звеньев уровня представления на уровень бизнес-логики приложения. Далее приведены главные преимущества использования многозвенной архитектуры:

- удобство сопровождения, которое заключается в возможности внесения изменений и обновлений в некоторые компоненты при минимальном влиянии на всё приложение;
- масштабируемость, которая возникает из распределенного развертывания;
- гибкость, которая появляется благодаря предыдущим двум пунктам;
- масштабируемость также приводит к повышению доступности приложения.

Проблемно-ориентированный архитектурный стиль основывается на предметной области, ее элементах, их поведении и связях между ними. Для применениях данного стиля необходимо иметь хорошее понимание предметной области или людей, которые бы смогли объяснить ее специфику разработчикам. Первое, что нужно сделать при принятии данного стиля – выработать единый язык, который бы знала вся команда разработчиков, который бы был избавлен от технических жаргонизмов и содержал только термины предметной области – только так можно избежать проблемы непонимания между участниками [14]. Преимущества, которые может предложить данный стиль:

- упрощение коммуникации между участниками процесса разработки благодаря выработке единого языка;
- модель предметной области обычно является расширяемой и гибкой при изменениях условий и бизнес-требований;
- хорошая тестируемость.

Компонентная архитектура основывается на декомпозиции системы в отдельные функциональные или логические компоненты, которые раскрывают другим компонентам только заранее определенные интерфейсы [13]. Преимущества данного подхода:

- легкость развертывания, которая заключается в обновлении компонентов без влияния на другие;
- использование компонентов сторонних разработчиков позволяет снизить расходы на разработку и поддержку;

- поддержка компонентами заранее определенных интерфейсов позволяет упростить разработку;
- возможность переиспользования компонентов также оказывает влияние на снижение стоимости.

Объектно-ориентированный архитектурный стиль выражается в разделении функциональности системы на множество автономных объектов, каждый из которых содержит некоторые данные и набор методов поведения, свойственных объекту. Обычной практикой является определение классов, соответствующих объектам предметной области. Применение данного стиля предполагает следующие преимущества [13]:

- соотнесение классов программы и объектов реального мира делает программное средство более понятным;
- полиморфизм и принцип абстракции предоставляет возможность повторного их использования;
- инкапсуляция повышает тестируемость объектов;
- улучшение расширяемости, которое возникает благодаря тому, что изменения в представлении данных не оказывают влияния на внешний интерфейс объекта, что не ограничивает его способность взаимодействовать с другими объектами;
- высокая сцепленность объектов, которая достигается использованием разных объектов для разного набора действий.

Таким образом, применение рассмотренных архитектур на этапе проектирования окажет большое влияние на успешность всего процесса разработки; какие бы стили не были применены, можно быть уверенным в правильности выбора за счет того, что у всех из них есть определенные и зачастую разные преимущества.

1.1.4 Проектирование баз данных

В настоящее время сложно представить сложные приложения, которые бы не использовали специальные средства для хранения информации.

База данных – представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью ЭВМ. Модель базы данных – описание базы данных с помощью определенного (в т.ч. графического) языка на некотором уровне абстракции.

Основные задачи проектирования баз данных:

- Обеспечение хранения в БД всей необходимой информации.
 - Обеспечение возможности получения данных по всем необходимым запросам.
 - Сокращение избыточности и дублирования данных.
 - Обеспечение целостности базы данных.
- Выделяют следующие уровни моделирования [15]:

- инфологический уровень: описание предметной области без привязок к каким-либо средствам реализации: языкам программирования, СУБД, и т.д;
- даталогический уровень: модель предметной области в привязке к средствам реализации;
- физический уровень: описывает конкретные таблицы, связи, индексы, методы хранения, настройки производительности, безопасности и т.д.

При ошибках моделирования могут возникать аномалии операций с БД. Аномалия – противоречие между моделью предметной области и моделью данных, поддерживаемой средствами конкретной СУБД [15].

Выделяют следующие виды аномалий:

- Аномалия вставки: при добавлении данных, часть которых у нас отсутствует, мы вынуждены или не выполнять добавление или подставлять пустые или фиктивные данные.
- Аномалия обновления – при обновлении данных мы вынуждены обновлять много строк и рискуем часть строк «забыть обновить».
- Аномалия удаления – при удалении части данных мы теряем другую часть, которую не надо было удалять.

Для устранения аномалий существует процесс нормализации БД.

Нормализация – группировка и/или распределение атрибутов по отношениям с целью устранения аномалий операций с БД, обеспечения целостности данных и оптимизации модели БД.

Отношение находится в первой нормальной форме, если все его атрибуты являются атомарными. Атрибут считается атомарным, если в предметной области не существует операции, для выполнения которой понадобилось бы извлечь часть атрибута.

Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме, и при этом любой атрибут, не входящий в состав ПК, функционально полно зависит от ПК.

Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме, и при этом любой его неключевой атрибут нетранзитивно зависит от первичного ключа [15].

Как правило, третья нормальная форма принимается достаточной и, поскольку дальнейшая нормализация может породить избыточность, то она обычно не проводится.

1.2 Обзор существующих аналогов

Для решения задач управления временем, задачами и контактами могут использоваться различные средства. Рассмотрим каждую из задач в отдельности.

1.2.1 Управление временем и расписаниями

Для участников учебного процесса управление временем заключается в следовании расписанию, составляемое деканатами или диспетчерской службой университетов. Еще тридцать лет назад расписание представляло собой таблицу на огромном листе бумаги, которую составляли вручную и вывешивали на доске объявлений. Чтобы узнать, какое занятие будет следующим, приходилось или заранее переписывать своё расписание всем студентам, или каждый раз идти к доске объявлений (что имеет свои преимущества, так как тогда есть возможность узнать обо всех изменениях в расписании).

С течением времени от ручного заполнения перешли к составлению электронных таблиц в специальных программных средствах. Тем не менее, облегчено было только составление расписания, а не пользование им: электронные таблицы печатались и по-прежнему вывешивались около деканатов. Потребовалось еще несколько лет и прогресс в развитии информационных технологий, чтобы студенты и преподаватели получили возможность получения расписания через сеть интернет без необходимости посещения университета.

В настоящее время на постсоветском пространстве существует огромное число высших учебных заведений. Некоторое время назад процессы, происходящие в таких учреждениях не были оцифрованы и автоматизированы вовсе. Лишь недавно начали появляться информационные системы, решающие те или иные задачи. Однако, каждый ВУЗ занимается этой проблемой самостоятельно и решает ее, насколько хватает квалифицированных специалистов, ресурсов и, самое главное, желания внедрять что-то инновационное со стороны руководства и участников учебного процесса этих ВУЗов.

Универсальной шкалы, позволяющей оценить удобство следования по всем деятельностим учебного процесса в разных ВУЗах, нет. Однако, можно попытаться это оценить по некоторым вторичным признакам, таким как, например, соответствие сайта ВУЗа современным представлениям об удобстве пользования (можно легко отличить сайты, сделанные за последние несколько лет, от сайтов, сделанных более десяти лет назад: со временем меняются и представления об удобстве пользования, и технологии, позволяющие его достичь), отзывах тех людей, которые участвуют в процессе обучения (то есть студентов, преподавателей) и некоторым другим. Рассмотрим возможности, предоставляемые информационными системами различных белорусских ВУЗов по состоянию на апрель 2017 года.

Белорусский государственный университет информатики как ведущий специализированный университет в области информационных технологий на постсоветском пространстве обладает достаточным числом квалифицированных специалистов (поскольку обучает их), чтобы автоматизировать многие части учебного процесса. Для пользования предоставляется система расписания: занятий и экзаменов для студентов и преподавателей, причём предоставляется

возможность не только просматривать расписание, находясь на сайте (пример расписания приведен на рисунке 1.1), но реализован API для получения актуальных данных о преподавателях, группах и расписании для них в формате XML. Помимо этого, наиболее близкий к информационным технологиям факультет компьютерных систем и сетей (КСиС) имеет в своём распоряжении портал факультета, на котором присутствует актуальная информация о событиях, происходящих на факультете, куда выкладываются расписания учебного процесса, списки групп, рейтинговые списки. Однако, по-прежнему существует и ряд проблем. Кроме факультета КСиС, своих порталов не имеет ни один из других факультетов, поэтому вся коммуникация со студентами может проходить только очно, а некоторые неформальные виды коммуникации – через группы факультетов в социальных сетях. Кафедры не имеют удобной для студентов доски объявлений (строго говоря, такие доски есть, но она находится настолько глубоко внутри сайта университета, что большая часть студентов о ней и не знает). Для того, чтобы внести любое малейшее изменение в расписание, преподавателю нужно обращаться в диспетчерскую службу: нет возможности сделать это автоматически. Но несмотря на все недостатки, БГУИР все равно обгоняет многие ВУЗы по степени информатизации и автоматизации учебного процесса.

Сайт ведущего университета страны общего профиля – Белорусского государственного университета – был значительно переработан около двух лет назад (то есть около февраля 2015 г.), что привело к значительному повышению удобства пользования и актуальности информации. Еще одним достоинством информационной системы БГУ является то, что для всех факультетов реализованы отдельные порталы. Однако, очень большим недостатком является то, что в этом университете не реализована единая система управления расписанием. Все факультеты решают задачу составления расписания и доставки его студентам и преподавателям по-своему. Есть факультеты, которые предоставляют более-менее удобные таблицы с расписанием (например, на рисунке 1.2 приведен пример расписания первого курса ФПМИ), есть факультеты, которые предоставляют мобильное приложение с расписанием (но только с расписанием этого факультета), однако, есть факультеты, которые предоставляют расписание в виде неизменяемого и недоступного для автоматизированного разбора документа (например, в формате PDF). Более того, некоторые кафедры вообще не выкладывают своё расписание на сайт факультета, и чтобы студенты и преподаватели могли с ним ознакомиться, им нужно приходить к доске объявлений своей кафедры. В итоге, нет ни программного интерфейса для доступа к расписанию, ни даже унифицированной системы расписания, что значительно затрудняет создание любых приложений для работы с ним. И эту проблему в данном ВУЗе можно решить только одним способом: предложить систему, форматы данных и программные интерфейсы и каким-либо образом убедить

Расписание занятий для студентов БГУИР

на весенний семестр 2016 - 2017 гг.

Сейчас 3-я учебная неделя

К расписанию преподавателей

[Поиск по группе](#) [Расширенный поиск](#)

Группы Поиск

Расписание для группы 351005

Занятия Сессия

■ - пары, которые проходят на текущей учебной неделе ■ - пары, которые проходят по другим учебным неделям

🖨️ [Печать](#)

Недели	Время	Дисциплина	Подгр.	Ауд.	Преподаватель	Примечание
Понедельник						
1, 3	13:25-15:00	ПОЦП (ЛР)		3226-5	Базылев Е. Н.	
1, 3	15:20-16:55	ПОВС (ЛР)		304-1	Герасимович В. Ю.	
2	18:45-22:00	ОегрС (ЛР)	1	2136-4	Петюкович Н. С.	
4	18:45-22:00	ОегрС (ЛР)	2	2136-4	Петюкович Н. С.	
Вторник						
	15:20-18:40	ПОЦП (ЛК)		409-1	Иванюк А. А.	
	18:45-20:20	ПОВС (ЛК)		409-1	Петровский Н. А.	
Среда						
1, 3	15:20-16:55	ПОЦП (ЛР)	1	216-4	Базылев Е. Н.	
1, 3	15:20-16:55	ОегрС (ЛР)	2	318-5	Петюкович Н. С.	
	17:05-18:40	ПОЦП (ЛК)		106-4	Иванюк А. А.	
	18:45-22:00	ПОВС (ЛК)		106-4	Петровский Н. А.	
2, 4	15:20-16:55	ОегрС (ЛР)	1	318-5	Петюкович Н. С.	
2, 4	15:20-16:55	ПОЦП (ЛР)	2	216-4	Базылев Е. Н.	
Четверг						
1, 3	13:25-15:00	ПОЦП (ЛР)	1	2136-4	Базылев Е. Н.	
1, 3	15:20-16:55	ПОВС (ЛР)	1	304-1	Герасимович В. Ю.	
2, 4	13:25-15:00	ПОЦП (ЛР)	2	2136-4	Базылев Е. Н.	
2, 4	15:20-16:55	ПОВС (ЛР)	2	304-1	Герасимович В. Ю.	
Пятница						
1, 3	15:20-18:40	ОегрС (ЛР)		2136-4	Петюкович Н. С.	
1, 3	18:45-20:20	ПОЦП (ЛР)		2136-4	Базылев Е. Н.	
Суббота						
	08:00-09:35	ОегрС (ЛК)		104-4	Мелких Е. Г.	
1, 3	09:45-11:20	ОегрС (ЛК)		104-4	Мелких Е. Г.	
2, 4	09:45-11:20	ПОВС (ЛК)		104-4	Петровский Н. А.	

Рисунок 1.1 – Пример расписания БГУИР

их использовать. Но процесс информатизации не стоит на месте, и стоит ожидать, что уже скоро такая унификация произойдёт.

Попытка нивелировать неудобства пользования официальными источниками расписания может состоять в использовании специализированных сервисов-календарей. Как и все веб-приложения, они доступны на любом компьютере, подключенному к сети интернет, кроме того, зачастую разработаны официальные приложения для мобильных устройств. Еще одним достоинством является универсальность, заключающаяся в том, что данные сервисы можно использовать не только для управления занятиями университета. Главным же недостатком является необходимость вручную заполнять все расписание (что, однако, даёт возможность самому настроить форму его отображения). Таким образом, на конфигурирование таких приложений может уходить несколько часов пару раз в год. Пример расписания, сформированного с помощью веб-сервиса Google Calendar приведен на рисунке 1.3.

Анализ существующих информационных систем показывает наличие разрозненности, отсутствие единого протокола обмена и хранения данных учебного процесса, что подтверждает актуальность разработки программного обеспечения, направленного на автоматизацию данной области.

1.2.2 Управление задачами

Управление задачами заключается в их учете и ранжированию по приоритету. Для этого могут использоваться различные записные книжки и календари.

Самым первым средством стали обычные бумажные записные книжки. И по сей день есть большое число их сторонников, для которых оказывается намного более удобным записать какие-то идеи на бумаге, чем в приложении на мобильном устройстве или компьютере [16]. Из преимуществ выделяют следующие:

- доступность, которая заключается в привычке постоянно держать блокнот около себя;
- надежность, которая обусловлена независимостью от сети интернет и источников питания;
- визуализация и фиксация мыслей на бумаге часто помогает лучше обдумать задачу;
- универсальность: блокнот можно использовать и для записей, и для рисования, и для ведения адресной книги.

Тем не менее, следующие существенные недостатки данного способа влияют на выбор людей в пользу автоматизированных сервисов:

- трудность внесения информации;
- высокая стоимость на единицу информации;
- сложность организации быстрого поиска.

Следующей альтернативой является использование приложений, эмули-

[Expand All](#) [Collapse All](#)



Mon Oct 24, 2016	09:00 – 11:00	⊕ БД - ерам
	15:20 – 16:55	⊕ ОБИПВИТ - 202-3
	17:05 – 18:40	⊕ АКГ - 202-3
Tue Oct 25, 2016	17:05 – 18:40	⊕ ПиРИС - 04-4
	18:45 – 20:20	⊕ БД - 04-4
	20:25 – 22:00	⊕ ОБИПВИТ - 311-4
Wed Oct 26, 2016	13:25 – 15:00	⊕ АКГ - 304-1
	15:20 – 16:55	⊕ САиММод - 04-4
	17:05 – 18:40	⊕ ЦОС - 213-5
	18:45 – 20:20	⊕ ЦОС - 213-5
Thu Oct 27, 2016	13:25 – 15:00	⊕ ПиРИС - 224-5
	15:20 – 16:55	⊕ ЦОС - 224-5
	17:05 – 18:40	⊕ СМпИМЭ - 209-4
	18:45 – 20:20	⊕ ЦОС - 209-4
Sat Oct 29, 2016	09:45 – 10:45	⊕ ПиРИС
Mon Oct 31, 2016	15:20 – 16:55	⊕ ОБИПВИТ - 202-3
	17:05 – 18:40	⊕ АКГ - 202-3

Рисунок 1.3 – Пример расписания, сформированного с помощью сервиса Google Calendar

рующих записные книжки на компьютерах и мобильных устройствах. Таким приложений создано большое количество, многие из них обладают некоторыми отличительными чертами. Например, существуют приложения, которые позволяют приклеивать заметки к некоторым областям экрана (так называемые sticky notes). Зачастую приложения записных книг предлагают удобные способы разграничения заметок по цели их создания, например: заметки для работы, списки покупок и так далее. Кроме того, большинство приложений предоставляет возможность прикрепления различных видов информации к заметке: ав-

тосоздаваемые списки, таблицы, изображения, фотографии с камеры, звуковые заметки.

1.2.3 Управление контактами

Задача сохранения контактов актуальна и для студентов, поскольку они за время учебы взаимодействуют с несколькими десятками преподавателей, и, в свою очередь, для преподавателей, поскольку только за один семестр они сталкиваются с несколькими сотнями студентов. Проблема обеспечения двусторонней связи между ними актуальна в связи с существованием следующих задач:

- проведение неформальных дистанционных консультаций;
- распространение индивидуальных заданий;
- распространение информации об изменениях в расписании.

В настоящее время для связи и сохранения контактов используются электронная почта и социальные сети. Электронная почта является стандартным средством коммуникации; практически все имеют хотя бы один зарегистрированный email. Тем не менее, слабая поддержка интерактивности и малый охват студентов сообщениями являются значительными недостатками данного средства. Частично данные недостатки решаются с помощью социальных сетей: там есть возможность создания круга контактов, есть возможность обмена мгновенными сообщениями и оповещения об изменениях большого количества людей. Однако, многие преподаватели уклоняются от пользования ими, по большей части из-за их развлекательной направленности. Чтобы исключить данный недостаток предлагается реализация социальной сети, ориентированной и специализированной на учебном процессе.

1.2.4 Обзор некоторых средств дистанционного обучения

Рассмотренные аспекты типичны для учебного процесса. При анализе возможностей их информатизации может возникнуть желание полностью оцифровать учебный процесс и осуществить полный переход к дистанционному обучению (или, по крайней мере, внедрить его наряду с другими формами обучения: очным и заочным).

Дистанционная форма обучения в настоящее время осваивается различными университетами нашей страны, например, БГУИР, БГУ. Кроме того, существует большое число онлайн-ресурсов, предоставляющих доступ к обучающим курсам; наиболее известными из них являются Stanford Online, Coursera, Udacity, Khan Academy. Часто их объединяют под названием массовые открытые онлайн-курсы (МООС – massive open online courses [17]).

В настоящее время практически любое учебное заведение может создать свой онлайн-ресурс дистанционного обучения. Для этого может применяться одна из специальных программных систем, наиболее известной и распространенной из которых является свободная система управления курсами Moodle.

Данная система разрабатывается сотрудниками специального фонда, который финансируется партнерами системы и грантами. Основная функциональность доступна в стандартной поставке, также есть возможность установки сторонних модулей (необязательно бесплатных). На рисунках 1.4а, 1.4б, 1.4в приведены примеры экранов развернутой и функционирующей системы Moodle.

Система Moodle разрабатывается с 2002 года [18], она обладает достаточной стабильностью, кроме того, процесс её установки автоматизирован с помощью специального мастера. Тем не менее, если сравнивать предложенные в ней концепции с проектируемой в данном дипломном проекте программной системой, то существенными оказываются следующие вопросы:

- Система Moodle является решением для дистанционного обучения. Конечно, ее применение возможно и для других форм, однако это не является целью её разработки, а побочной возможностью.

- Данная система требует нового развертывания и инстанцирования у каждого нового заказчика, то есть предполагается архитектурный стиль множественности экземпляров программного средства (*multiinstance*). При его применении требуется наличие специалистов поддержки у каждого заказчика. Кроме того, большое число экземпляров системы лишает возможности экономии вычислительных мощностей и средств хранения. Для решения данной проблемы планируется использование мультиарендного архитектурного стиля (*multitenancy*), когда создается единый экземпляр приложения, который обслуживает множество заказчиков (в нашем случае – множество ВУЗов).

- Сложность пользования системой в связи с запутанностью интерфейса. Разработчики реализовали достаточно большой набор функциональности, который, однако, не обладает свойством удобства пользования. О данном факте свидетельствуют многочисленные отзывы студентов и преподавателей университетов, в которых была внедрена данная система [18].

Все озвученные системы дистанционного обучения относятся к предметной области обучения студентов. Значит, и их проектирование и разработка начиналась и отталкивалась от анализа очной формы обучения. Таким образом решалась задача переноса взаимодействия участников учебного процесса в онлайн-среду. Однако целью данного дипломного проекта является не разработка системы для дистанционного обучения, а привнесение их положительных средств и свойств в очное образование с единственной целью – облегчения некоторых задач участников учебного процесса.

1.3 Требования к проектируемому программному средству

По результатам изучения предметной области, анализа литературных источников и обзора существующих систем-аналогов сформулируем требования к проектируемому программному средству.

а) домашняя страница системы Moodle

б) экран календаря Moodle

в) экран домашней страницы курса Moodle

Рисунок 1.4 – Примеры экранов системы Moodle

1.3.1 Назначение проекта

Назначением проекта является разработка программного средства, автоматизирующего основные задачи участников учебного процесса университетов: управление расписанием, заданиями и коммуникацией.

1.3.2 Основные функции

Программное средство должно поддерживать следующие основные функции:

- регистрация и аутентификация;
- поддержка системы ролей;
- отображение расписания занятий;
- отображение списка изучаемых дисциплин (для студентов) и преподаваемых дисциплин с типами занятий (для преподавателей);
- возможность управления индивидуальными заданиями и материалами по дисциплинам;
- отправка результатов выполнения индивидуальных заданий;
- оценивание/отклонение результатов выполнения заданий;
- управление очередями на защиту индивидуальных заданий;
- отправка сообщений другим пользователям системы;
- управление списками групп, их отображение со всеми выставленными оценками;
- возможность проверки посещения занятий.

1.3.3 Требования к входным данным

Входные данные для программного средства должны быть представлены в виде вводимого пользователем с помощью клавиатуры текста и выбора доступных опций пользовательского интерфейса.

Должны быть реализованы проверки вводимых данных на корректность с отображением информации об ошибках в случае их некорректности.

1.3.4 Требования к выходным данным

Выходные данные программного средства должны быть представлены посредством отображения информации с помощью различных элементов пользовательского интерфейса.

1.3.5 Требования к временным характеристикам

Производительность программно-аппаратного комплекса должна обеспечивать следующие временные характеристики: время реакции не запроса пользователя не должно превышать 1 секунды при минимальной скорости соединения 1 Мбит/с. Допускается невыполнение данного требования в случае, когда невозможность обеспечить заявленную производительность обусловлена объективными внешними причинами.

1.3.6 Требования к надежности

Надежное функционирование программы должно быть обеспечено выполнением следующих организационно-технических мероприятий:

- организация бесперебойного питания;
- выполнение рекомендаций Министерства труда и социальной защиты РБ, изложенных в Постановлении от 23 марта 2011 г. «Об утверждении Норм времени на работы по обслуживанию персональных электронно-вычислительных машин, организационной техники и офисного оборудования»;
- выполнение требований ГОСТ 31078-2002 «Защита информации. Испытания программных средств на наличие компьютерных вирусов»;
- необходимым уровнем квалификации пользователей.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), нефатальным сбоем операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств. Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

1.3.7 Требования к аппаратному обеспечению серверной части

ЭВМ, на которой должна функционировать серверная часть программного средства, должна обладать следующими минимальными характеристиками:

- процессор Intel Core i5 с тактовой частотой 2 ГГц;
- жесткий диск объемом 100 Гб;
- оперативная память 4 Гб;
- сетевая карта Ethernet 100 Мбит/с.

Также для функционирования серверной части требуется установленный Apache HTTP Server, который является кроссплатформенным программные средством, вследствие чего вопрос о целевой операционной системе не рассматривается. Кроме того, процедуры установки и настройки данного веб-сервера выходят за рамки данного проекта и также не рассматриваются.

1.3.8 Требования к аппаратному обеспечению клиентской части

Клиентская часть программного средства должна функционировать на ЭВМ со следующими минимальными характеристиками:

- процессор Intel Pentium 4 с тактовой частотой 2 ГГц и более;
- оперативная память 2 Гб и более;
- сетевая карта Ethernet 10/100 Мбит.

Для корректной работы программного средства необходим один из следующих браузеров с соответствующей минимальной версией:

- Google Chrome 49;
- Vivaldi 1.0;
- Opera 34;
- Mozilla Firefox 43;
- Apple Safari 9.0;
- Microsoft Edge 20.

1.3.9 Выбор технологий программирования

Язык программирования, на котором будет реализована система, заслуживает большого внимания, так как вы будете погружены в него с начала конструирования программы до самого конца. Исследования показали, что выбор языка программирования несколькими способами влияет на производительность труда программистов и качество создаваемого ими кода. Если язык хорошо знаком программистам, они работают более производительно. Данные, полученные при помощи модели оценки Сосомо II, показывают, что программисты, использующие язык, с которым они работали три года или более, примерно на 30% более продуктивны, чем программисты, обладающие аналогичным опытом, но для которых язык является новым [19]. В более раннем исследовании, проведенном в IBM, было обнаружено, что программисты, обладающие богатым опытом использования языка программирования, были более чем втрое производительнее программистов, имеющих минимальный опыт [20].

Язык программирования TypeScript, указанный в задании на дипломное проектирование, является кроссплатформенным языком программирования. Данный ЯП представляет собой надмножество языка JavaScript, что означает, что их объединяют общие синтаксис и семантика управляющих конструкций; ключевой отличительной особенностью является возможность использования строгой типизации, что значительно упрощает статические проверки кода. Кроме того, он компилируется в обычный JavaScript, что означает возможность запуска кода в любом браузере или движке, который поддерживает стандарт ECMAScript 3 или более новый [21].

Несмотря на то, что TypeScript, так же как и JavaScript, изначально предназначался для запуска в браузере клиента, в настоящее время разработаны фреймворки, позволяющие использовать его для разработки под различными платформами: Windows 10 [22], iOS, Android [23] и другие.

Исходя из достоинств данного языка программирования, можно сделать вывод, что он наиболее подходящий для решения проблем, схожих с поднимаемыми в данной пояснительной записке. Именно поэтому TypeScript и выбран

как основной язык программирования в задании к текущему дипломному проекту.

Однако, выбранный язык программирования является средством для программирования клиентской части приложения. Поскольку для приложения в любом случае понадобится база данных, то есть два варианта:

- осуществлять запросы к БД напрямую с клиентского приложения;
- реализовать серверную прослойку между клиентской частью и базой данных.

Первый подход крайне небезопасен: очень опасно предоставлять открытый доступ к БД. В это же время, второй способ, помимо ее сокрытия, предоставляет возможность проверки подлинности и предоставления прав пользователям. В связи с этим появляется проблема выбора технологий для серверной части. Основным влияющим фактором является имеющийся опыт команды разработки, в связи с чем была выбрана технология .Net и язык программирования C#.

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Она была призвана решить ряд наболевших проблем в мире разработки ПО, скопившихся на момент ее выхода, что отражено в целях, которые были поставлены в ходе ее разработке.

При разработке платформы .NET учитывались следующие цели [24]:

- Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.
- Обеспечение среды выполнения кода, минимизирующей конфликты при развертывании программного обеспечения и управлении версиями.
- Обеспечение среды выполнения кода, гарантирующей безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем.
- Обеспечение среды выполнения кода, исключающей проблемы с производительностью сред выполнения сценариев или интерпретируемого кода.
- Обеспечение единых принципов работы разработчиков для разных типов приложений, таких как приложения Windows и веб-приложения.
- Разработка взаимодействия на основе промышленных стандартов, которое обеспечит интеграцию кода платформы .NET Framework с любым другим кодом.

Несмотря на то, что платформа .Net поддерживает несколько языков программирования, основным является язык C#. Он является простым, современным, объектно-ориентированным, обеспечивающим безопасность типов языком программирования. Он соответствует международному стандарту Европейской ассоциации производителей компьютеров — стандарт ECMA-334, а

также стандарту Международной организации по стандартизации (International Standards Organization, ISO) и Международной электротехнической комиссии — стандарт ISO/IEC 23270. Компилятор Microsoft C# для .NET Framework согласуется с обоими этими стандартами [25].

Одной из сред программирования, которая поддерживает одновременно и C#, и TypeScript, является Microsoft Visual Studio, которая входит в линейку продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Она включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Visual Studio позволяет создавать и подключать сторонние дополнения для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов или инструментов для прочих аспектов процесса разработки программного обеспечения. Именно поэтому она и выбрана в качестве основной среды программирования.

Язык программирования TypeScript можно использовать для создания приложений для различных платформ. Для проектируемого программного средства актуальны следующие характеристики:

- нет необходимости в организации ресурсоемких вычислений;
- желательна возможность использования мгновенных уведомлений и оповещений;
- желательна доступность приложения на различных устройствах.

По результатам обзора возможных платформ, представленных в пункте 1.1.2, было принято решение выбрать основной для разработки платформу веб-приложений. После завершения разработки первой версии программного средства будет рассматриваться вопрос разработки мобильного приложения.

Фактор опыта использования оказал влияние на выбор системы управления базами данных для разрабатываемого приложения. СУБД N*DB является особенно приспособленной для веб-приложений. Ее отличительной особенностью является возможность масштабируемости, а также отказоустойчивость. Основным способом взаимодействия с данной СУБД является предложенная разработчиком клиентская библиотека на языке C#.

Сформулированные требования позволят осуществить успешное проектирование и разработку программного средства.

2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Функциональная модель программного средства

Функциональная модель программного средства представлена в виде схемы алгоритма процесса обучения и диаграммы вариантов использования и информационной модели предметной области. Варианты использования отражают функциональность системы в ответ на внешние воздействия с точки зрения получения значимого результата для пользователей. Информационная модель предметной области в дальнейшем будет использоваться при проектировании базы данных для программного средства.

2.1.1 Анализ процесса образования

Перед началом проектирования необходимо проанализировать процесса образования. В связи с отсутствием опыта преподавания у исполнителя данного дипломного проекта выглядит целесообразным проведение данного анализа с точки зрения студента. Результат анализа представлен в виде схемы алгоритма на рисунках 2.1 и 2.1. Схема ограничивается процессом обучения в семестре и не включает проверку знаний в виде зачетов и экзаменов, завершающих обучение по дисциплине.

Одной из особенностей данной схемы является цикличность, с которой выполняется процесс обучения. Цикл начинается изучением некоторой теории по дисциплине, затем на практическом или лабораторном занятии преподаватель разъясняет некоторые сложные моменты, которые могут возникнуть при выполнении индивидуальных заданий, выдаёт условие заданий, а также вариант. Студент на занятии или дома выполняет задание. После этого на одном из занятий или вне их происходит защита результатов выполнения заданий. После этого происходит оценивание результатов преподавателем. Если задание выполнено не полностью, то оно отправляется на доработку. Если же оценка не является удовлетворительной, то преподаватель вправе выдать новое задание.

Таким образом на схеме алгоритма приведен типичный вариант обучения студента, который широко практикуется в университетах и практически никак не автоматизирован. Программное средство, разработка которой ведется в данном дипломном проекте, не изменит устоявшейся системы сразу, однако значительно упростит выполнение многих ее этапов.

2.1.2 Варианты использования программного средства

По результатам анализа предметной области и существующих аналогов можно сделать вывод, что проектируемое программное средство должно поддерживать ряд функций для уменьшения так называемых накладных расходов процесса обучения, ключевыми из которых являются следующие:

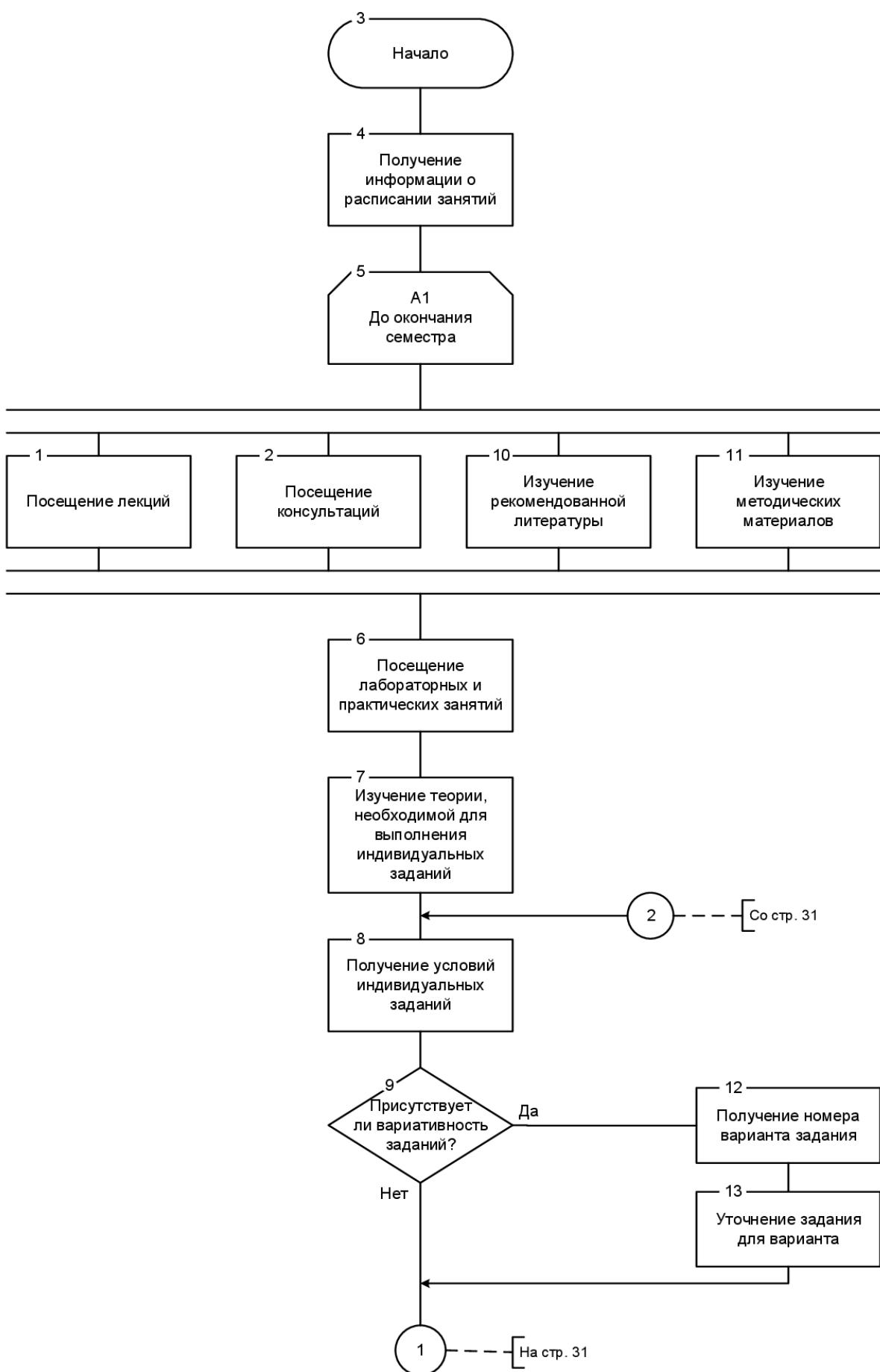
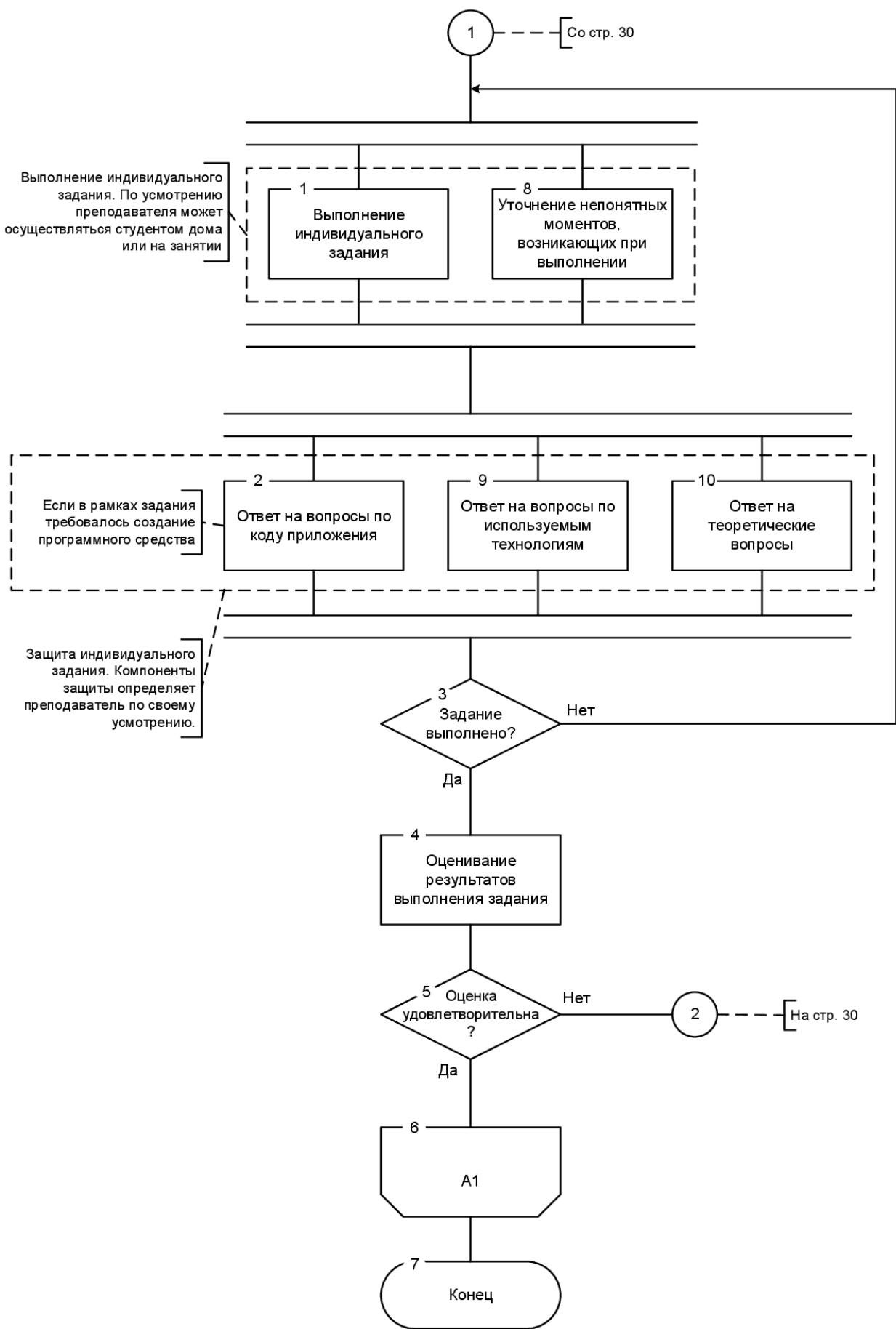


Рисунок 2.1 – Схема алгоритма процесса образования с точки зрения студента



– *Система ролей*, которая позволяет поддерживать много факультетов и университетов в единой системе.

– *Расписание занятий*. Загрузка расписания с использованием внешнего API университетов (при наличии) и его отображение в виде последовательного списка с разбиением по дням позволит студентам и преподавателям быстрее и легче понимать, какие занятия запланированы на каждый из дней.

– *Индивидуальные задание*. Возможность использования единой системы значительно упрощает распространение заданий и для студентов, и для преподавателей.

– *Результаты заданий*. Возможность отправить выполненное задание без необходимости посещения университета особенно актуально для работающих студентов.

– *Управление списками* студентов, которое упрощает для преподавателей возможность проверок посещения занятий, а также слежение за прогрессом выполнения индивидуальных заданий студентами.

– *Прогресс выполнения* заданий актуален также и для студента; удобство контроля значительно повышается при использовании наглядного интерфейса.

– *Учебные материалы*. Их распространение является такой же проблемой, как и распространение индивидуальных заданий.

– *Виртуальная зачетка*. Для преподавателя часто бывает очень важно сформировать образ студента на основании его истории обучения.

– *Сообщения* между преподавателями и студентами позволяют проводить удаленные консультации, а также помогают формировать круг профессиональных контактов.

Диаграмма вариантов использования, разработанная с использованием нотации UML 2.1, представлена на рисунке 2.2.

Рассмотрим подробно представленные на рисунке прецеденты.

Регистрация, аутентификация и авторизация – функции, которые доступны для роли «Гость» (пользователь, не зарегистрированный в системе). В первой версии приложения планируется реализация собственной системы авторизации; в дальнейшем будет добавлена возможность регистрации с помощью внешних поставщиков данных (Google, VK, Facebook и др.).

Для незарегистрированного пользователя также будет предоставляться доступ к просмотру некоторой информации *структуре университета и преподавателей*. Однако, такие данные, как, например, материалы для занятий, условия индивидуальных заданий, будут доступны только для авторизованных пользователей.

После регистрации пользователь получает доступ к *поиску пользователей*, системе *сообщений*. Однако, для участия в учебном процессе необходимо *подтверждение роли*. Только после подтверждения преподаватели получают доступ к предметам, которые они преподают, а студенты зачисляются в групп-

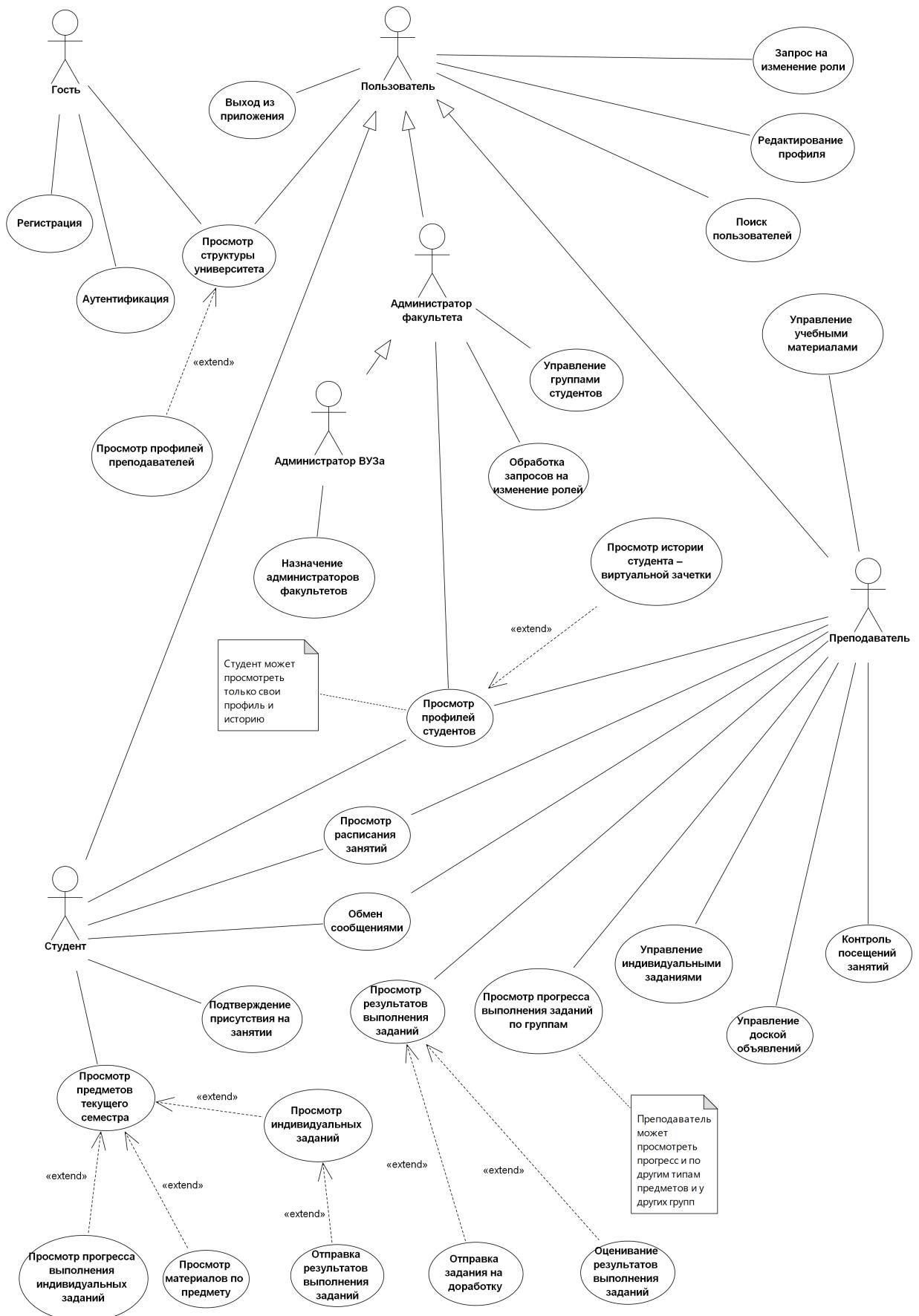


Рисунок 2.2 – Диаграмма вариантов использования ПС

пы.

Для управления группами студентов и подтверждения ролей в системе предусматривается роль «Администратор факультета». В дальнейшем избавление от данной роли и децентрализация ее функций следующим образом: студенты и преподаватели при подаче заявки указывают свой факультет. Данные заявки рассматриваются не администраторами, а пользователями с ролью «Преподаватель». Решение о подтверждение роли выносится на основании порогового принципа: заявка утверждается при ее подтверждении несколькими пользователями.

Для утверждения администраторов факультетов планируется создание роли «Администратор ВУЗа». Предполагается, что на данную роль будут назначаться пользователи, с которых начинается внедрение системы в очередном университете (так называемые «нулевые пользователи»).

Основная функциональность системы предусматривается для непосредственных участников учебного процесса: студентов и преподавателей.

Одной из самых главных функций, разделяемых всеми подтвержденными пользователями, является *просмотр расписания занятий*. Как было указано в пункте 1.2.1, расписания часто представляются в виде сложных для понимания таблиц. Представление расписания на каждый день в виде списка является намного более удобным способом. Таким образом, суть данной функции заключается в «разворачивании» расписания с целью ответить на вопрос «какие занятия предстоят сегодня?»

Функция коммуникации реализуется с помощью *обмена сообщениями*. В приложении к учебному процессу она реализуется в виде дистанционных консультаций. Кроме того, она способствует формированию профессиональных контактов между студентами и преподавателями.

Часто для формирования впечатления о добросовестности студентов преподаватели осуществляют *контроль посещений занятий*. Затем эти данные могут учитываться на экзамене, зачете или других формах проверки знаний.

Основная вид деятельности студентов в университете – выполнение индивидуальных заданий. Для этого проектируются следующие функции:

- управление индивидуальными заданиями, что подразумевает создание условий этих заданий преподавателями;
- просмотр заданий студентом для их выполнения;
- отправка результатов выполнения заданий, что означает возможность дистанционной защиты (в случае, когда преподаватель не ставит условия очной проверки знаний во время защиты);
- просмотр результатов выполнения заданий, результатом которого становится либо *оценивание результатов*, либо *отправка на доработку*.

Студенты во время своей учебы только за один семестр могут выполнять десятки заданий. Тогда представляют интерес следующие функции: *просмотр*

прогресса выполнения заданий студентом в разрезе изучаемых им предметов и преподавателем в разрезе обучаемых групп. Данная информация необходима для недопущения отставания от учебных планов и по ее результатам будет осуществляться либо самоконтроль студентами, либо внешний контроль преподавателями.

Практически для любого курса реализовано большое количество учебно-методических комплексов, существует большое число книг, рекомендуемых в рамках изучения того или иного курса. Именно с этой целью планируются функции *управления учебными материалами и просмотр материалов*.

По результатам изучения учебных курсов студент как правило получает оценку. Данные оценки в настоящее время (2017 год) дублируются в двух местах: ведомость деканата и зачетная книжка студента. Функция *виртуальной зачетки* предлагается с целью повышения прозрачности хранения оценок; кроме того, ее успешное внедрение может впоследствии привести к отказу от перечисленных примитивных средств учета.

2.1.3 Разработка инфологической модели базы данных

Исходя из необходимости использования в проектируемом приложении базы данных, разработаем ее инфологическую модель. Для ее создания будем использовать расширение диаграммы классов UML 2.1, предназначенное для моделирования баз данных. Полученная диаграмма (рисунок 2.3) будет являться моделью базы данных инфологического уровня [15].

Одной из особенностей разработанной модели является отсутствие в ней типов перечисления. Вместо них повсеместно (например, в таблицах Speciality, TutorRoleRequest, StudentAttendance, DisciplineMaterial и некоторых других) используется целочисленный тип Byte. Это связано с особенностями планируемой к использованию СУБД. По сути, это заглушки, вместо которых будут использоваться соответствующие типы перечислений.

Можно заметить, что все остальные используемые типы данных соответствуют типам языка программирования C#. Причиной использования данной нотации в модели является то, что для доступа к базе данных будет использоваться клиентская библиотека, реализованная для платформы Microsoft .NET. Кроме того, данная библиотека является очень важным ограничением, которое влияет на выбор архитектуры и используемые технологии.

В информационной модели предусмотрены возможности для обеспечения безопасности аутентификации и авторизации: в таблице User предусмотрены поля для хранения хешированного пароля и соли, и, вдобавок, строки с названием хеширующего алгоритма.

Предполагается, что в проектируемой системе не будет возможности изменения набора ролей – существующие выбранные роли очень жестко привязываются к структуре БД. С другой стороны, предметная область и не предполагает выделения большого количества ролей, а практически любое действую-

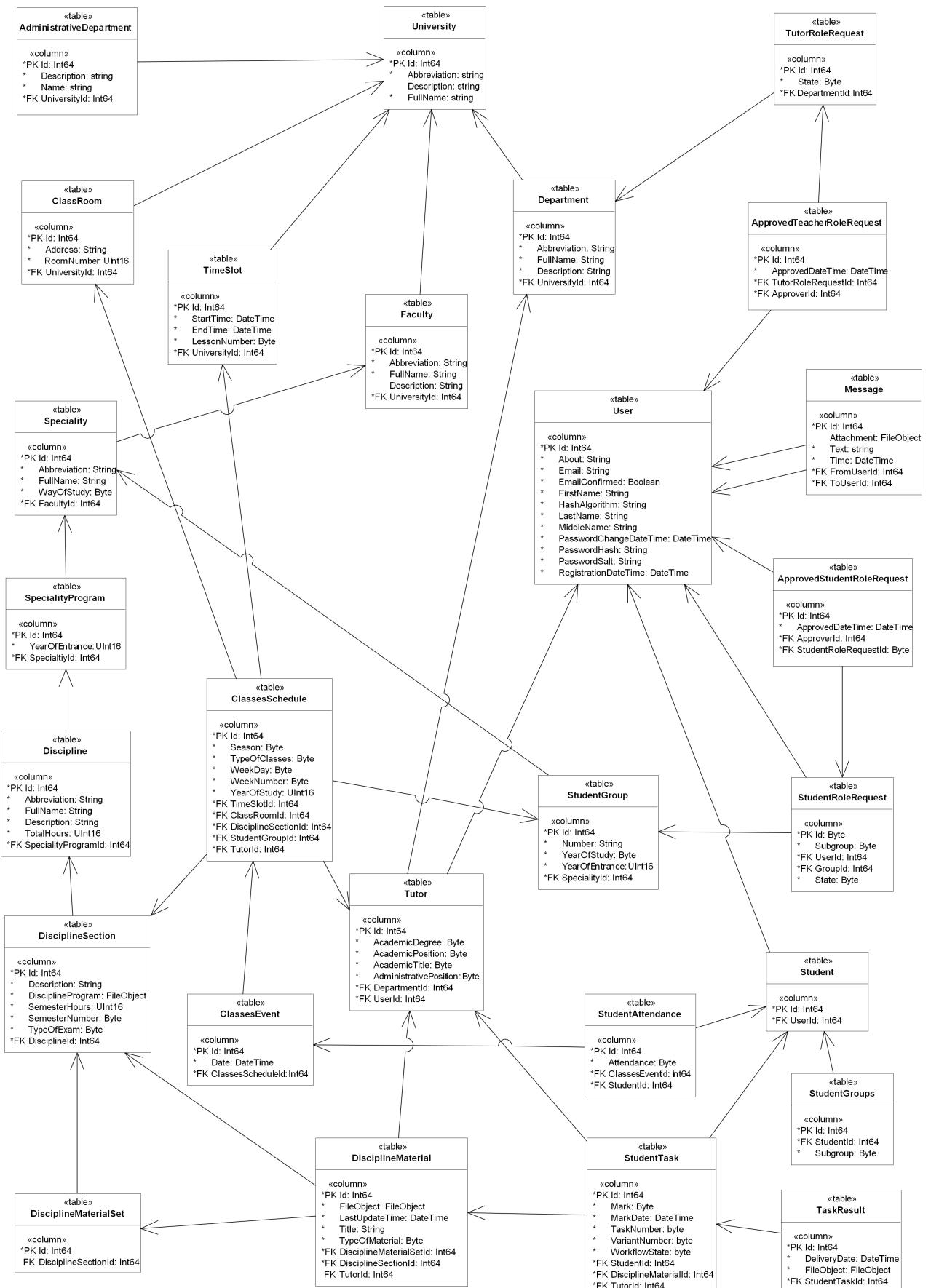


Рисунок 2.3 – Инфологическая модель базы данных

щее лицо можно свести к одной из существующих.

В проектируемой системе предполагается существование иерархии сущностей. На вершине находятся университеты; предполагается поддержка многих ВУЗов одной системой. Университеты имеют в своем составе следующие единицы: административные отделы (используются в информационно-справочных целях), факультеты, кафедры.

2.2 Разработка спецификации функциональных требований

С учетом требований, определенных в подразделе 1.3, представим детализацию функций проектируемого ПС.

2.2.1 Функция регистрации

Функция регистрации должна быть реализована с учетом следующих требований:

- а) Процесс регистрации инициируется пользователем системы (на рисунке 2.2 представлен в виде роли «Гость»).
- б) Функция реализуется собственными средствами без обращения к внешним поставщикам.
- в) Для регистрации пользователь обязан предоставить адрес электронной почты и установить пароль.
- г) Правильность предоставленного адреса электронной почты должна проверяться путем отправки письма со ссылкой, переход по которой означает подтверждение пользователя.
- д) Хранение пароля допускается только в хешированном виде; применяющийся алгоритм должен по криптостойкости быть равным или превосходить алгоритмы семейства SHA-2. Использование соли обязательно.
- е) Должна быть предусмотрена возможность смены пароля и электронной почты после регистрации. Правильность нового адреса почты проверяется отправкой на него письма с подтверждающей ссылкой.

2.2.2 Функция аутентификации

Функция аутентификации должна быть реализована с учетом следующих требований:

- а) Инициатором является пользователь, при этом ему необходимо предоставить адрес электронной почты и пароль, заданные при регистрации.
- б) Должна быть реализована возможность повторной аутентификации пользователя без необходимости ввода какой-либо информации.
- в) Должна быть реализована возможность восстановления пароля:
 - 1) Для восстановления пароля пользователь должен предоставить адрес электронной почты, зарегистрированный в системе.
 - 2) На предоставленный адрес высыпается уникальная ссылка.

3) После перехода пользователем по данной ссылке ему предоставляется возможность установить новый пароль.

2.2.3 Система ролей

При реализации системы ролей следует учесть требования:

а) Должны быть реализованы следующие роли:

- 1) студент;
- 2) преподаватель;
- 3) администратор факультета;
- 4) администратор университета.

б) Для роли администратор университета должна быть реализована возможность назначения любого пользователя администратором факультета.

в) Для получения роли студента или преподавателя у пользователей должна быть возможность создать соответствующую заявку. Подтверждение заявок возлагается на администраторов факультетов.

г) При подаче заявки на роль студента необходимо предоставить информацию о группе, в которой состоит пользователь.

д) При подаче заявки на роль преподавателя необходимо указать кафедру, сотрудником которой является пользователь.

е) Должна быть реализована возможность наличия у одного пользователя нескольких необязательно разных ролей.

2.2.4 Функция просмотра расписания

Просмотр расписания входит в число основных функций разрабатываемого приложения. При реализации данной функции необходимо учесть следующие требования:

а) Необходимо обеспечить отображение расписания занятий в виде последовательного списка, отсортированного в порядке возрастания даты и времени их начала.

б) Расписание должно отображаться на основании указанной группы и подгруппы (для студентов) и указанного полного имени (для преподавателей). В случае, когда подгруппа студента не указана, то должно отображаться расписание всех подгрупп.

в) В расписании для каждого занятия должна быть отображена следующая информация:

- 1) время начала занятия;
- 2) время окончания занятия;
- 3) название (аббревиатура) предмета;
- 4) тип занятия;
- 5) аудитория проведения занятия (с указанием учебного корпуса);
- 6) фамилия и инициалы преподавателя (для студентов) или номера групп (для преподавателей).

г) Расписание должно автоматически загружаться с использованием открытого API университетов.

2.2.5 Функция просмотра предметов

Функция просмотра предметов должна быть реализована с учетом следующих требований:

а) У студентов и преподавателей должна быть возможность просмотра соответственно изучаемых и преподаваемых дисциплин.

б) В списке дисциплин должна быть отображена следующая информация:

- 1) аббревиатура дисциплины;
- 2) фамилии преподавателей или номера групп;
- 3) общее число часов, отводимое планом.

2.2.6 Функция управления заданиями

Функция управления заданиями также является ключевой. При ее реализации должны быть учтены следующие требования:

а) У преподавателя должна быть возможность создания, модификации и удаления индивидуальных заданий для студентов.

б) При создании индивидуального задания необходимо указать его тип, а также предоставить файл с условием.

в) Студенты должны иметь возможность отправлять результаты выполнения заданий с помощью проектируемой системы.

г) В случае отправки заданий на проверку они должны помещаться в очередь.

д) Преподаватель должен иметь возможность просматривать полученные от студентов результаты выполнения и оценивать или отклонять их.

е) Оценивание работ должно производиться по правилам, принятых в ВУЗах: отметки от 10 до 4, а также зачтено. Отметки ниже указанных семантически означают отклонение работы и системой реализовываться не должны.

ж) В случае отклонения результатов выполнения задания (отправки на доработку) или в других случаях студент должен иметь возможность повторно загружать результаты выполнения заданий; при этом должна формироваться их история.

з) Преподаватели также должны иметь возможность загружать учебно-методические материалы, литературу, которая может быть полезной для студентов.

и) Студенты должны иметь возможность просматривать прогресс по всем индивидуальным заданиям по всем предметам на одной странице приложения.

к) Преподаватели должны иметь возможность просматривать количество работ в очереди на проверку на одной странице.

2.2.7 Функция обмена сообщениями

Функция коммуникации должна реализовывать следующие требования:

- а) Должна быть возможность отправки сообщений одного пользователя другому.
- б) История сообщений должна быть представлена в виде диалогов, где отображается текст сообщения, имя пользователя и время отправки сообщения.
- в) Должна быть возможность прикрепления некоторого файла к сообщению.
- г) У пользователей должна быть возможность создания объявлений, которые были бы видны всем другим пользователям. Их история должна отображаться в виде доски объявлений.

2.2.8 Функция просмотра истории студента

Следующая группа требований относится к деятельности преподавателя:

- а) У преподавателя должна быть возможность проверки посещаемости студентами занятий. Проверка должна быть реализована в виде ручного выставления преподавателем отметок о присутствии.
- б) Отметки о посещаемости, а также о выполнении индивидуальных заданий должны отображаться в виде таблиц по каждой группе.
- в) Должна быть возможность просмотра данных таблиц другими преподавателями, если они преподают один и тот же предмет (но, возможно, разные типы занятий).

3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

3.1 Разработка архитектуры программного средства

Как было указано в пункте 1.3.9, на основании анализа вариантов проектирования приложения для различных платформ, проведенного в пункте 1.1.2, было принято решение выбрать основной для разработки платформу веб-приложений.

Классической архитектурой для приложений данного типа является двузвенная клиент-серверная архитектура. В простейшем случае единственная задача сервера – возврат по запросу статической страницы с некоторой информацией. Однако в настоящее время данный подход считается несовременным и используется крайне редко: в основном для информационных сайтов [26]. Абсолютное большинство ресурсов сети Интернет реализованы с высокой степенью интерактивности, то есть предоставляют некоторые элементы, с которыми пользователь может взаимодействовать. Таким образом, клиентская часть приложения, помимо простого отображения информации, должна быть управляемой с помощью пользовательских действий. Выбранная в пункте 1.3.9 программная платформа позволяет нам это реализовать.

Вследствие интерактивности появляется необходимость в хранилище данных, которая обычно решается с помощью реляционных баз данных. В пункте 1.3.9 указано, что в приложении, описываемом в данном дипломном проекте будет использоваться СУБД N*DB, которая может запускаться на физически отдельной ЭВМ. Таким образом, приходим к трехзвенной архитектуре, схема которой представлена на рисунке 3.1.



Рисунок 3.1 – Вариант трехзвенной архитектуры

Появляется вопрос о необходимости серверной части приложения, ведь клиентская часть с помощью, например, протокола HTTP может сама без промежуточных звеньев обращаться к базе данных. При выборе данного варианта исчезает необходимость в дублировании сущностей модели (в базе, на клиенте и на сервере), а также в создании кода, который бы просто перенаправлял запросы от клиента к базе и переупаковывал данные из одних структур в другие.

при отправке клиенту. Однако от данной модификации архитектуры было решено отказаться, поскольку ему свойственны существенные риски нарушения требований безопасности: потребовалось бы наличие открытого API для взаимодействия клиента с БД, которое могло бы легко использоваться злоумышленником. Серверная часть же может использоваться для реализации аутентификации и авторизации.

3.2 Разработка даталогической и физической моделей базы данных

Как было упомянуто ранее, в программном средстве, описываемом данным дипломным проектом, будет использоваться специализированная СУБД N*DB. Описание схемы БД для нее производится с помощью специальной нотации. Тем не менее, разрабатывать модель схемы можно с помощью любых средств, однако затем разработанную модель нужно вручную перевести в формат, используемый СУБД.

На даталогическом уровне модель предметной области представляется в привязке к конкретной СУБД и описывает способ организации данных безотносительно их физического размещения. Описывать модель можно с помощью специальных графических нотаций [15].

Модель даталогического уровня разработаем на основании инфологической модели, описание которой приведено в пункте 2.1.3.

СУБД N*DB разрабатывалась с расчетом на использование программистами в своих приложениях специальной клиентской библиотеки на языке C#. В связи с этим большинство стандартных типов данных одноименны представленным в данном ЯП. Кроме того, данная СУБД предлагает уникальный вариант ее использования. Первоначально, по разработанной в специальной нотации схеме БД генерируется исходный код классов модели. Данный исходный код можно включить свой проект и использовать данные классы модели как обычные регулярные классы. Затем, для сохранения экземпляров классов или их последующего поиска и извлечения и используется клиентская библиотека, которая сериализует экземпляры классов при передаче и десериализует их при получении. Таким образом, взаимодействие с данной СУБД несколько отличается от взаимодействия с традиционным SQL решениями. Но это даёт определенные преимущества. Например, появляется возможность использовать сколь угодно сложные структуры данных и легко сохранять и извлекать их. Таким образом, даталогическую модель используемой в разрабатываемом приложении БД можно проектировать с использованием традиционной диаграммы классов UML 2.1.

Описание схемы БД в специальной нотации, используемой в N*DB, приведено в приложении Б.

Можно выделить несколько особенностей данной схемы.

Модель базы данных при разработке подверглась некоторой денормали-

зации. Чаще всего она выражалась в добавлении дополнительных ссылок одних объектов на другие. Причинами такого решения являются следующие:

- попытка повысить соответствие модели предметной области;
- упрощение манипуляций с данными.

Можно заметить, что достаточно часто в качестве типа поля используется тип Byte. Его использование означает, что данные поля смогут принимать только некоторые определенные значения. Вместо использования типов перечисления было принято решение использовать простой целочисленный тип, поскольку СУБД N*DB предоставляет возможность создания специальных справочников. Данная СУБД изначально проектировалась как распределенная, а справочники – это данные, которые будут храниться на всех узлах.

Необычным для регулярных SQL баз данных является использование списков в качестве типов полей. Как было сказано ранее, возможность из использования появилась вследствие поддержки СУБД сложных агрегирующих типов.

Вследствие этого можно заметить еще одно проявление денормализации: в некоторой дочерней сущности содержится ссылка на родительскую, а родительская содержит список дочерних. Опять же, данное решение вследствие отсутствия необходимости поиска позволит значительно повысить производительность при выборке данных.

Физический уровень моделирования БД описывает конкретные таблицы, связи, индексы, методы хранения, настройки производительности, безопасности. Описывать модель можно с помощью средств, уместных для предметной области [15].

Индексы – это специальные структуры, применяемые для ускорения операций взаимодействия с данными. В СУБД N*DB они имеют название вторичных индексов. Целесообразно реализовать следующие индексы:

- по названиям университетов, административных отделов, факультетов, кафедр, специальностей;
- по соответствующим аббревиатурам;
- по фамилии и имени пользователей;
- по дню недели и времени начала занятий;
- по номеру и году поступления групп.

Для идентификации используется специальное поле id, которое внедряется во все сущности с помощью наследования от сущности Entity.

Остальные настройки будут применяться непосредственно при развертывании программной системы, поэтому в данном разделе не рассматриваются.

3.3 Проектирование и разработка серверной части программного средства

3.3.1 Этапы разработки программного средства

Архитектурный стиль разделения *функциональности* программной системы на уровни поднимает сразу несколько проблем. Их решение в типичном случае заключается в выполнении следующих этапов [13]:

- а) определение стратегии разбиения на уровни;
- б) определение уровней;
- в) распределение функциональности по уровням и компонентам;
- г) уточнение количества уровней: при необходимости некоторые из них объединяются;
- д) установление правил взаимодействия уровней;
- е) идентификация функциональности, которая используется на всех уровнях (*cross cutting concerns*);
- ж) определение интерфейсов уровней;
- з) выбор стратегии развертывания;
- и) выбор конкретных протоколов взаимодействия.

К данному этапу разработки дипломного проекта выполнены шаги с 3.3.1а по 3.3.1д. Шаг 3.3.1е вследствие использования различных программных средств, применяемых на уровнях, будет выполняться позднее – на соответствующих уровнях.

Таким образом, следующий этап – это определение интерфейсов уровней. СУБД имеет заранее определенный интерфейс, установленный ее разработчиком. Предполагается, что из остальных двух частей программной системы инициатором любых действий будет являться клиентская. Следовательно, единственный межуровневый интерфейс, который следует определить – это интерфейс серверной части.

Веб-приложения, в стиле которых планируется создание клиентской части, обычно запускаются большим числом пользователей. Таким образом очевидна автономность и независимость частей проектируемой программной системы. Исходя из этих соображений, целесообразным является применение сервис-ориентированного архитектурного стиля.

3.3.2 Программный интерфейс серверной части

Предварительным условием для проектирования серверной части приложения является определение его интерфейса в высокоуровневых терминах предметной области. Таким образом, далее приведены методы его API, которые основаны на функциональных требованиях, определенных в подразделе 2.2:

– метод регистрации пользователя системы, принимающий адрес электронной почты, хеш-сумму пароля, строку соли и название применявшегося алгоритма хеширования;

- метод смены пароля, также принимающий хеш-сумму пароля, соль и название алгоритма;
- метод смены пользователем электронной почты;
- метод аутентификации, принимающий хеш пароля и адрес электронной почты;
- метод восстановления пароля, принимающий адрес электронной почты;
- метод извлечения персональных данных пользователей;
- метод изменения фамилии, имени, отчества пользователя;
- метод изменения значения поля «О себе» пользователя;
- метод смены роли пользователя, принимающий объект сущности пользователя и роль, которую желается установить;
- метод подачи заявки на роль студента, принимающий номер группы;
- метод подачи заявки на роль преподавателя, в качестве параметра в котором выступает объект кафедры, сотрудником которой является преподаватель;
- методы подтверждения и отклонения заявок студентов;
- методы подтверждения и отклонения заявок преподавателей;
- метод, принимающий диапазон дат и номер группы и возвращающий список занятий;
- метод, который аналогичен предыдущему, но принимающий в добавок номер подгруппы;
- метод, принимающий диапазон дат и фамилию, имя и отчество преподавателя и возвращающий список его занятий;
- метод, возвращающий список изучаемых студентов дисциплин;
- метод, возвращающий список дисциплин преподавателя;
- метод создания индивидуального задания преподавателем, принимающий номера групп и файл с условием задания;
- методы модификации условия индивидуального задания, включающие поиск и собственно изменение;
- метод отправки результатов выполнения заданий студентом;
- метод отклонения результатов задания преподавателем;
- метод оценивания результатов;
- метод извлечения всех результатов выполнения заданий, полученных и не проверенных преподавателем;
- метод добавления преподавателем файла учебно-методических материалов;
- метод получения прогресса выполнения заданий по всем предметам студента, в том числе полученные оценки (при их наличии);
- метод отправки сообщения одного пользователя другому;
- метод отправки сообщения с прикреплением;

- метод получения списка сообщения, переданных между двумя пользователями, принимающий помимо идентификационных номеров пользователей число сообщения для возврата и смещение, с какого сообщения начинать их выборку;
 - метод создания объявления пользователя;
 - метод извлечения объявлений пользователя, также принимающий их число и номер, с которого начинать выборку;
 - метод получения списка студентов;
 - метод получения списка студентов с их оценками по индивидуальным заданиям, а также с отметками посещения;
 - метод совершения проверки посещения группой занятия;
 - метод поиска пользователей;
 - метод поиска преподавателей;
 - метод поиска факультетов;
 - метод поиска кафедр;
 - метод поиска групп студентов;
 - метод, возвращающий все группы факультета;
 - методы выборки, создания и модификации специальностей факультета;
 - методы выборки, создания и модификации программ специальностей.

На рисунке 3.2 приведена схема алгоритма обработки запросов перечисленных методов программного интерфейса серверной частью приложения. Далее приведены архитектурные решения, которые были применены при реализации данной части программной системы.

Поскольку в ПС проектируется реализация сложных независимых систем ролей и пользовательских функций, которые находят своё отражение в серверной части приложения, то встаёт вопрос об их сопряжении. А поскольку данное сопряжение вследствие особенностей предметной области является достаточно нетривиальным: часть функций уникальна для некоторых функций, часть разделяется ими всеми. Поэтому целесообразным является разбиение их на следующие группы:

- а) функции, доступные без аутентификации;
- б) функции, доступные для всех аутентифицированных пользователей;
- в) функции студентов;
- г) функции преподавателей;
- д) функции студентов и преподавателей;
- е) функции администраторов факультетов;
- ж) функции администраторов ВУЗов.

Часть алгоритма, отвечающее за извлечение расписания занятий из БД, реализована на примере БГУИР, в котором занятия происходят по одной из 4 учебных недель, отсчет которых начинается с 1-го сентября.

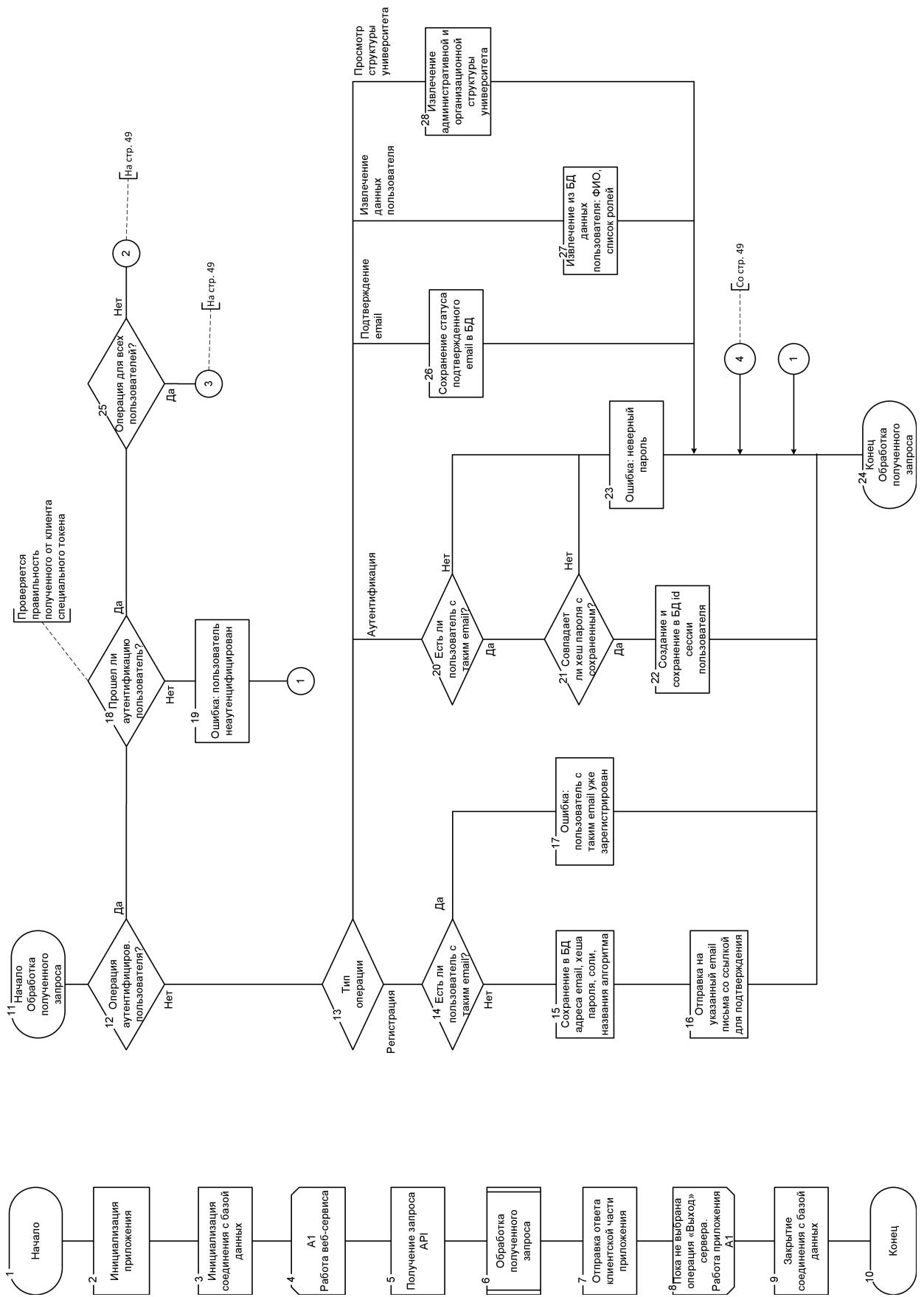


Рисунок 3.2 – Схема программы серверной части программного средства

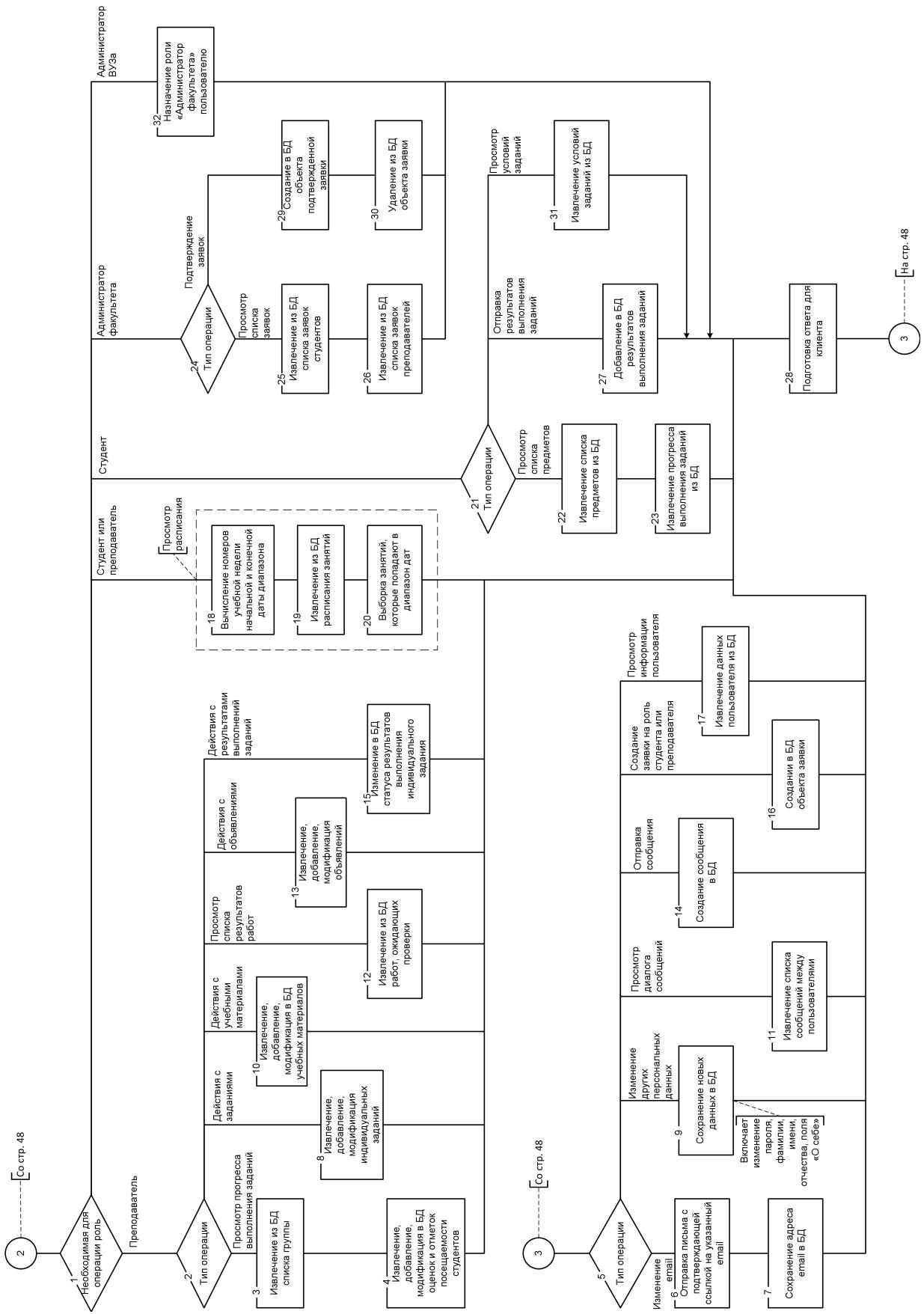


Рисунок 3.2 – Схема программы серверной части программного средства (окончание)

Серверная проверка аутентификации пользователей будет осуществляться с помощью технологии создания и проверки специального токена.

Большое количество блоков, в которых осуществляется доступ к БД, подтверждает назначение всей серверной части приложения: прослойка между клиентским веб-приложением и базой данных с добавлением проверок аутентификации и авторизации.

Реализовав перечисленные методы API, можно будет достигнуть очень важной цели: внутренняя бизнес-логика, логика доступа к данным и сами данные будут надежно защищены.

3.3.3 Выбор протоколов коммуникации

В пункте 3.3.1 приведены этапы, в соответствии с которыми рекомендуется производить проектирование программного системы [13]. Этап 3.3.1з будет рассмотрен позднее вследствие неоднородности и отсутствия схожести в применяющихся технологий между уровнями. Таким образом, следующий актуальный этап – 3.3.1и: выбор конкретных протоколов взаимодействия.

Можно выделить два основных стиля, которые (с необходимыми модификациями в каждом конкретном случае) применяются в информационных системах, содержащих веб-сервисы [13]: REST (Representational State Transfer – протокол передачи состояния представления) и SOAP (Simple Object Access Protocol – простой протокол доступа к объектам). Технически REST является архитектурным шаблоном проектирования, построенным на основе использования простых глаголов (называемых запросами или методами). На практике данный архитектурный стиль применяется в связке с протоколом прикладного уровня модели OSI HTTP. SOAP же является XML-ориентированным протоколом, причем стандарт не специфицирует применяемые протоколы передачи данных.

Основное отличие между данными протоколами состоит в способе манипулирования состоянием. SOAP предусматривает осуществление переходов между различными состояниями с помощью взаимодействия с единственной конечной точкой ввода информации, через которую предоставляется доступ к функциональности сервиса. Протокол REST предполагает использование ограниченного набора операций, которые могут применяться к ресурсам, адресуемым с помощью уникальных адресов URI.

Данный протокол в большей степени соответствует распределенной архитектуре клиент-серверных приложений, поскольку серверная часть публично доступна для множества заведомо неизвестных клиентских приложений. REST обладает свойством отсутствия состояния, что означает, что каждый запрос должен содержать в себе достаточный набор данных для его осуществления. Кроме того, данный протокол не запрещает использование различных форматов передачи данных, например, JSON, что оказывается огромным преимуществом перед SOAP вследствие планируемого использования в клиентской части

языка TypeScript, который нативно поддерживает данный формат.

3.4 Проектирование и разработка клиентской части программного средства

Задачи клиентской части приложения включают отображение пользовательского интерфейса и обработку действий пользователя. Типичными этапами его проектирования являются следующие [13, с. 78]:

- Идентификация типа клиентской части приложения, которое удовлетворяет установленным требованиям. Осуществление данного этапа осуществлено в подразделе 3.1.
- Выбор технологии пользовательского интерфейса. Осуществляется на основе анализа требуемой для реализации функциональности.
- Проектирование UI. Хорошей практикой является реализация модульности, а также принципа разделения ответственности компонентов.
- Определение стратегии и протоколов обмена информации между уровнями. Поскольку рассматриваемый уровень является самым верхним, а его протоколы связи с серверной частью приложения были рассмотрены в пункте 3.3.3, то данный вопрос считается решенным и рассматриваться не будет.

Ранее был проведен анализ и осуществлен выбор целевой платформы для реализации клиентской части приложения, был выбран основной язык программирования. Однако, существует огромное число специализированных технологий и фреймворков по созданию пользовательских интерфейсов. Представляется целесообразным провести уточнение средств разработки.

3.4.1 Уточнение выбора технологий программирования

Язык программирования TypeScript является надмножеством языка JavaScript. Основной его особенностью является наличие проверок типов во время компиляции. Обычный JavaScript код может быть скомпилирован TypeScript компилятором (на выходе получится тот же самый код), однако по умолчанию данная возможность выключена, компилятор будет производить ошибку. Тем не менее, у разработчиков программного обеспечения есть возможность использования неимоверно большого числа JavaScript библиотек и фреймворков, которая заключается в создании и использовании специальных файлов (их принято создавать с расширением .d.ts), которые содержат определения типов (type definitions) и создаются для каждой предназначено к использованию библиотеки. Определенную проблему может представлять создание таких файлов: они бывают довольно громоздкими и их создание требует некоторых временных затрат. Тем не менее, существует проект по созданию таких файлов обычными разработчиками [27]. В репозитории данного проекта находятся уже созданные .d.ts файлы для огромного числа существующих библиотек, что означает, что TypeScript разработчики глубоко интегрированы в экосистему JavaScript разра-

ботки.

Одной из широко используемых в настоящее время библиотек по созданию интерактивных интерфейсов веб-приложения является библиотека React. Выбор в ее пользу был осуществлен вследствие наличия опыта по ее использованию у членов команды. Особенностями данной библиотеки являются следующие [28]:

- Кроссплатформенность. Переиспользование существующего кода возможно даже на других платформах благодаря проекту React Native.
- Декларативность: использование элементов (стандартных для React объектов, которые представляют собой HTML-теги) и компонентов (объекты, создаваемые разработчиком).
- JSX: техника создания и использования компонентов с помощью HTML-подобного синтаксиса, который применяется прямо в JavaScript коде.
- Виртуальный DOM: дерево React элементов, которое отрисовывается в браузере, причем изменения в нём не требуют полной перерисовки всего интерфейса. Позволяет значительно улучшить быстродействие при использовании библиотеки.

Строго говоря, использование JSX является опциональным и может как не использоваться при разработке с React, так и использоваться при разработке с помощью других библиотек. Однако, он значительно упрощает исходный код компонентов. Например, следующий JavaScript код

```
return <div>Hello {this.props.children}</div>;
```

после компиляции будет преобразован в следующий

```
return React.createElement(  
    "div",  
    null,  
    "Hello ",  
    this.props.children  
)
```

Лаконичность использования JSX очевидна. Некоторым кажется сомнительным данный подход, в котором смешивается исполняемый код и код разметки. Однако, данное смешение формирует исходный код представления. Кроме того, в других фреймворках используются специальные конструкции по реализации, например, циклов, условий, в то время как при использовании JSX нужды в новых конструкциях нет, ведь используются стандартные операторы языка JavaScript (в текущем проекте – TypeScript).

Одна из классификаций компонентов React предполагает их разделение на pure (простые) и stateful (имеющие внутреннее состояние) [29]. Простые компоненты, реализованные в виде классов, унаследованных от React.Component, переопределяют метод render(), который принимает некоторый набор

исходных данных и возвращает элемент, предназначенный для отображения. Компоненты другого класса имеют внутри себя некоторые данные, образующие их состояние. Чтобы React узнал, что данные изменились, и осуществил перерисовку измененных компонентов, требуется использовать специальный метод `this.setState(...)`.

Для упрощения управления состоянием компонентов могут применяться различные техники. Одной из них является использование специальной библиотеки MobX. Его идея заключается в недопущении неконсистентного состояния приложения, что достигается отслеживанием данной библиотекой всего дерева используемых объектов и обновлении интерфейса при любом их изменении. Рекомендуемый и наиболее удобный способ использования MobX заключается в использовании специальных декораторов, с помощью которых помечаются классы, методы и поля классов. Не смотря на их официальный экспериментальный статус, они поддерживаются компилятором TypeScript. В терминологии MobX существуют следующие объекты программы:

- React компонент, который автоматически перерисовывается при изменении состояния. Помечается декоратором `@observer`.
- Объекты в классах, находящиеся под наблюдением библиотеки. Обозначаются с помощью декоратора `@observable`.
- Вычисляемые методы, возвращающие некоторое значение на основании состояния объекта или других вычисляемых методов. Подвергаются оптимизации со стороны MobX. Обозначаются декоратором `@computed`.
- Методы, изменяющие состояние объектов. Обычно это некоторые операции ввода-вывода, а также передачи данных. Помечаются с помощью `@action`.
- Методы, которые должны автоматически запускаться при любых изменениях содержащихся в них данных. Типичный пример их использования: логгирующие операции. Обозначаются с помощью `@autorun`.

Однако, MobX представляет собой только библиотеку. Ограничения на архитектуру приложения не накладываются, однако сохраняется необходимость ее реализации программистом. Одним из вариантов является разбиение всех компонентов на контейнерные и презентационные [30]. Особенности создания презентационных компонентов следующие:

- Их задача заключается в определении, как должны *выглядеть* элементы.
- Не зависят от других частей приложения.
- Получают данные исключительно в качестве входных параметров.
- Не изменяют данные.
- Редко имеют собственное состояние (в таком случае это состояние UI, а не собственно данные).
- Обычно содержат JSX разметку и имеют относящиеся к ним стили.

В это же время особенности контейнерных компонентов заключаются в следующем:

- Их задача заключается в определении, как элемент должны *работать*.
- Предоставляют данные и методы их обработки другим компонентам.
- Как правило имеют некоторые данные в виде состояния.
- Обычно не содержат JSX разметки и никогда не имеют стилей.

Использование данного подхода имеет следующие преимущества:

- Достигается выполнение принципа разделения ответственности.
- Появляется возможность легкого переиспользования компонентов с различными источниками данных.

- Одни и те же презентационные компоненты могут использоваться повсеместно в приложении.
- Презентационные компоненты формируют «палитру». Их внешний вид может изменяться без влияния на логику приложения.

Еще один важный момент относительно проектирования UI, помимо содержания и интерактивных пользовательских взаимодействий, касается стилизации внешнего вида компонентов. Существует несколько подходов по ее исполнению [31]: использование внутренних (*inline*) и внешних стилей.

Первый подход имеет множество недостатков:

- невозможно использование медиа-запросы, чтобы, например, реагировать на изменения ширины экрана или отличать мобильное устройство от настольного компьютера;
- невозможно использование псевдоклассов и псевдоэлементов, например: `:hover`, `:active`, `:before`, `:after`;
- значительное увеличение размера разметки и снижение производительности;
- невозможно переопределение стилей по более сложным селекторам.

Тем не менее, Facebook, как компания разработчик React, несмотря на описанные недостатки, выступает за использование *inline* стилей. Причиной этому является стремление обеспечить единство с React Native, поскольку там возможность стилизации обеспечивается только данным подходом.

Второй подход – использование внешних стилей – является более традиционным для веб-разработки. Его эволюционное развитие включает несколько этапов:

- Создание одного файла `.css` со стилями, который подключается один раз глобально в главный `html` файл. В React компонентах используется тег `className` со значениями имен классов из этого файла.
- Разбиение единого файла стилей на множество файлов в соответствии с реализованными компонентами. В файлы компонентов подключаются лишь те стили, которые там необходимы. Преимущества данного подхода заключаются в простоте поиска файла, в который необходимо внести изменения, и в

простоте удаления компонентов и соответствующих им файлов стилей.

- Использование различных препроцессоров стилей, таких как SASS/SCSS, LESS и прочих.

Использование SASS по сравнению с обычным CSS предоставляет следующие преимущества [32]:

- переменные, в которых можно хранить значения цветов, шрифтов, а также любые другие значения;
- вложенность, что приводит к большей наглядности файлов стилевых таблиц за счет соответствия иерархии HTML кода;
- фрагментирование, то есть обеспечение модульности;
- импортирование других файлов стилей, которое, в отличие от CSS, вместо создания новых HTTP запросов подставляет указанный файл в тот, где он вызывается, таким образом на выходе получается единственный файл стилей;
- примеси (mixins), которые представляют собой группы деклараций, используемые по нескольку раз;
- наследование, позволяющее привносить наборы свойств от одного селектора к другому;
- использование математических операторов.

Таким образом, на основании проведенного уточнения технологий для использования выбираем следующие: React в связке с MobX и SCSS, кроме того, для разметки в коде повсеместно будет использоваться JSX.

3.4.2 Настройка инструмента сборки проекта

Рассмотренные технологии достаточно разнородны, применяются для решения различных задач. Тем не менее их согласованное взаимодействие обеспечивает правильность работы всего приложения. Для его реализации могут использоваться различные средства, однако стандартом индустрии является Webpack.

Webpack представляет собой инструмент для сборки веб-приложений. Главная его задача состоит в объединении большого числа файлов исходного кода в один, что значительно улучшает производительность за счет необходимости клиента загружать единственный файл. Однако его возможности далеко не ограничиваются одной лишь сборкой.

Его настройка производится с помощью специального конфигурационного файла webpack.config.js. Содержание файла, используемого в данном дипломном проекте, приведено в приложении В. В таблице 3.1 приведено описание используемых параметров.

Как было упомянуто ранее, основная задача Webpack состоит в сборке всех файлов исходного кода проекта в один. Данная процедура осуществляется с помощью загрузчиков (loaders). Большая их часть разрабатывается открытым сообществом разработчиков. Их использование обязательно в проектах, в кото-

Таблица 3.1 – Параметры инструмента сборки Webpack, используемые в проекте

Название параметра	Описание
entry	Указывает главный файл исходного кода (так называемая точка входа в приложение).
output	Выходной файл, производимый в результате сборки проекта. Именно его необходимо подключать в главном HTML файле.
devtool	Способ создания карт кода (source maps). Применяется при отладке приложения для ускорения нахождения проблемных участков кода.
resolve.extensions	Позволяет в исходном коде в конструкциях import не указывать расширения файлов.
resolve.modules	Указывает директории, в которых должен осуществляться поиск импортируемых файлов.
module.rules	Указывает набор и порядок запуска загрузчиков (loaders).
externals	Позволяет не включать некоторые библиотеки в окончательную сборку. Предполагается, что данные библиотеки будут доступны во время исполнения.
devServer	Настройки отладочного сервера.

рых используются файлы исходного кода, не являющихся обычными JavaScript файлами.

Основной язык программирования данного проекта – TypeScript. В связи с этим необходимо подключить специальный загрузчик (ts-loader), который бы компилировал файлы с исходным кодом. Однако для совместимости перед ним подключен другой загрузчик source-map-loader, который загружает карты кода и передает их указанному инструменту по их обработке (параметр devtool таблицы 3.1). Затем в очереди обработки загрузчики, ответственные за подключение всех файлов стилей (sass/scss, css). В самую последнюю очередь с помощью file-loader загружаются вспомогательные файлы различных расширений, которые необходимы для работы специальных шрифтов, а также загрузки статических файлов, таких как картинки.

Для облегчения разработки и отладки в параметрах специфицируется средство для создания карт кода (его назначение состоит в сопоставлении файлов исходного кода выходному файлу, так что становится проще находить и отлаживать конкретные строки кода), а также специальный веб-сервер. Данный сервер (webpack-dev-server), которые позволяет производить разработку и сразу же наблюдать получаемый результат. Это достигается за счет отслеживания изменяемых файлов, их перекомпиляции на лету (Hot-Module-Replacement)

и отправки сообщения браузеру на обновление страницы. Кроме того, данный сервер работает очень быстро за счет того, что он загружает всё приложение в оперативную память.

3.4.3 Настройка пакетного менеджера

В настоящее время при веб-разработке не принято реализовывать абсолютно все компоненты и всю функциональность самостоятельно. Вместо этого существует большое число сторонних библиотек, разрабатываемых миллионами разработчиков по всему миру. Стандартом индустрии является использование специальных пакетных менеджеров для управления ими. Наиболее часто используемым является npm (Node Package Manager).

Его настройка для конкретного проекта осуществляется с помощью конфигурационного файла package.json (содержание файла, используемого в данном дипломном проекте, приведено в приложении В). В таблице 3.2 приведено описание используемых параметров.

Таблица 3.2 – Параметры пакетного менеджера npm, используемые в проекте

Название параметра	Описание
version	Версия пакета проекта.
name	Название пакета проекта.
private	Используется для предотвращения случайной публикации пакета проекта.
scripts	Скрипты, которые могут быть использованы для сборки и запуска проекта.
dependencies	Список зависимостей, необходимых для работы приложения.
devDependencies	Список зависимостей, необходимых для разработки и тестирования.

Самыми важными параметрами являются version и name: без них проект даже не будет запускаться. Параметр private запрещает публикацию кода проекта в открытые базы библиотек.

Параметр scripts специфицирует команды, с помощью которых может быть запущена компиляция, сборка и запуск проекта. Следующая команда

```
npm start webpack
```

производит сборку проекта, а затем погружается в режим слежения, который рекомпилирует исходные файлы в случае их изменения. Можно заметить, что даже инструмент сборки Webpack подключается в качестве зависимости (devDependency).

Другой пример: следующая команда

```
npm start run
```

осуществляет запуск отладочного сервера (описание его работы приведено в пункте 3.4.2). Для запуска данных команд прямо из среды разработки Visual Studio может использоваться специальное расширение NPM Task Runner.

3.4.4 Проектирование клиентской части приложения

Далее рассмотрим вопрос взаимодействия пользователя с клиентской частью приложения. Известно, что удобство пользования программным средством может во многом определять успешность проекта в целом [6, с. 44].

Схема работы клиентской части программной системы представлена на рисунке 3.3. Среди ее особенностей можно выделить в высокой степени соответствие диаграмме прецедентов, которая была составлена и рассмотрена в пункте 2.1.2. Кроме того, данный чертеж представляет собой отображение чертежа схемы серверной части с отличием в том, что там делается акцент на взаимодействие с базой данных, а в данном чертеже – на отображении данных и интерактивном взаимодействии с пользователем.

Таким образом, составленная схема клиентской части ПС будет использована при разработке навигации (routing) по страницам веб-приложения.

3.4.5 Конструирование клиентской части приложения

Наконец, по завершению этапов проектирования и подготовки можно приступить к собственно созданию исходных кодов приложения. Ключевые фрагменты кода, описание которых приведено в данном пункте, приведены в приложении А.

Для начала необходимо создать корневой файл index.html. Ключевая его особенность состоит в следующем теге

```
<div id="root"></div>
```

Несмотря на то, что он объявляется пустым, именно в него библиотека React подставит всё приложение.

Кроме этого, в данном файле подключается основная зависимость приложения – непосредственно сама библиотека React и библиотека по управлению виртуальным деревом элементов браузера, а также пакет bundle.js, в который будет собран весь созданный исходный код. Помимо библиотек, подключающиеся некоторые файлы стилей, необходимые для корректной работы некоторых компонентов, например, специальных шрифтов, предоставляющих возможность использования большого количества специальных иконок.

Следующий важный файл – корневой файл исходного TypeScript кода index.tsx. Единственная его цель – вызов специального метода ReactDOM.render(), который и подставляет в указываемый тег каркас приложения.

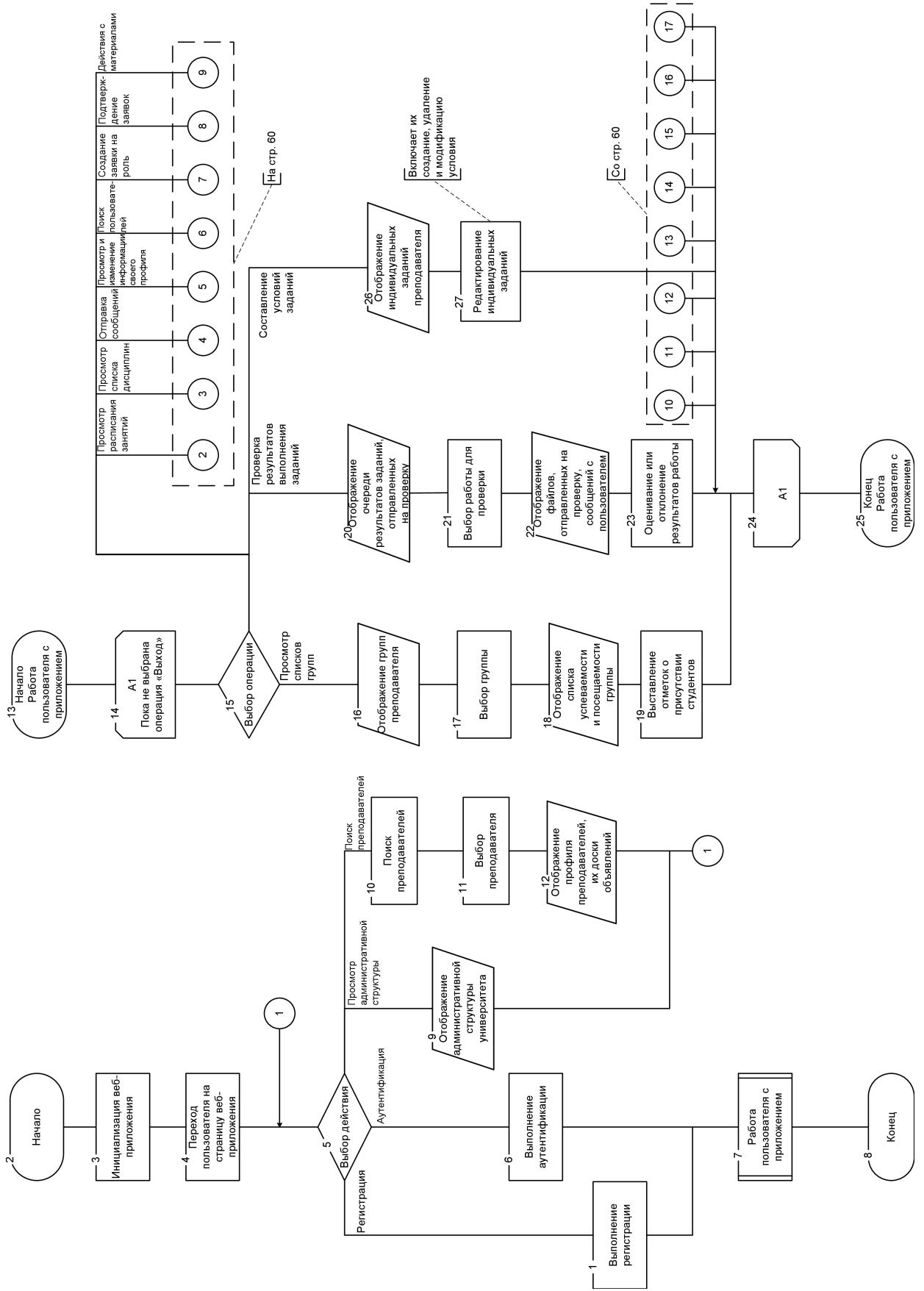


Рисунок 3.3 – Схема программы клиентской части программного средства

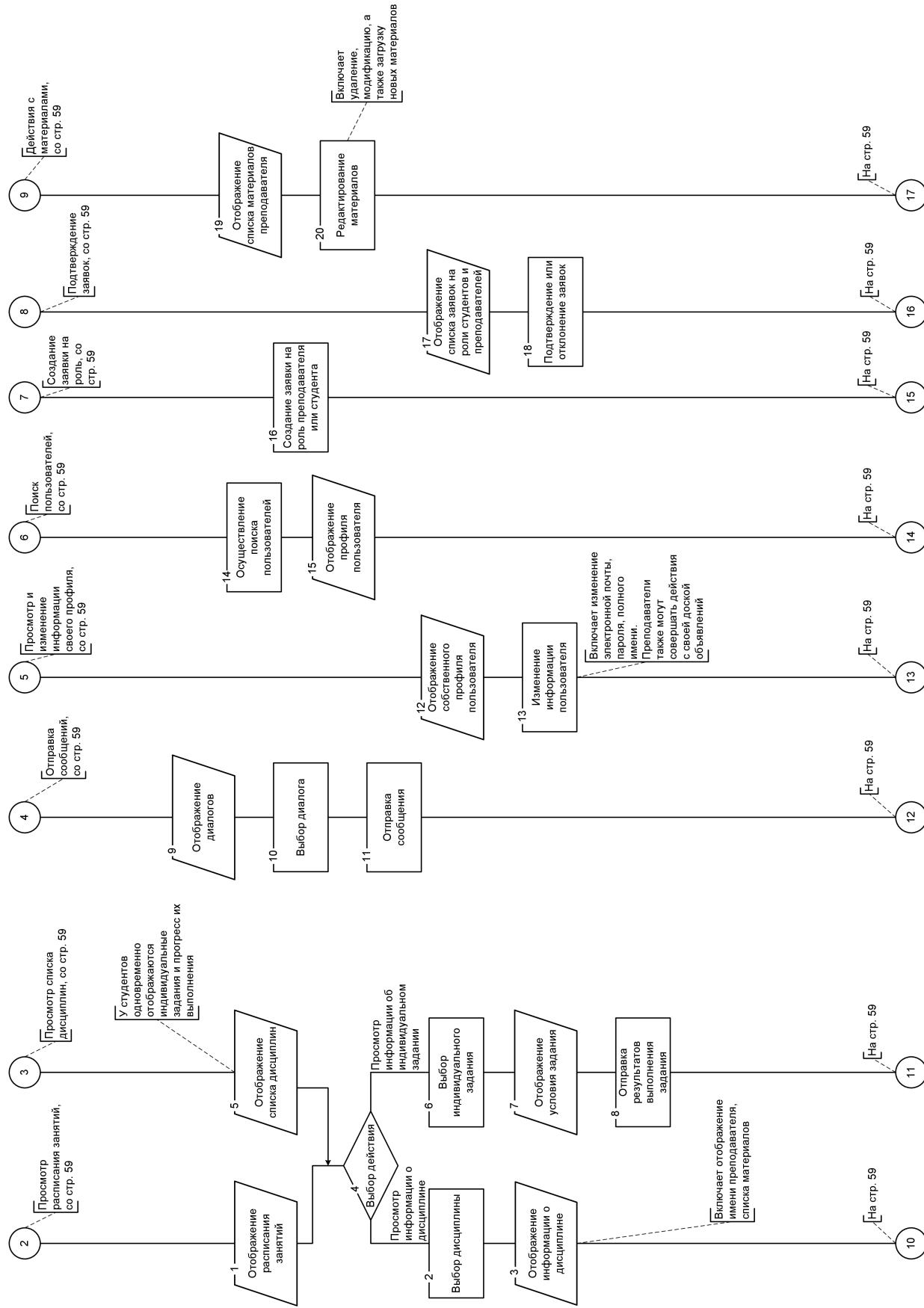


Рисунок 3.3 – Схема программы клиентской части программного средства (окончание)

Каркас приложения, определенный в файле App.tsx, представляет собой класс AppRouter, в котором и задаются пути маршрутизации страниц веб-приложения. В нем используется стандартный элемент Router, который при переходе пользователем по адресам приложения создаёт экземпляр класса App и передает в качестве его содержимого соответствующую страницу.

Задача компонента App – рендеринг шапки сайта и бокового меню. В качестве дочерних для него React подставляет такие компоненты, как Agenda (отображающий список расписания занятий), Disciplines (отображающий список дисциплин) и другие.

В качестве решения по организации пользовательского интерфейса был принят шаблон, состоящий из двух колонок. Таким образом, в левой колонке будут отображаться списки, а в правой – детали выбранного из списка элемента. Кроме того, для повышения удобства пользования было выдвинуто требование, чтобы правая колонка не изменяла содержимого при переходах пользователям по страницам приложения. Данное требование удовлетворяется в связи с использованием библиотеки React, поскольку она достаточно эффективно реализует механизм перерисовки только измененных областей.

Задача дочерних компонентов класса App состоит в обращении к классам, обеспечивающим сетевое взаимодействие с сервером, получении от них данных, а затем создании «чистых» (pure) компонентов, которые эти данные отображают. При этом разработка производится путем движения сверху-вниз (от общих компонентов к очень маленьким частным), причем достигается высокая степень модульности: число уровней иерархии достигает не менее трех, и только самые частные компоненты отвечают за непосредственный вывод информации.

За хранение данных, а также их извлечение отвечают специальные классы, называемые «хранилищами» (stores). Данный подход предлагается в качестве лучших практик разработчиками библиотеки MobX [33]. Предлагаемый ими подход заключается в создании специальных классов, называемых хранилищами предметной области (domain stores), ответственных за определенные аспекты работы приложения, и разделении данных между ними. В данном дипломном проекте, такими аспектами являются хранение расписания, списка предметов, сообщений и так далее. Кроме того, предлагается создание еще одного хранилища, ответственного за хранение состояния интерфейса (UiStore), которое может хранить следующую информацию: сессионные данные, текущую тему приложения, язык (locale), разрешение экрана, состояние элементов интерфейса и так далее. Например, мы будем хранить в данном классе компоненты, ответственные за левую и правую колонку приложения.

Помимо кода компонентов, необходимо создать код их стилизации. Благодаря использованию SASS/SCSS можно достигнуть такой же степени модульности, что и при реализации компонентов. Поэтому часто классы компо-

нентов имеют единственный им соответствующий файл стилей и импортируют лишь его.

Таким образом, разработает исходные коды клиентской части приложения.

3.5 Развёртывание программного средства

После выявления, а также завершения проектирования всех компонентов программного средства появляется вопрос о планировании развертывания всей системы. Необходимо составить описание требуемых аппаратно-программных комплексов, которые понадобятся для обеспечения функционирования распределенного приложения. Для этих целей целесообразным выглядит составление диаграммы развертывания стандарта UML 2.1.

Данная диаграмма представлена на рисунке 3.4. Она отражает следующие особенности развертывания:

- На узле окончного устройства в качестве среды выполнения перечислен список браузерных программных средств, с помощью которых можно использовать клиентскую часть приложения.
 - В качестве операционной системы для сервера клиентской части перечислен список поддерживаемых веб-сервером ОС.
 - При необходимости Apache HTTP Server может быть заменен другим HTTP-сервером.
 - Для серверной части программного средства и базы данных показано их развертывание на отдельных узлах. При самом развертывании в зависимости от условий поставщика вычислительных мощностей данные элементы программной системы могут быть объединены на одном узле.
 - В свою очередь, помимо упрощения, возможно и усложнение схемы развертывания, например, база данных будет развернута на нескольких узлах. Тем не менее, все узлы должны удовлетворять отображенным условиям.
 - Все серверы: и клиентской части, и серверной, и базы данных, – могут быть физически расположены в различных data-центрах.
 - Предполагается, что пользовательское окончное устройство значительно удалено от серверов программной системы, доступ осуществляется через сеть Интернет, что и упрощенно показано на диаграмме.
- Таким образом, после развертывания программного средства пользователи могут уже начать им пользоваться.

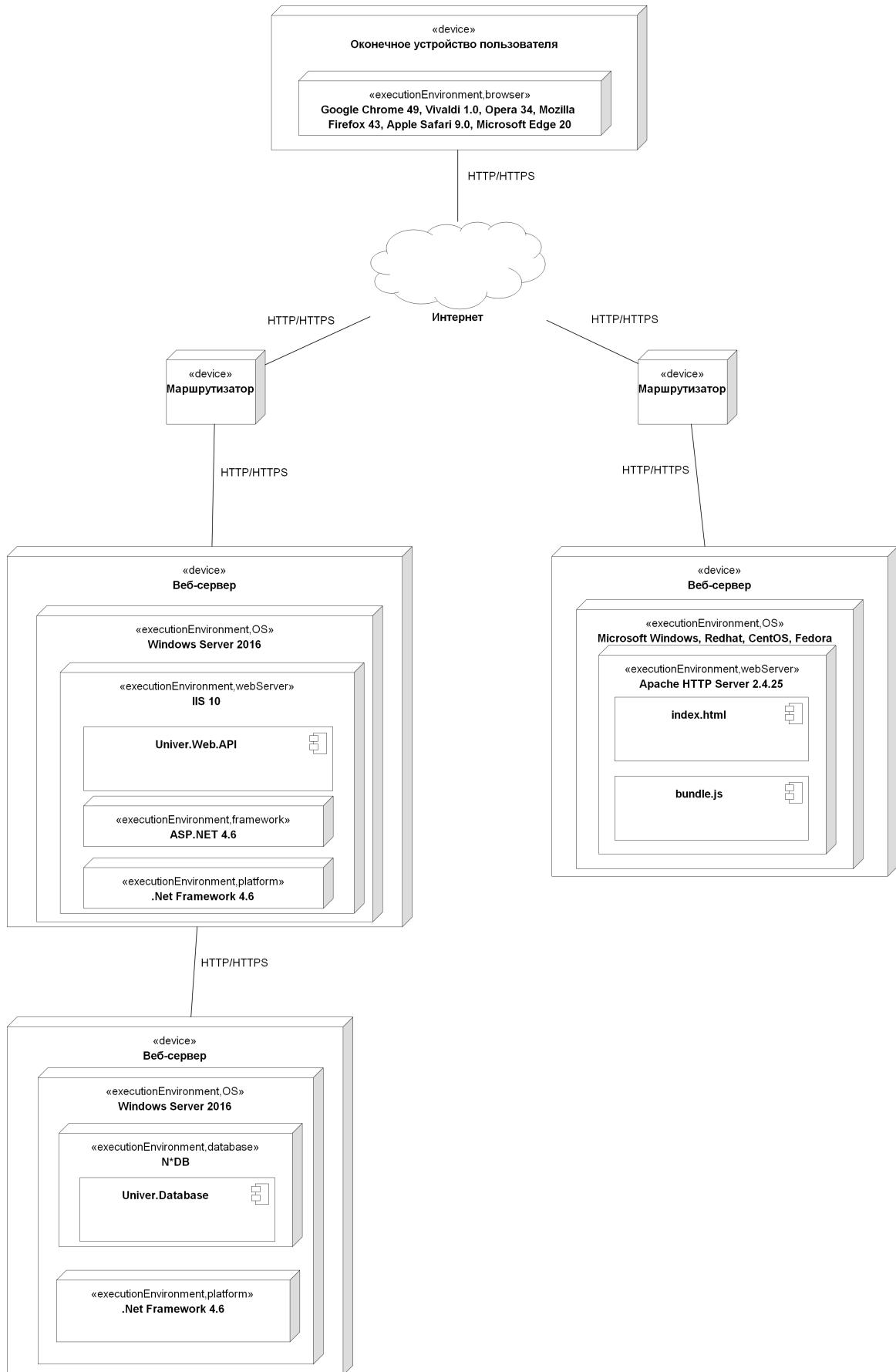


Рисунок 3.4 – Диаграмма развертывания ПС

4 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта [34]. Вот уже несколько десятков лет его стабильно включают в планы разработки как одна из основных работ, причем выполняемая практически на всех этапах проектов. Важность своевременного выявления дефектов подчеркивается выявленной эмпирически зависимостью между временем допущения ошибки и стоимостью ее исправления: график данной функции круто возрастает.

Тестирование можно классифицировать по очень большому количеству признаков. Основные виды классификации включают следующие [34]:

- а) по запуску кода на исполнение:
 - 1) статическое тестирование – без запуска программного средства;
 - 2) динамическое тестирование – с запуском;
- б) по степени автоматизации:
 - 1) ручное тестирование – тестовые случаи выполняет человек;
 - 2) автоматизированное тестирование – тестовые случаи частично или полностью выполняет специальное инструментальное средство;
- в) по принципам работы с приложением:
 - 1) позитивное тестирование – все действия с приложением выполняются строго в соответствии с требованиями без недопустимых действий или некорректных данных;
 - 2) негативное тестирование – проверяется способность приложения продолжать работу в критических ситуациях недопустимых действий или данных.

В данном разделе проведем динамическое ручное тестирование. Его целью является подтверждение соответствия работы программного средства установленным в начале разработки требованиям. Успешное выполнение приведенных в данном разделе тестовых случаев должно подтвердить работоспособность программного средства в основных сценариях использования, а также устойчивость к неверным входным данным. В таблице 4.1 приведен список тестовых случаев, относящихся к позитивному тестированию, в таблице 4.2 – к негативному.

Таблица 4.1 – Тестовые случаи позитивного тестирования

Модуль (экран)	Описание тестового случая	Ожидаемые результаты	Тестовый случай пройден?
1			
Аккаунт	<p>1. Регистрация. Предусловие: необходим существующий ящик электронной почты.</p> <ol style="list-style-type: none"> 1) Нажать кнопку «Регистрация» на главной странице ПС. 2) Ввести адрес электронной почты. 3) Ввести пароль 12345678. 4) Ввести пароль из предыдущего пункта в поле подтверждения пароля. 5) Нажать кнопку «Зарегистрироваться». 6) Проверить ящик электронной почты, дождаться получения электронного письма. 7) Перейти по ссылке из полученного письма. <p>2. Аутентификация. Предусловие: необходим зарегистрированный в системе аккаунт.</p> <ol style="list-style-type: none"> 1) Нажать кнопку «Вход» на главной странице ПС. 2) Ввести адрес электронной почты и пароль аккаунта. 3) Нажать кнопку «Войти». 	<p>Отображается страница регистрации. На указанный адрес электронной почты приходит письмо со ссылкой. При переходе по ссылке появляется сообщения «Аккаунт подтвержден».</p>	Да

Продолжение таблицы 4.1

1	2	3	4
Аккаунт	<p>3. Редактирование профиля.</p> <p>Предусловие: необходимо произвести аутентификацию.</p> <ol style="list-style-type: none"> 1) Нажать кнопку «Мой профиль». 2) Нажать кнопку «Редактировать». 3) Изменить значения полей имени, фамилии, отчества, описания. 4) Обновить страницу. 5) Нажать кнопку «Мой профиль». 	<p>Открывается страница профиля пользователя.</p> <p>Открывается страница редактирования личной информации. В профиле отображается новая информация.</p>	Да
Роли	<p>4. Подача заявки.</p> <p>Предусловие: необходимо наличие зарегистрированного пользователя с ролью администратора факультета.</p> <ol style="list-style-type: none"> 1) Произвести регистрацию в системе. 2) Нажать кнопку «Подача заявки на роль студента». 3) Ввести номер группы, нажать кнопку «Отправить заявку». 4) Произвести аутентификацию в качестве пользователя с ролью администратора факультета. 5) Нажать кнопку «Заявки». 6) Нажать кнопку «Подтвердить». 	<p>Открывается главная страница приложения. Открывается окно подачи заявки. Отображается уведомление об отправке заявки.</p> <p>Заявка отображается у администратора факультета.</p> <p>После подтверждения заявки появляется соответствующее уведомление.</p>	Да

Продолжение таблицы 4.1

1	2	3	4
Расписание	<p>5. Отображение расписания преподавателя.</p> <p>Примечание: подтверждение роли администратором факультета не рассматривается.</p> <ol style="list-style-type: none"> 1) Произвести аутентификацию в системе. 2) Заполнить личную информацию пользователя в соответствии с некоторым существующим преподавателем. 3) Нажать кнопку «Подача заявки на роль преподавателя». 4) Выбрать кафедру, нажать кнопку «Отправить заявку». 5) Дождаться подтверждения заявки. 6) Нажать кнопку «Расписание». 	<p>Открывается главная страница приложения. Открывается окно подачи заявки. Отображается уведомление об отправке заявки. Отображается расписание преподавателя.</p>	Да
Сообщения	<p>6. Передача сообщений.</p> <p>Предусловие: необходимо наличие двух зарегистрированных пользователей.</p> <ol style="list-style-type: none"> 1) Произвести аутентификацию в системе. 2) Нажать кнопку «Сообщения», затем «Новое сообщение». 3) С помощью поля поиска найти второго пользователя, которому будет отправлено сообщение. 4) Ввести некоторый текст ненулевой длины, нажать кнопку «Отправить». 5) Произвести аутентификацию в качестве другого пользователя. 6) Нажать кнопку «Сообщения». 7) Выбрать диалог с первым пользователем. 	<p>Открывается экран диалогов. Открывается окно выбора пользователя для отправки сообщения. Есть возможность поиска и выбора пользователя. Созданное сообщение отображается отправленным.</p>	Да

Продолжение таблицы 4.1

1	2	3	4
Прогресс студента	<p>7. Отображение прогресса студента. Предусловие: наличие зарегистрированного пользователя с ролью студента, созданные для него группы индивидуальные занятия.</p> <ol style="list-style-type: none"> 1) Произвести аутентификацию в системе. 2) Нажать кнопку «Расписание». 3) Выбрать сроку с нелекционным занятием. 4) Выбрать индивидуальное задание. 5) Нажать кнопку «Предметы». 6) Выбрать сроку с некоторым предметом. 7) Выбрать индивидуальное задание. 	<p>Отображается расписание студента. В строках практических и лабораторных занятий с помощью специальных символов и цветов отображается прогресс выполнения заданий. В правой части отображается экран с деталями данного задания. Отображается список дисциплин студента. В его строках отображается прогресс выполнения заданий. В правой части отображается экран с деталями выбранного задания.</p>	Да

Продолжение таблицы 4.1

1	2	3	4
Материалы	<p>8. Редактирование материалов преподавателя.</p> <p>Предусловие: наличие зарегистрированного пользователя с ролью преподавателя.</p> <ol style="list-style-type: none"> 1) Произвести аутентификацию в системе. 2) Нажать кнопку «Предметы». 3) Выбрать предмет из списка. 4) Нажать кнопку «Добавить» в области материалов. 5) Выбрать файл материалов для загрузки. 6) Повторить добавление материалов не менее трех раз. 7) Нажать кнопку «Удалить» у одного из загруженных материалов. 	<p>Отображается список предметов преподавателя.</p> <p>В правой части отображается экран с детальной информацией о предмете.</p> <p>Открывается стандартный диалог загрузки файлов. Загруженные файлы немедленно отображаются в списке материалов.</p> <p>Удаленный материал немедленно исчезает.</p>	Да
Задания	<p>9. Создание индивидуальных заданий.</p> <p>Предусловие: наличие зарегистрированного пользователя с ролью преподавателя.</p> <ol style="list-style-type: none"> 1) Произвести аутентификацию в системе. 2) Нажать кнопку «Предметы». 3) Выбрать предмет из списка. 4) Нажать кнопку «Добавить» в области индивидуальных заданий. 5) Выбрать файл условия для загрузки. 	<p>Отображается список предметов преподавателя.</p> <p>В правой части отображается экран с детальной информацией о предмете.</p> <p>Открывается стандартный диалог загрузки файлов. Задание немедленно отображается в списке.</p>	Да

Продолжение таблицы 4.1

1	2	3	4
Результаты	<p>10. Отправка результатов выполнения заданий.</p> <p>Предусловие: наличие зарегистрированного пользователя с ролью студента и созданного для него задания.</p> <ol style="list-style-type: none"> 1) Произвести аутентификацию в системе. 2) Нажать кнопку «Предметы». 3) Выбрать предмет из списка. 4) В области "Результаты" нажать кнопку «Добавить». 5) Выбрать файл для загрузки. 6) Нажать кнопку «Отправить». 7) Дождаться подтверждения выполнения задания. 	<p>Отображается список предметов студента. В правой части отображается экран с детальной информацией о предмете. Открывается стандартный диалог загрузки файлов. После отправки и проверки преподавателем появляется оценка.</p>	Да
Результаты	<p>11. Проверка результатов выполнения заданий.</p> <p>Предусловие: наличие зарегистрированных пользователей с ролями преподавателя и студента. Студент отправил результаты не проверку.</p> <ol style="list-style-type: none"> 1) Произвести аутентификацию в системе. 2) Нажать кнопку «Очередь проверки». 3) Выбрать полученные результаты. 4) Выбрать и скачать полученный файл с результатами. 5) Выставить оценку. 	<p>Отображается список предметов преподавателя. В правой части отображается экран с детальной информацией о предмете. Скачивается файл с результатами выполнения заданий. Результаты помещаются как проверенные.</p>	Да

Таблица 4.2 – Тестовые случаи негативного тестирования

Модуль (экран)	Описание тестового случая	Ожидаемые результаты	Тестовый случай пройден?
1			
Аккаунт	<p>1. Повторная регистрация одного email.</p> <p>Предусловие: необходим существующий ящик электронной почты.</p> <ol style="list-style-type: none"> Произвести регистрацию в системе. Подтвердить email с помощью ссылки, полученной в электронном письме, отправленном на указанный адрес. Выйти из системы. Произвести регистрацию с тем же email. 	<p>Регистрация производится успешно, письмо приходит, при переходе по ссылке отображается сообщение об успешности подтверждения. При попытке регистрации с тем же email появляется сообщение о невозможности регистрации.</p>	Да
Аккаунт	<p>2. Аутентификации с неправильными данными.</p> <p>Предусловие: необходимо наличие зарегистрированного пользователя.</p> <ol style="list-style-type: none"> Нажать кнопку «Вход» на главной странице приложения. Ввести email, который заведомо не зарегистрирован в системе и некоторый пароль. В поле email ввести адрес электронной почты зарегистрированного пользователя. В поле пароля ввести заведомо неверный пароль. 	<p>Во всех случаях отображается одинаковое сообщение о неверном email или пароле.</p>	Да

Продолжение таблицы 4.2

1	2	3	4
Расписание	<p>3. Неверный диапазон дат для отображения расписания.</p> <p>Предусловие: необходим зарегстрированный аккаунт пользователя с ролью студента.</p> <ol style="list-style-type: none"> 1) Аутентифицироваться в системе. 2) Нажать кнопку «Расписание». 3) Нажать кнопку выбора начальной даты. 4) Выбрать дату 15.05.2017. 5) Нажать кнопку выбора конечной даты. 6) Выбрать дату 10.05.2017. 	<p>Отображается страница расписания занятий. Появляются выпадающие списки выбора границ диапазона дат. Появляется сообщение о неправильном выбранном интервале дат.</p>	Отображается главная страница приложения. Отображается страница главная страница приложения.
Роли	<p>4. Неверная группа в заявке студента.</p> <p>Предусловие: необходим зарегстрированный аккаунт пользователя.</p> <ol style="list-style-type: none"> 1) Аутентифицироваться в системе. 2) Нажать кнопку «Подача заявки на роль студента». 3) В качестве группы указать группу 999999. 4) Нажать кнопку «Отправить заявку». 5) Дождаться изменения статуса заявки. 	<p>Отображается главный экран приложения. Появляется окно создания заявки на роль студента. После просмотра заявки администратором факультета статус заявки будет изменен на "Отклонена поскольку такой группы не существует, что противоречит бизнес-правилам.</p>	Да

Продолжение таблицы 4.2

1	2	3	4
Аккаунт	<p>5. Заполнение личных данных неверной информацией.</p> <p>Предусловие: необходим существующий адрес электронной почты.</p> <ol style="list-style-type: none"> Произвести регистрацию в системе с произвольным паролем и существующим адресом электронной почты. Подтвердить адрес электронной почты путем перехода по ссылке из письма. Аутентифицироваться в системе. Нажать кнопку «Мой профиль». Нажать кнопку «Редактировать». Проверить, чтобы поля имени и фамилии были пустыми. Нажать кнопку «Сохранить». Ввести в поля имени и фамилии строк, содержащих знаки препинания и цифры. Нажать кнопку «Сохранить». 	<p>Регистрация проходит успешно. На указанный адрес электронной почты приходит письмо с подтверждающей ссылкой. Аутентификация происходит успешно.</p> <p>Открывается страница профиля. Открывается страница редактирования профиля. Появляется сообщение "Имя и фамилия не могут быть пустыми". Появляется сообщение "Введены некорректные имя и/или фамилия".</p>	Да
Аккаунт	<p>6. Аутентификация без подтверждения.</p> <p>Предусловие: необходим существующий адрес электронной почты.</p> <ol style="list-style-type: none"> Произвести регистрацию в системе с произвольным паролем и существующим адресом электронной почты. Нажать кнопку «Вход». Ввести указанные при регистрации email и пароль. 	<p>Регистрация происходит успешно. При аутентификации появляется сообщение "Данный аккаунт не подтвержден".</p>	Да

5 МЕТОДИКА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА

В данном разделе приведены основные сведения по работе с программным средством.

Приложение данного дипломного проекта (а точнее, его клиентская часть) не требует установки и настройки на конечных устройствах пользователя, поскольку представляет собой веб-приложение. Как и было заявлено в требованиях, для корректной работы программного средства необходим один из следующих браузеров с соответствующей минимальной версией:

- Google Chrome 49;
- Vivaldi 1.0;
- Opera 34;
- Mozilla Firefox 43;
- Apple Safari 9.0;
- Microsoft Edge 20.

Далее рассмотрены основные функции, предоставляемые приложением пользователям.

При открытии главной страницы приложения отображается экран расписания (рисунок 5.1). Для того, чтобы отобразить расписание уже прошедших занятий, следует нажать кнопку «Предыдущая неделя». Для выбора даты для отображения расписания следует нажать кнопку с пиктограммой календаря.

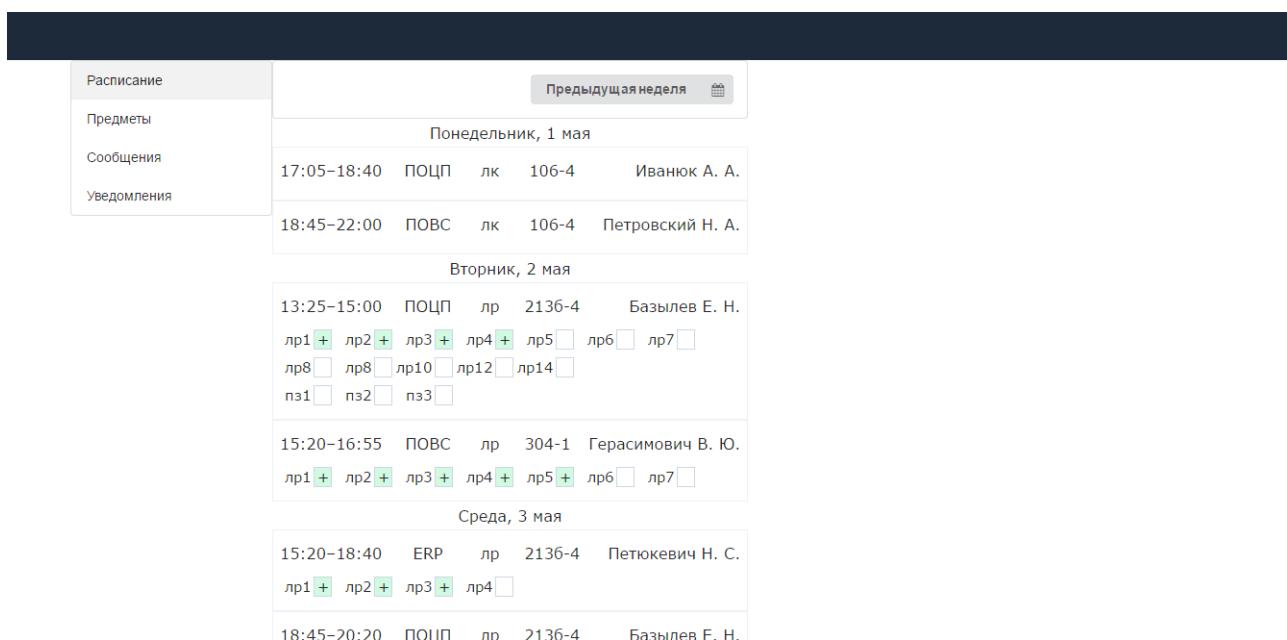


Рисунок 5.1 – Экран расписания

Далее, при нажатии на строку расписания в правой части приложения отображается страница с детальной информацией о предмете (рисунок 5.2). При наведении на его аббревиатуру отображается его полное название, а при наведении на фамилию преподавателя – его полное имя. На данной странице

можно просмотреть материалы по этому предмету, которые добавил преподаватель, список индивидуальных заданий. Кроме того, на данной странице есть возможность сразу написать некоторое сообщение преподавателю.

Расписание	Предыдущая неделя					ПОЦП	Программное обеспечение цифрового проектирования							
Предметы	Понедельник, 1 мая					Базылев Е. Н.								
Сообщения	Понедельник, 1 мая						Материалы							
Уведомления	17:05-18:40	ПОЦП	лк	106-4	Иванюк А. А.		Задания							
	18:45-22:00	ПОВС	лк	106-4	Петровский Н. А.		Сообщения							
	Вторник, 2 мая													
	13:25-15:00	ПОЦП	лр	2136-4	Базылев Е. Н.									
	лр1	+ <input checked="" type="checkbox"/>	лр2	+ <input checked="" type="checkbox"/>	лр3	+ <input checked="" type="checkbox"/>	лр4	+ <input checked="" type="checkbox"/>	лр5	<input type="checkbox"/>	лр6	<input type="checkbox"/>	лр7	<input type="checkbox"/>
	лр8	<input type="checkbox"/>	лр8	<input type="checkbox"/>	лр10	<input type="checkbox"/>	лр12	<input type="checkbox"/>	лр14	<input type="checkbox"/>				
	пз1	<input type="checkbox"/>	пз2	<input type="checkbox"/>	пз3	<input type="checkbox"/>								
	15:20-16:55	ПОВС	лр	304-1	Герасимович В. Ю.									
	лр1	+ <input checked="" type="checkbox"/>	лр2	+ <input checked="" type="checkbox"/>	лр3	+ <input checked="" type="checkbox"/>	лр4	+ <input checked="" type="checkbox"/>	лр5	+ <input checked="" type="checkbox"/>	лр6	<input type="checkbox"/>	лр7	<input type="checkbox"/>
	Среда, 3 мая													
	15:20-18:40	ERP	лр	2136-4	Петюкович Н. С.									
	лр1	+ <input checked="" type="checkbox"/>	лр2	+ <input checked="" type="checkbox"/>	лр3	+ <input checked="" type="checkbox"/>	лр4	<input type="checkbox"/>						
	18:45-20:20	ПОШП	лд	2136-4	Базылев Е. Н.									

Рисунок 5.2 – Экран расписания с открытой страницей предмета

Для открытия экрана со списком предметов (рисунок 5.3) следует нажать кнопку «Предметы» в главном меню, расположенном слева. На данной странице также есть возможность просмотра детальной информации о предмете, для этого необходимо выбрать один из списка.

Расписание	лр1	<input type="checkbox"/>	лр2	<input checked="" type="checkbox"/>	лр3	<input checked="" type="checkbox"/>	лр4	<input type="checkbox"/>	пз1	<input checked="" type="checkbox"/>	пз2	<input checked="" type="checkbox"/>	пз3	<input checked="" type="checkbox"/>
Предметы	САиММод	Мельник Н. И.			лк	144	ауд.ч.							
Сообщения	САиММод	Мельник Н. И.			лр	144	ауд.ч.							
Уведомления	лр1	+ <input checked="" type="checkbox"/>	лр2	+ <input checked="" type="checkbox"/>	лр3	<input type="checkbox"/>	лр4	<input type="checkbox"/>						
	ПОЦП	Иванюк А. А.			лк	126	ауд.ч.							
	ПОЦП	Базылев Е. Н.			лр	126	ауд.ч.							
	лр1	+ <input checked="" type="checkbox"/>	лр2	+ <input checked="" type="checkbox"/>	лр3	+ <input checked="" type="checkbox"/>	лр4	+ <input checked="" type="checkbox"/>	лр5	<input type="checkbox"/>	лр6	<input type="checkbox"/>	лр7	<input type="checkbox"/>
	лр8	<input type="checkbox"/>	лр8	<input type="checkbox"/>	лр10	<input type="checkbox"/>	лр12	<input type="checkbox"/>	лр14	<input type="checkbox"/>				
	пз1	<input type="checkbox"/>	пз2	<input type="checkbox"/>	пз3	<input type="checkbox"/>								
	ПОВС	Петровский Н. А.			лк	162	ауд.ч.							
	ПОВС	Герасимович В. Ю.			лр	162	ауд.ч.							
	лр1	+ <input checked="" type="checkbox"/>	лр2	+ <input checked="" type="checkbox"/>	лр3	+ <input checked="" type="checkbox"/>	лр4	+ <input checked="" type="checkbox"/>	лр5	+ <input checked="" type="checkbox"/>	лр6	<input type="checkbox"/>	лр7	<input type="checkbox"/>
	ERP	Мелких Е. Г.			лк	90	ауд.ч.							
	ERP	Петюкович Н. С.			лр	90	ауд.ч.							
	лр1	+ <input checked="" type="checkbox"/>	лр2	+ <input checked="" type="checkbox"/>	лр3	+ <input checked="" type="checkbox"/>	лр4	<input type="checkbox"/>						

Рисунок 5.3 – Экран предметов

Кроме того, на данном экране, а также на ранее рассмотренном экране есть возможность выбора и просмотра детальной информации об индивидуальных заданиях. Для этого нужно нажать соответствующую кнопку в строке

предмета. На странице задания (рисунок 5.4) можно просмотреть его условие, а также отправить результаты выполнения. Кроме того, на данной странице также есть возможность отправки сообщений преподавателю.

						Мельник Николай Иосифович		
	Расписание	ПиРИС	Хмелёва А. В.	лк	124 ауд.ч.	САиММод	Мельник Н. И.	
Предметы		ПиРИС	Хмелёва А. В.	лк	124 ауд.ч.	Лабораторная работа №4		
Сообщения		лр1 <input type="checkbox"/>	лр2 <input checked="" type="checkbox"/> 9	лр3 <input checked="" type="checkbox"/> 8	лр4 <input type="checkbox"/>	пз1 <input checked="" type="checkbox"/> 9	пз2 <input checked="" type="checkbox"/> 7	
Уведомления		САиММод	Мельник Н. И.	лк	144 ауд.ч.	Задание		
		лр1 <input checked="" type="checkbox"/> +	лр2 <input checked="" type="checkbox"/> +	лр3 <input checked="" type="checkbox"/> +	лр4 <input type="checkbox"/>	Построить аналитическую и имитационную модели и сравнить результаты исследования. 35) В СМО вида М/М/2/1 поступают заявки двух видов. Заявка первого вида появляется на входе с вероятностью p , второго с вероятностью $(1-p)$. Заявка первого вида имеет более высокий приоритет и может вытеснить заявку второго вида из канала в очередь, если место в очереди свободно или из системы, если место занято. В случае, когда заявка первого вида застает систему в состоянии обслуживания заявок первого вида, то она ставится в очередь, если место ожидания свободно или занято заявкой второго вида (менее приоритетная заявка теряется). Найти относительные пропускные способности $Q1$ и $Q2$. $=0,5$, $=0,9$, $p=0,5$.		
		ПОЦП	Иванюк А. А.	лк	126 ауд.ч.	Результаты		
		ПОЦП	Базылев Е. Н.	лр	126 ауд.ч.	Сообщения		
		лр1 <input checked="" type="checkbox"/> +	лр2 <input checked="" type="checkbox"/> +	лр3 <input checked="" type="checkbox"/> +	лр4 <input checked="" type="checkbox"/> +	лр5 <input type="checkbox"/>	лр6 <input type="checkbox"/>	
		лр8 <input type="checkbox"/>	лр8 <input type="checkbox"/>	лр10 <input type="checkbox"/>	лр12 <input type="checkbox"/>	лр14 <input type="checkbox"/>	лр7 <input type="checkbox"/>	
		пз1 <input type="checkbox"/>	пз2 <input type="checkbox"/>	пз3 <input type="checkbox"/>				
		ПОВС	Петровский Н. А.	лк	162 ауд.ч.			
		ПОВС	Герасимович В. Ю.	лр	162 ауд.ч.			
		лр1 <input checked="" type="checkbox"/> +	лр2 <input checked="" type="checkbox"/> +	лр3 <input checked="" type="checkbox"/> +	лр4 <input checked="" type="checkbox"/> +	лр5 <input checked="" type="checkbox"/> +	лр6 <input type="checkbox"/>	
							лр7 <input type="checkbox"/>	
		ERP	Мелких Е. Г.	лк	90 ауд.ч.			

Рисунок 5.4 – Экран предметов с открытой страницей задания

Таким образом, в данном разделе приведены примеры использования некоторых их основных возможностей разработанного программного средства. Следование им должно значительно упростить выполнение некоторых рутинных задач студентов и преподавателей.

6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И ВНЕДРЕНИЯ ПРОГРАММНОГО СРЕДСТВА

6.1 Характеристика программного средства

Программное средство, разрабатываемое в рамках дипломного проекта, предназначено для автоматизации основных задач участников учебного процесса университетов: управление расписанием, заданиями и коммуникацией. Существует большое число средств, с помощью которых могут решаться перечисленные задачи, однако данное многообразие приводит к разрозненности рабочих окружений разных людей. Вследствие ее возникновения у всех участников учебного процесса появляется необходимость в совершении рутинных действий по настройке своей рабочей среды. Кроме того, различия в используемых средствах приводят к необходимости дополнительных действий по соединению интерфейсов коммуникации. Данные проблемы могут быть решены путем создания системы, которая бы объединяла и унифицировала задачи управления событиями календаря, задачами и контактами.

Разработки проектов программных средств связана со значительными затратами ресурсов. В связи с этим создание и реализация каждого проекта программного обеспечения нуждается в соответствующем технико-экономическом обосновании [35], которое и описывается в данном разделе.

6.2 Определение объема и трудоемкости программного средства

Целесообразность создания ПС требует проведения предварительной экономической оценки. Экономический эффект у разработчика ПС зависит от объема инвестиций в разработку проекта, цены на готовый продукт и количества проданных копий, и проявляется в виде роста чистой прибыли.

Оценка стоимости создания ПС со стороны разработчика предполагает составление сметы затрат, вычисление цены и прибыли от реализации разрабатываемого программного средства.

Исходные данные, которые будут использоваться при расчете сметы затрат, представлены в таблице 6.1.

Перед определением сметы затрат на разработку программного средства необходимо определить его объем. Однако, на стадии ТЭО нет возможности рассчитать точные объемы функций, вместо этого с помощью применения действующих нормативов рассчитываются прогнозные оценки.

В качестве метрики измерения объема программных средств используется строка их исходного кода (LOC – lines of code). Данная метрика широко распространена, поскольку она непосредственно связана с конечным продуктом, может применяться на всем протяжении проекта и, кроме того, может использоваться для сопоставления размеров программного обеспечения. Далее

Таблица 6.1 – Исходные данные

Наименование показателя	Условное обозначение	Значение
Категория сложности		3
Дополнительный коэффициент сложности	$\sum_{i=1}^n K_i$	0,13
Степень охвата функций стандартными модулями	K_T	0,7
Коэффициент новизны	K_N	0,9
Количество дней в году	D_g	365
Количество праздничных дней в году	D_p	9
Количество выходных дней в году	D_v	103
Количество дней отпуска	D_o	21
Количество разработчиков	$Ч_p$	3
Тарифная ставка первого разряда, руб.	T_c^1	265
Среднемесячная норма рабочего времени, ч.	Φ_p	168,3
Продолжительность рабочей смены, ч.	T_c	8
Коэффициент премирования	K	1,3
Норматив дополнительной заработной платы	H_d	20%
Норматив отчислений в ФСЗН	H_{cz}	34%
Норматив отчислений по обязательному страхованию	H_{oc}	0,6%
Норматив расходов по статье «Материалы»	H_{mz}	5%
Норматив расходов по статье «Машинное время»	H_{mb}	15%
Понижающий коэффициент к статье «Машинное время»		0,5
Стоимость машино-часа, руб.	$Ц_m$	0,8
Норматив расходов по статье «Научные командировки»	H_{pk}	15%
Норматив расходов по статье «Прочие затраты»	H_{pz}	20%
Норматив расходов по статье «Накладные расходы»	H_{nr}	50%
Уровень рентабельности	Y_{pp}	10%
Ставка НДС	H_{dc}	20%
Норматив затрат на освоение ПО	H_o	10%
Норматив затрат на сопровождение ПО	H_c	20%

под строкой исходного кода будем понимать количество исполняемых операторов.

Расчет объема функций программного средства и общего объема приведен в таблице 6.2.

Таблица 6.2 – Перечень и объём функций программного модуля

№ функции	Наименование (содержание)	Объём функции, LoC
101	Организация ввода информации	100
102	Контроль, предварительная обработка и ввод информации	500
109	Организация ввода/вывода информации в интерактивном режиме	190
111	Управление вводом/выводом	2600
204	Обработка наборов и записей базы данных	1900
207	Манипулирование данными	8000
208	Организация поиска и поиск в БД	7500
304	Обслуживание файлов	500
305	Обработка файлов	800
309	Формирование файла	1000
506	Обработка ошибочных и сбойных ситуаций	500
507	Обеспечение интерфейса между компонентами	750
601	Отладка прикладных программ в интерактивном режиме	4300
707	Графический вывод результатов	300
	Общий объем	28 940

Исходя из определенной 3-ей категории сложности и общего объема ПС $V_o = 28\ 940$, нормативная трудоемкость $T_n = 520$ чел./д. [35, приложение 3]. Перед определением общей трудоемкости разработки необходимо определить несколько коэффициентов.

Коэффициент сложности, который учитывает дополнительные затраты труда, связанные с обеспечением интерактивного доступа и хранения, и поиска данных в сложных структурах [35, приложение 4, таблица П.4.2]

$$K_c = 1 + \sum_{i=1}^n K_i = 1 + 0,06 + 0,07 = 1,13, \quad (6.1)$$

где K_i – коэффициент, соответствующий степени повышения сложности за счет конкретной характеристики;
 n – количество учитываемых характеристик.

Коэффициент K_t , учитывающий степень использования при разработке стандартных модулей, для разрабатываемого приложения, в котором степень охвата планируется на уровне около 50%, примем равным 0,7 [35, приложение 4, таблица П.4.5].

Коэффициент новизны разрабатываемого программного средства K_n примем равным 0,9, так как разрабатываемое ПС принадлежит определенному па-

раметрическому ряду существующих программных средств [35, приложение 4, таблица П.4.4].

Исходя из выбранных коэффициентов, общая трудоемкость разработки $T_o = T_h \cdot K_c \cdot K_t \cdot K_n = 520 \cdot 1,13 \cdot 0,7 \cdot 0,9 = 370$ чел./д.

Для расчета срока разработки проекта примем число разработчиков $Ч_p = 3$. Исходя из комментария к постановлению Министерства труда и социальной защиты Республики Беларусь от 05.10.16 №54 «Об установлении расчетной нормы рабочего времени на 2017 год» [36], эффективный фонд времени работы одного человека составит

$$\Phi_{\text{эфф}} = D_r - D_{\text{п}} - D_{\text{в}} - D_o = 365 - 9 - 103 - 21 = 232 \text{ д.}, \quad (6.2)$$

где D_r — количество дней в году;

$D_{\text{п}}$ — количество праздничных дней в году;

$D_{\text{в}}$ — количество выходных дней в году;

D_o — количество дней отпуска.

Тогда трудоемкость разработки проекта

$$T_p = \frac{T_o}{Ч_p \cdot \Phi_{\text{эфф}}} = \frac{370}{3 \cdot 232} = 0,53 \text{ г.} = 194 \text{ д.} \quad (6.3)$$

Исходя из того, что разработкой будет заниматься 3 человека, можно запланировать фонд рабочего времени для каждого исполнителя

$$\Phi_{\text{ни}} = \frac{T_p}{Ч_p} = \frac{194}{3} \approx 65 \text{ д.} \quad (6.4)$$

6.3 Расчет сметы затрат

Основной статьей расходов на создание ПО является заработка плата разработчиков проекта. Информация об исполнителях перечислена в таблице 6.3. Кроме того, в таблице приведены данные об их тарифных разрядах, приведены разрядные коэффициенты, а также по формулам 6.5 и 6.6 рассчитаны месячный и часовой оклады.

Таблица 6.3 – Работники, занятые в проекте

Исполнители	Разряд	Тарифный коэффициент	Месячный оклад, руб.	Часовой оклад, руб.
Руководитель проекта	17	3,98	1054,70	6,27
Ведущий инженер-программист	15	3,48	922,20	5,48
Инженер-программист II категории	11	2,65	702,25	4,17

$$T_m = T_m^1 \cdot T_k, \quad (6.5)$$

$$T_q = \frac{T_m}{\Phi_p}, \quad (6.6)$$

где T_m — месячный оклад;

T_m^1 — тарифная ставка 1-го разряда (положим ее равной 265 руб.);

T_k — тарифный коэффициент;

T_q — часовой оклад;

Φ_p — среднемесячная норма рабочего времени (в 2017 г. составляет 168,3 ч. [36]).

Тогда основная заработная плата исполнителей составит

$$\begin{aligned} Z_o &= \sum_{i=1}^n T_{qi} \cdot T_q \cdot \Phi_{pi} \cdot K = \\ &= (6,27 + 5,48 + 4,17) \cdot 8 \cdot 65 \cdot 1,3 = 10761,92 \text{ руб.}, \end{aligned} \quad (6.7)$$

где T_{qi} — часовая тарифная ставка i -го исполнителя, руб.;

T_q — количество часов работы в день;

Φ_{pi} — плановый фонд рабочего времени i -го исполнителя, д.;

K — коэффициент премирования (принятый равным 1,3).

Дополнительная заработная плата включает выплаты, предусмотренные законодательство о труде: оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей, и определяется по нормативу, установленному в организации, в процентах к основной заработной плате. Приняв данный норматив $H_d = 20\%$, рассчитаем дополнительные выплаты

$$Z_d = \frac{Z_o \cdot H_d}{100\%} = \frac{10761,92 \cdot 20\%}{100\%} = 2152,38 \text{ руб.} \quad (6.8)$$

Отчисления в фонд социальной защиты населения и в фонд обязательного страхования определяются в соответствии с действующим законодательством по нормативу в процентном отношении к фонду основной и дополнительной зарплат по следующим формулам

$$\begin{aligned} Z_{cz} &= \frac{(Z_o + Z_d) \cdot H_{cz}}{100\%}, \\ Z_{oc} &= \frac{(Z_o + Z_d) \cdot H_{oc}}{100\%}. \end{aligned} \quad (6.9)$$

В настоящее время нормы отчислений в ФСЗН $H_{cz} = 34\%$ и в фонд

обязательного страхования $H_{oc} = 0,6\%$. Исходя из этого, размеры отчислений

$$Z_{cz} = \frac{(10\,761,92 + 2152,38) \cdot 34\%}{100\%} = 4390,86 \text{ руб.},$$

$$Z_{oc} = \frac{(10\,761,92 + 2152,38) \cdot 0,6\%}{100\%} = 77,49 \text{ руб.} \quad (6.10)$$

Расходы по статье «Материалы» отражают траты на магнитные носители, бумагу, красящие материалы, необходимые для разработки ПО определяются по нормативу к фонду основной заработной платы разработчиков. Исходя из принятого норматива $H_{mz} = 5\%$ определим величину расходов

$$M = \frac{Z_o \cdot H_{mz}}{100\%} = \frac{10\,761,92 \cdot 5\%}{100\%} = 538,10 \text{ руб.} \quad (6.11)$$

Расходы по статье «Машинное время» включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам на 100 строк исходного кода. Норматив зависит от характера решаемых задач и типа ПК; для текущего проекта примем $H_{mb} = 15\%$ [35, приложение 6]. Примем величину стоимости машино-часа $\Pi_m = 0,8$ руб. Тогда, применяя понижающий коэффициент 0,5, получим величину расходов

$$P_m = \Pi_m \cdot \frac{V_o}{100} \cdot H_{mb} = 0,8 \cdot \frac{28\,940}{100} \cdot 15\% \cdot 0,5 = 1736,40 \text{ руб.} \quad (6.12)$$

Расходы по статье «Научные командировки» определяются по нормативу в процентах к основной заработной плате. Принимая норматив равным $H_{phk} = 15\%$ получим величину расходов

$$P_{hk} = \frac{Z_o \cdot H_{phk}}{100\%} = \frac{10\,761,92 \cdot 15\%}{100\%} = 1614,29 \text{ руб.} \quad (6.13)$$

Расходы по статье «Прочие затраты» включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу в процентах к основной заработной плате. Принимая норматив равным $H_{pz} = 20\%$ получим величину расходов

$$\Pi_3 = \frac{Z_o \cdot H_{pz}}{100\%} = \frac{10\,761,92 \cdot 20\%}{100\%} = 2152,38 \text{ руб.} \quad (6.14)$$

Затраты по статье «Накладные расходы», связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды, относятся к конкретному ПО по нормативу в процентном отношении к основной заработной плате исполнителей. Принимая норматив равным

$H_{hp} = 50\%$ получим величину расходов

$$P_h = \frac{Z_o \cdot H_{hp}}{100\%} = \frac{10761,92 \cdot 50\%}{100\%} = 5380,96 \text{ руб.} \quad (6.15)$$

Общая сумма расходов по смете определяется как сумма вышерассчитанных показателей

$$C_{\pi} = Z_o + Z_d + Z_{c3} + Z_{oc} + M + P_m + P_{hk} + \Pi_3 + P_h = \\ = 28804,78 \text{ руб.} \quad (6.16)$$

Рентабельность определяется из результатов анализа рыночных условий и переговоров с потребителями ПО. Исходя из принятого уровня рентабельности $Y_{pp} = 10\%$, прибыль от реализации ПО составит

$$\Pi_o = \frac{C_{\pi} \cdot Y_{pp}}{100\%} = \frac{28804,78 \cdot 10\%}{100\%} = 2880,48 \text{ руб.} \quad (6.17)$$

На основании расчета прибыли и уровня себестоимости рассчитаем прогнозируемую цену программного средства без учета налогов

$$\Pi_{\pi} = C_{\pi} + \Pi_o = 28804,78 + 2880,48 = 31685,26 \text{ руб.} \quad (6.18)$$

Далее рассчитаем налог на добавленную стоимость

$$НДС = \frac{\Pi_{\pi} \cdot H_{dc}}{100\%} = \frac{31685,26 \cdot 20\%}{100\%} = 6337,05 \text{ руб.} \quad (6.19)$$

НДС включается в прогнозируемую отпускную цену

$$\Pi_o = \Pi_{\pi} + НДС = 31685,26 + 6337,05 = 38022,31 \text{ руб.} \quad (6.20)$$

Организация-разработчик участвует в освоении и внедрении ПО и несет соответствующие затраты, которые определяются по нормативу $H_o = 10\%$ от себестоимости ПО в расчете на три месяца

$$P_o = \frac{C_{\pi} \cdot H_o}{100\%} = \frac{28804,78 \cdot 10\%}{100\%} = 2880,48 \text{ руб.} \quad (6.21)$$

Кроме того, организация-разработчик осуществляет сопровождение ПО, которое также оплачивается заказчиком. Расчет осуществляется в соответствии с нормативом $H_c = 20\%$ от себестоимости ПО

$$P_c = \frac{C_{\pi} \cdot H_c}{100\%} = \frac{28804,78 \cdot 20\%}{100\%} = 5760,96 \text{ руб.} \quad (6.22)$$

Экономическим эффектом разработчика будет являться сумма прибыли с

вычетом налога на прибыль

$$\Pi_{\text{ч}} = \Pi_{\text{o}} - \frac{\Pi_{\text{o}} \cdot H_{\text{п}}}{100\%} = 2880,48 - \frac{2880,48 \cdot 18\%}{100\%} = 2361,99 \text{ руб.} \quad (6.23)$$

6.4 Оценка экономической эффективности применения программного средства у пользователя

В результате применения нового ПО пользователь может понести значительные капитальные затраты на приобретение и освоение ПО, доукомплектованием ЭВМ новыми техническими средствами и пополнение оборотных средств. Однако, если приобретенное ПО будет в достаточной степени эффективнее базового, то дополнительные капитальные затраты быстро окупятся.

Для определения экономического эффекта от использования нового ПО у потребителя необходимо сравнить расходы по всем основным статьям сметы затрат на эксплуатацию нового ПО с расходами по соответствующим статьям базового варианта. При этом за базовый вариант примем ручной вариант. Исходные данные для расчета приведены в таблице 6.4. Размер средней зарплаты приведен на февраль 2017 г.

Таблица 6.4 – Исходные данные

Наименование показателя	Условное обозначение	Значение в базовом варианте	Значение в новом варианте
Затраты пользователя на приобретение ПО	K _{пр}	—	38 022,31 руб.
Затраты пользователя на освоение	K _{ос}	—	2880,48 руб.
Затраты пользователя на сопровождение	K _с	—	5760,96 руб.
Трудоемкость на задачу, чел./ч.	T _{c1, T_{c2}}	2	0,1
Средняя зарплата, руб.	Z _{см}	716,5	716,5
Количество выполняемых задач	A _{1, A₂}	1460	1460
Время простоя сервиса, мин. в день	P _{1, P₂}	50	10
Стоимость одного часа простоя, руб.	C _п	79,8	79,8

Общие капитальные вложения заказчика (потребителя) вычисляются следующим образом

$$K_{\text{o}} = K_{\text{пр}} + K_{\text{ос}} + K_{\text{с}} = 38 022,31 + 2880,48 + 5760,96 = \\ = 46 663,75 \text{ руб.} \quad (6.24)$$

где $K_{\text{пр}}$ — затраты пользователя на приобретение ПО по отпускной цене;
 $K_{\text{ос}}$ — затраты пользователя на освоение;
 K_c — затраты пользователя на оплату услуг по сопровождению.

В качестве типичного примера использования разрабатываемого ПС предполагается сценарий ее использования ежедневно 4 раза в день, при этом предполагается снижение трудоемкости с $T_{c1} = 2$ чел./ч. до $T_{c2} = 0,1$ чел./ч.. Приняв среднемесячную заработную плату работника $Z_{\text{см}} = 716,5$ руб., рассчитаем экономию затрат на заработную плату в расчете на одну задачу $C_{\text{зe}}$ и за год C_3

$$C_{\text{зe}} = \frac{Z_{\text{см}} \cdot (T_{c1} - T_{c2})}{\Phi_p} = \frac{716,5 \cdot (2 - 0,1)}{168,3} = 8,09 \text{ руб.}, \quad (6.25)$$

$$C_3 = C_{\text{зe}} \cdot A_2 = 8,09 \cdot 1460 = 11811,4 \text{ руб.}, \quad (6.26)$$

где Φ_p — среднемесячная норма рабочего времени, ч.

Экономия с учетом начислений на зарплату

$$C_h = C_3 \cdot \frac{100\% + K}{100\%} = 11811,4 \cdot \frac{100\% + 1,3}{100\%} = 11964,95 \text{ руб.}, \quad (6.27)$$

где K — норматив начислений на зарплату, руб.

Экономия за счет сокращения простоев сервиса

$$C_c = \frac{(\Pi_1 - \Pi_2) \cdot D_{\text{пр}} \cdot C_{\text{п}}}{60} = \frac{(50 - 10) \cdot 300 \cdot 79,8}{60} = 15960,00, \text{ руб.}, \quad (6.28)$$

где $D_{\text{пр}}$ — плановый фонд работы сервиса, д.

Тогда общая годовая экономия текущих затрат, связанных с использованием нового ПО

$$C_o = C_h + C_c = 11964,95 + 15960,00 = 27924,95 \text{ руб.} \quad (6.29)$$

Внедрение нового ПО позволит пользователю сэкономить на текущих затратах, то есть получить на эту сумму дополнительную прибыль. Для пользователя в качестве экономического эффекта выступает лишь чистая прибыль. Принимая размер ставки налога на прибыль $H_{\text{п}} = 18\%$ получим

$$\Delta\Pi_q = C_o - \frac{C_o \cdot H_{\text{п}}}{100\%} = 27924,95 - \frac{27924,95 \cdot 18\%}{100\%} = 22898,46 \text{ руб.} \quad (6.30)$$

В процессе использования нового ПО чистая прибыль в конечном итоге возмещает капитальные затраты. Однако полученные при этом суммы прибыли

и затрат по годам приводят к единому времени – расчетному году (за расчетный год принят 2017-й год) путем умножения результатов и затрат за каждый год на коэффициент дисконтирования α . При расчете используются следующие коэффициенты: 2017 г. – 1, 2018 г. – 0,8696, 2019 г. – 0,7561, 2020 г. – 0,6575. Все рассчитанные данные экономического эффекта сведены в таблицу 6.5.

В результате технико-экономического обоснования применения ПС были получены следующие значения показателей эффективности:

- чистая прибыль разработчика составит $\Pi_{\text{ч}} = 2361,99$ руб.;
- затраты заказчика окупятся уже на четвертом году использования;
- экономическая эффективность для заказчика, выраженная в виде чистого дисконтированного дохода, составит 5618,02 руб. за четыре года использования данного ПС; более высокий прирост прибыли заказчик получит по истечению данного срока.

Таблица 6.5 – Расчет экономического эффекта от использования нового ПО

Название показателя	2017 г. 1	2018 г. 2	2019 г. 3	2020 г. 4
<i>Результаты</i>				
Коэффициент приведения	1	0,8696	0,7561	0,6575
Прирост прибыли за счет экономии затрат ($\Pi_{\text{ч}}$), руб.	–	22 898,46	22 898,46	22 898,46
То же с учетом фактора времени, руб.	–	19 912,50	17 313,53	15 055,74
<i>Затраты</i>				
Приобретение ПО ($K_{\text{пр}}$), руб.	38 022,31	–	–	–
Освоение ПО ($K_{\text{ос}}$), руб.	2880,48	–	–	–
Сопровождение ПО ($K_{\text{с}}$), руб.	5760,96	–	–	–
Всего затрат (K_{o}), руб.	46 663,75	–	–	–
То же с учетом фактора времени, руб.	46 663,75	–	–	–
<i>Экономический эффект</i>				
Превышение результата над затратами, руб.	–46 663,75	19 912,5	17 313,53	15 055,74
То же с нарастающим итогом, руб.	–46 663,75	–26 751,25	–9437,72	5618,02

Полученные результаты свидетельствуют об эффективности разработки и внедрения проектируемого программного средства.

ЗАКЛЮЧЕНИЕ

Предметной областью данного дипломного проекта является информатизация некоторых задач участников учебного процесса: студентов и преподавателей. Был проведен поиск существующих программные средства этого рода, по его результатам был сделан вывод о несуществовании полных аналогов. Студенты и преподаватели адаптируют различные средства для использования в профессиональной деятельности; были выявлены недостатки данного подхода. Было предложено программное средство, которое должно унифицировать используемые средства и подходы в процессе обучения касательно задач управления временем, задачами и коммуникаций.

На основании проведенного анализа предметной области были выдвинуты требования к программному средству. В качестве технологий разработки были выбраны наиболее современные существующие на данный момент средства, широко применяемые в индустрии. Спроектированное программное средство было успешно протестировано на соответствие спецификации функциональных требований. Уже исходя только из анализа предметной области и факта несуществования полных аналогов можно было сделать вывод о целесообразности проектирования и разработки программной системы. Результаты, полученные в ходе выполнения технико-экономического обоснования только подтвердили данный вывод.

Разработано программное средство, целевой платформой которого является веб-приложение и которое поддерживает следующие функции:

- отображение расписания занятий;
- создание и управление индивидуальными заданиями и материалами;
- отображение прогресса выполнения заданий по предметам (для студентов) и прогресса выполнения заданий по всем людям (для преподавателей);
- отправка результатов выполнения заданий;
- возможность оценивания результатов выполнения преподавателем;
- обмен сообщениями между пользователями системы;
- подтверждение студентов и преподавателей с помощью введения специальной роли администратора факультета.

Следующая основная цель – внедрение и популяризация программного средства среди преподавателей (в первую очередь) и студентов. Параллельно с этим будет производиться дальнейшая его разработка. Будет внедрена поддержка сессии: экзаменов и зачётов, – с формированием виртуальной зачетки студента. Кроме того, будут внедряться новые функции, например: автоматизация проверки посещаемости с использованием геолокации и QR-кодов, автоматизация проверки результатов выполнения заданий по формальным признакам, создание расписания занятий для всех групп, поддержка различных типов занятий, включая разовые и дополнительные консультации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] ISTQB glossary – Specification [Электронный ресурс]. — Электронные данные. — Режим доступа: <http://glossary.astqb.org/search/specification>. — Дата доступа: 25.03.17.
- [2] Как правильно переходить границу: кроссплатформенность в мобильном приложении [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://habrahabr.ru/company/parallels/blog/254325/>. — Дата доступа: 26.03.17.
- [3] Stollman, Richard. What is free software? [Электронный ресурс]. — Электронные данные. — Режим доступа: <http://www.gnu.org/philosophy/free-sw.en.html>. — Дата доступа: 09.04.17.
- [4] Тайм-менеджмент. Полный курс : Учебное пособие / Г. А. Архангельский, М. А. Лукашенко, Т. В. Телегина, С. В. Бехтерев. — М. : Альпина Паблишер, 2012.
- [5] 6 key elements for better Task Management [Электронный ресурс]. — Электронные данные. — Режим доступа: <http://www.techrepublic.com/article/6-key-elements-for-better-task-management/>. — Дата доступа: 21.03.17.
- [6] Макконнелл, С. Совершенный код. Мастер-класс / Пер. с англ. / С. Макконнелл. — М. : Издательско-торговый дом «Русская редакция», 2010. — 896 с.
- [7] Sheriff, Paul D. Designing for Web or Desktop? [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://msdn.microsoft.com/en-us/library/ms973831.aspx>. — Дата доступа: 27.03.17.
- [8] Shaaban, Sam. Web-Based vs “Desktop” Software [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://nurelm.com/web-based-vs-desktop-software/>. — Дата доступа: 27.03.17.
- [9] Bychkov, Dmitriy. Desktop vs. Web Applications: A Deeper Look and Comparison [Электронный ресурс]. — Электронные данные. — Режим доступа: <http://www.seguetech.com/desktop-vs-web-applications/>. — Дата доступа: 27.03.17.
- [10] Summerfield, Jason. Mobile Website vs. Mobile App: Which is Best for Your Organization? [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://www.hwsolutions.com/services/mobile-web-development/mobile-website-vs-apps/>. — Дата доступа: 28.03.17.
- [11] Garlan, David. An Introduction to Software Architecture / David Garlan, Mary Shaw / Ed. by V Ambriola, G Tortora. — New Jersey : World Scientific Publishing Company, 1994. — Vol. I. — Режим доступа: http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf. — Дата доступа: 28.03.17.
- [12] Волосевич, А. А. Архитектура программного обеспечения: Курс лекций для студентов специальности 1-40 01 03 Информатика и технологии программирования / А. А. Волосевич. — Минск : БГУИР, 2013.

[13] Team, Microsoft Patterns & Practices. Microsoft® Application Architecture Guide (Patterns & Practices) / Microsoft Patterns & Practices Team. — Microsoft Press, 2009.

[14] Marinescu, Floyd. Domain-Driven Design Quickly / Floyd Marinescu, Abel Avram. — LULU PR, 2007.

[15] Куликов, С. С. Базы данных. Рабочая тетрадь. / С. С. Куликов. — Минск .

[16] Ершов, Александр. Сила бумажных записных книжек [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://habrahabr.ru/post/89572/>. — Дата доступа: 31.03.17.

[17] McAuley, Alexander. The MOOC model for digital practice [Электронный ресурс]. — Электронные данные. — Режим доступа: http://davecormier.com/edblog/wp-content/uploads/MOOC_Final.pdf. — Дата доступа: 01.04.17.

[18] Moodle, как платформа организации eLearning и дистанционного обучения [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://habrahabr.ru/post/139629/>. — Дата доступа: 07.05.17.

[19] Software Cost Estimation with Cocomo II / Barry W. Boehm [et al.]. — Prentice Hall, 2000.

[20] Walston, Claude E. A Method of Programming Measurement and Estimation. / Claude E. Walston, Charles P. Felix // IBM Systems Journal. — 1977. — Vol. 16, no. 1. — Pp. 54–73.

[21] TypeScript [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://www.typescriptlang.org>. — Дата доступа: 26.03.17.

[22] Modern Apps - Use TypeScript in Modern Apps [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://msdn.microsoft.com/en-us/magazine/dn201754.aspx>. — Дата доступа: 26.03.17.

[23] NativeScript [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://www.nativescript.org>. — Дата доступа: 26.03.17.

[24] Общие сведения о платформе .NET Framework [Электронный ресурс]. — Электронные данные. — Режим доступа: [https://msdn.microsoft.com/ru-ru/library/zw4w595w\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/zw4w595w(v=vs.110).aspx). — Дата доступа: 09.04.17.

[25] Введение в язык C# и .NET Framework [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://msdn.microsoft.com/ru-ru/library/z1zx9t92.aspx>. — Дата доступа: 09.04.17.

[26] МакЛафлин, Бретт. От Web-сайтов к Web-приложениям: Часть 1. Web-сайт или Web-приложение? [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://www.ibm.com/developerworks/ru/library/web-websiteapp/>. — Дата доступа: 16.04.17.

[27] DefinitelyTyped [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://github.com/DefinitelyTyped/DefinitelyTyped>. — Дата до-

ступа: 23.04.17.

[28] Краткое руководство по React JS [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://habrahabr.ru/post/248799/>. — Дата доступа: 23.04.17.

[29] React. A javascript library for building user interfaces [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://facebook.github.io/react/>. — Дата доступа: 23.04.17.

[30] Abramov, Dan. Presentational and Container Components [Электронный ресурс]. — Электронные данные. — Режим доступа: https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0. — Дата доступа: 23.04.17.

[31] Жуков, Антон. Стилизация React-компонентов [Электронный ресурс]. — Электронные данные. — Режим доступа: habrahabr.ru/company/devexpress/blog/283314/. — Дата доступа: 24.04.17.

[32] Основы Sass [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://sass-scss.ru/guide/>. — Дата доступа: 24.04.17.

[33] Best Practices for building large scale maintainable projects [Электронный ресурс]. — Электронные данные. — Режим доступа: <https://mobx.js.org/best/store.html>. — Дата доступа: 27.04.17.

[34] Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. — ЕРАМ Systems, RD Dep., 2017. — http://svyatoslav.biz/software_testing_book/.

[35] Палицын, В.А. Технико-экономическое обоснование дипломных проектов: Метод. пособие для студ. всех спец. БГУИР. В 4-х ч. Ч. 4: Проекты программного обеспечения / В.А. Палицын. — Минск : БГУИР, 2006. — 76 с.

[36] Пещенко, Е.А. Производственный календарь на 2017 год [Электронный ресурс]. — Электронные данные. — Режим доступа: <http://www.mintrud.gov.by/system/extensions/spaw/uploads/files/Kommetarij-2017-RV.pdf>. — Дата доступа: 06.04.17.

ПРИЛОЖЕНИЕ А

(обязательное)

Исходный код

```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="../node_modules/semantic-ui-css/semantic.min.css"
        />
</head>
<body>
    <div id="root"></div>

<!-- Dependencies -->
<script src="../node_modules/react/dist/react.js"></script>
<script src="../node_modules/react-dom/dist/react-dom.js"></script>

<!-- Main -->
<script src="../dist/bundle.js"></script>
</body>
</html>

// index.tsx
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Router, hashHistory, Route, IndexRedirect } from "react-router";

import { AppRouter } from "components/App";

import "styles/index.scss";

ReactDOM.render(
    <AppRouter/>,
    document.getElementById("root")
);

// App.tsx
import * as React from "react";
import { observer } from "mobx-react";
import { observable } from "mobx";
import { Link } from "react-router";
import { Router, hashHistory, Route, IndexRedirect } from "react-router";
import { Menu } from "semantic-ui-react";

import { Agenda } from "components/agenda/Agenda";
import { Disciplines } from "components/disciplines/Disciplines";

import "styles/app.scss";

@observer
export class AppRouter extends React.Component<{}, {}>{
    render() {
        return (
            <Router history={hashHistory}>
                <Route path="/" component={App}>
                    <IndexRedirect to="agenda" />
                </Route>
            </Router>
        );
    }
}
```

```

        <Route path="agenda" component={Agenda} />
        <Route path="disciplines" component={Disciplines} />
    </Route>
</Router>
);
}
}

@observer
export class App extends React.Component<{}, {}> {
    render() {
        return (
            <div id="app">
                <header id="main-header">
                    {/*this.props.header*/}
                </header>
                <div id="app-body">
                    <Menu vertical borderless>
                        <Menu.Item as={Link} to="/agenda" activeClassName="active"></
                            Menu.Item>
                        <Menu.Item as={Link} to="/disciplines" activeClassName="active">
                        </Menu.Item>
                        <Menu.Item as={Link} to="" activeClassName="active"></Menu.Item>
                        >
                        <Menu.Item as={Link} to="" activeClassName="active"></Menu.Item>
                        >
                    </Menu>
                    <section id="app-content">{this.props.children}</section>
                </div>
            </div>
        );
    }
}

// TwoSidePanel.tsx
import * as React from "react";
import Scrollbars from "react-custom-scrollbars";

import "styles/panels.scss";

export interface ITwoSidePanelProps {
    left?: React.ReactNode;
    right?: React.ReactNode;
}

export class TwoSidePanel extends React.Component<ITwoSidePanelProps, {}> {
    render() {
        return (
            <div className="two-side-panel">
                <div className="left-side">{this.props.left}</div>
                <Scrollbars autoHide>
                    <div className="right-side">{this.props.right}</div>
                </Scrollbars>
            </div>
        );
    }
}

// Agenda.tsx

```

```

import * as React from "react";
import { observer } from "mobx-react";

import { TwoSidePanel } from "components/TwoSidePanel";
import { AgendaView } from "components/agenda/AgendaView";

import uiStore from "stores/UiStore";

import "styles/agenda/agenda.scss";

@observer
export class Agenda extends React.Component<{}, {}> {
    render() {
        return (
            <div id="agenda">
                <TwoSidePanel left={<AgendaView />} right={uiStore.secondaryPage} />
            </div>
        );
    }
}

// AgendaView.tsx
import * as React from "react";
import { observer } from "mobx-react";
import Scrollbars from "react-custom-scrollbars";
import { Segment, Button } from "semantic-ui-react";

import { AgendaLine, AgendaLineDate } from "components/agenda/AgendaLine";
import agendaStore from "stores/AgendaStore";

import { dateCompare, areDifferentDays } from "utils/timeutils";
import "styles/agenda/agenda-view.scss";

@observer
export class AgendaView extends React.Component<{}, {}> {
    render() {
        return (
            <div id="agenda-view">
                <Segment>
                    <Button.Group compact floated="right">
                        <Button compact> </Button>
                        <Button icon="calendar" compact></Button>
                    </Button.Group>
                </Segment>
                {this.renderAgendaList()}
            </div>
        );
    }

    renderAgendaList() {
        return (
            <Scrollbars autoHide>
                <div id="agenda-list">
                    {
                        () => {
                            let agendaItems = agendaStore.agendaItems
                                .sort((a, b) => dateCompare(a.startDate, b.startDate));
                    }
                </div>
            </Scrollbars>
        );
    }
}

```

```

        let agendaLines: React.ReactNode[] = [];
        for (let i = 0; i < agendaItems.length; i++) {
            if (!agendaItems[i - 1] || areDifferentDays(agendaItems[i - 1].startDate, agendaItems[i].startDate))
                agendaLines.push(<AgendaLineDate date={agendaItems[i].startDate} key={i + agendaItems.length} />);
            agendaLines.push(<AgendaLine item={agendaItems[i]} key={i} />);
        }
        return agendaLines;
    })()
}
</div>
</Scrollbars>
);
}
}

// AgendaLine.tsx
import * as React from "react";
import { observer } from "mobx-react";
import { formatDate } from "utils/timeutils";

import { AgendaItem } from "models/agenda";
import { Task } from "models/tasks";
import { TaskLine } from "components/elements/TasksLine";

import uiStore from "stores/UiStore";

import "styles/agenda/agenda-line.scss";

@observer
export class AgendaLine extends React.Component<{ item: AgendaItem }, {}> {
    render() {
        return (
            <div className="agenda-item" onClick={(e) => this.onClickAgendaItem(this.props.item)}>
                <div className="agenda-item-info">
                    <span className="agenda-item-time">{this.props.item.formatTimeSlot()}</span>
                    <span className="agenda-item-subject">{this.props.item.subject.abbreviation}</span>
                    <span className="agenda-item-subject-type">{this.props.item.subject.type}</span>
                    <span className="agenda-item-auditorium">{this.props.item.auditorium}</span>
                    <span className="agenda-item-lecturer">{this.props.item.subject.tutor.getReducedName()}</span>
                </div>
                <div className="agenda-item-status">
                    {
                        this.props.item.subject.tasks ?
                            <TaskLine tasks={this.props.item.subject.tasks} /> :
                            null
                    }
                </div>
            </div>
        );
    }
}

```

```

        onClickAgendaItem(agendaItem: AgendaItem) {
            uiStore.selectSubject(agendaItem.subject);
        }
    }

export class AgendaLineDate extends React.Component<{ date: Date }, {}> {
    render() {
        return (
            <div className="agenda-line-date">
                ${formatDate(this.props.date)}
            </div>
        );
    }
}

// Disciplines.tsx
import * as React from "react";
import { observer } from "mobx-react";

import { TwoSidePanel } from "components/TwoSidePanel";
import { DisciplinesView } from "components/disciplines/DisciplinesView";

import uiStore from "stores/UiStore";

import "styles/disciplines/disciplines.scss";

@observer
export class Disciplines extends React.Component<{}, {}>{
    render() {
        return (
            <div id="disciplines">
                <TwoSidePanel left={<DisciplinesView />} right={uiStore.secondaryPage} />
            </div>
        );
    }
}

// DisciplinesView.tsx
import * as React from "react";
import { observer } from "mobx-react";
import Scrollbars from "react-custom-scrollbars";

import { Subject } from "models/subjects";

import { TaskLine } from "components/elements/TasksLine";

import subjectsStore from "stores/SubjectsStore";
import uiStore from "stores/UiStore";

import "styles/disciplines/disciplines-view.scss";

@observer
export class DisciplinesView extends React.Component<{}, {}> {
    render() {
        return (
            <div id="disciplines-view">
                <Scrollbars autoHide>

```

```

        {this.renderDisciplines()}
    </Scrollbars>
</div>
);
}

renderDisciplines() {
    return subjectsStore.subjects.map(this.renderDiscipline);
}

renderDiscipline = (subject: Subject, index: number) => {
    return (
        <div className="discipline-line" key={index} onClick={(e) => this.
            onClickDiscipline(subject)}>
            <div className="discipline-info">
                <span className="discipline-abbreviation">{subject.abbreviation}</
                    span>
                <span className="discipline-tutor">{subject.tutor.getReducedName()
                    }</span>
                <span className="discipline-type">{subject.type}</span>
                <span className="discipline-hours">{subject.totalHours} ..</span>
            </div>
            <div className="discipline-status">
                {subject.tasks ? <TaskLine tasks={subject.tasks} /> : null}
            </div>
        </div>
    );
}

onClickDiscipline(subject: Subject) {
    uiStore.selectSubject(subject);
}
}

// SubjectPage.tsx
import * as React from "react";
import { observer } from "mobx-react";
import { Accordion } from "semantic-ui-react";

import { SubjectName } from "components/elements/SubjectName";
import { TutorName } from "components/elements/TutorName";

import { AgendaItem } from "models/agenda";
import { Subject } from "models/subjects";

import "styles/pages/subject-page.scss";

@observer
export class SubjectPage extends React.Component<{ subject: Subject }, {}> {
    render() {
        return (
            <div className="agenda-item-page">
                {this.renderHeader()}
                {this.renderContent()}
            </div>
        );
    }
}

renderHeader() {

```

```

        return (
            <div className="agenda-item-page-header">
                <span className="agenda-item-page-header-subject">
                    <SubjectName subject={this.props.subject} />
                </span>
                <span className="agenda-item-page-header-lecturer">
                    <TutorName tutor={this.props.subject.tutor} />
                </span>
            </div>
        );
    }

    renderContent() {
        return (
            <Accordion className="agenda-item-page-books" styled fluid exclusive={false}>
                <Accordion.Title></Accordion.Title>
                <Accordion.Content>

                </Accordion.Content>

                <Accordion.Title></Accordion.Title>
                <Accordion.Content>

                </Accordion.Content>
            </Accordion>
        );
    }
}

// TaskPage.tsx
import * as React from "react";
import { observer } from "mobx-react";
import { Accordion } from "semantic-ui-react";

import { SubjectName } from "components/elements/SubjectName";
import { TutorName } from "components/elements/TutorName";

import { Task } from "models/tasks";
import { Subject } from "models/subjects";

import { subjectsStore } from "stores/SubjectsStore";

import "styles/pages/task-page.scss";

@observer
export class TaskPage extends React.Component<{ task: Task }, {}> {
    render() {
        return (
            <div className="task-page">
                {this.renderHeader()}
                {this.renderContent()}
            </div>
        );
    }
}

```

```

}

renderHeader() {
    return (
        <div className="task-page-header">
            <span>
                <span className="task-page-subject">
                    <SubjectName subject={this.props.task.subject} />
                </span>
                <span className="task-page-lecturer">
                    <TutorName tutor={this.props.task.subject.tutor} />
                </span>
            </span>
            <span>
                <span className="task-page-task-type">
                    {subjectsStore.taskTypes.get(this.props.task.type) + " " + this
                     .props.task.number}
                </span>
            </span>
        </div>
    );
}

renderContent() {
    return (
        <Accordion styled fluid defaultActiveIndex={[0]} exclusive={false}>
            <Accordion.Title></Accordion.Title>
            <Accordion.Content>
                {this.props.task.description}
            </Accordion.Content>

            <Accordion.Title></Accordion.Title>
            <Accordion.Content>

            </Accordion.Content>
        </Accordion>
    );
}
}

// SubjectName.tsx
import * as React from "react";
import { observer } from "mobx-react";
import { Popup } from "semantic-ui-react";

import { Subject } from "models/subjects";

export class SubjectName extends React.Component<{ subject: Subject }, {}> {
    render() {
        const subjectAbbreviation = (
            <span className="subject-abbreviation">
                {this.props.subject.abbreviation}
            </span>
        );

```

```

        return (
            <Popup trigger={subjectAbbreviation} content={this.props.subject.name}
                   position="right center" flowing/>
        );
    }
}

// TutorName.tsx
import * as React from "react";
import { observer } from "mobx-react";
import { Popup } from "semantic-ui-react";

import { Tutor } from "models/roles";

export class TutorName extends React.Component<{ tutor: Tutor }, {}> {
    render() {
        const tutorReducedName = (
            <span className="tutor-name">
                {this.props.tutor.getReducedName()}
            </span>
        );
        return (
            <Popup trigger={tutorReducedName} content={this.props.tutor.getFullName
                ()} position="right center" flowing/>
        );
    }
}

import * as React from "react";
import { observer } from "mobx-react";

import { Task } from "models/tasks";

import uiStore from "stores/UiStore";
import subjectStore from "stores/SubjectsStore";

import "styles/elements/tasks-line.scss";

export class TaskLine extends React.Component<{ tasks: Task[] }, {}> {
    render() {
        return (
            <div className="task-line">
                {
                    Array.from(subjectStore.taskTypes).map(([taskType], index) => {
                        let filteredTasks = this.props.tasks.filter((task) => task.type
                            === taskType);
                        if (filteredTasks.length > 0) {
                            return (
                                <span className="tasks-by-type" key={index}>
                                    <span className="tasks">
                                        {
                                            filteredTasks
                                                .sort((a, b) => Number(a.number) - Number(b.
                                                    number))
                                                .map((task, index) => (
                                                    <span className="task" key={index}>
                                                        <span className="task-number">{task.
                                                            type}{task.number}</span>
                                                    {this.renderAgendaLineTaskMark(task)}
                                                
                                            )
                                        }
                                    </span>
                                </span>
                            );
                        }
                    })
                }
            </div>
        );
    }
}

```

```

                </span>
            ))
        }
        </span>
    );
}
);
}

renderAgendaLineTaskMark(task: Task) {
    return (
        <span className="task-mark" onClick={(e) => { this.onClickTask(task); e.
            stopPropagation(); }}>
            {task.mark ? <span className="task-mark-set">{task.mark}</span> :
            null}
        </span>
    );
}

onClickTask(task: Task) {
    uiStore.selectTask(task);
}

// AgendaStore.ts
import { observable, action } from 'mobx';

import { AgendaItem } from "models/agenda";

import * as AgendaMocks from "mocks/agendaMocks";

class AgendaStore {
    @observable agendaItems: AgendaItem[];

    constructor() {
        this.loadAgenda();
    }

    @action loadAgenda() {
        this.agendaItems = AgendaMocks.agendaItems;
    }
}

export const agendaStore = new AgendaStore();
export default agendaStore;

// SubjectStore.ts
import { observable, action } from 'mobx';

import { Subject } from "models/subjects";

import * as TaskMocks from "mocks/tasksMocks";
import * as SubjectMocks from "mocks/subjectsMocks";

class SubjectsStore {

```

```

    @observable activityTypes: Map<string, string>;
    @observable taskTypes: Map<string, string>;

    @observable subjects: Subject[];

    constructor() {
        this.loadActivityTypes();
        this.loadTaskTypes();
        this.loadSubjects();
    }

    @action loadActivityTypes() {
        this.activityTypes = SubjectMocks.activityTypes;
    }

    @action loadTaskTypes() {
        this.taskTypes = TaskMocks.taskTypes;
    }

    @action loadSubjects() {
        this.subjects = SubjectMocks.subjects;
    }
}

export const subjectsStore = new SubjectsStore();
export default subjectsStore;

// UiStore.tsx
import * as React from "react";
import { observable, action } from "mobx";

import agendaStore from "stores/AgendaStore";

import { Task } from "models/tasks";
import { AgendaItem } from "models/agenda";
import { Subject } from "models/subjects";

import { TaskPage } from "components/pages/TaskPage";
import { SubjectPage } from "components/pages/SubjectPage";

class UIStore {
    // Right panel in TwoSidePanel.
    @observable secondaryPage: React.ReactNode;

    @observable selectedTask: Task;
    @observable selectedAgendaItem: AgendaItem;
    @observable selectedSubject: Subject;

    @action selectTask(task: Task) {
        this.selectedTask = task;
        this.secondaryPage = <TaskPage task={this.selectedTask}>;
    }

    //@action selectAgendaItem(agendaItem: AgendaItem) {
    //    this.selectedAgendaItem = agendaItem;
    //    this.secondaryPage = <AgendaItemPage agendaItem={this.selectedAgendaItem}>;
    //}
}

```

```

    @action selectSubject(subject: Subject) {
      this.selectedSubject = subject;
      this.secondaryPage = <SubjectPage subject={this.selectedSubject} />;
    }

}

export const uiStore = new UIStore();
export default uiStore;

// index.scss
*, *:before, *:after {
  box-sizing: border-box;
}

body {
  #root {
    margin: 0;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: medium;
  }
}

.panel {
  margin-bottom: 10px !important;

  .panel-heading a {
    cursor: pointer;
  }
}

// _colors.scss
$light-background-color: #C5D3E2;
$dark-background-color: #1C2A39;
$light-background-hover-color: rgba(172, 186, 201, 0.15);
$dark-background-hover-color: #6C7A89;

$light-background-active-color: #9CAAB9;
$dark-background-active-color: #4C5A69;

$lighter-border-color: rgba(188, 202, 217, 0.25);
$light-border-color: rgba(188, 202, 217, 0.75);
$dark-border-color: #2C3A49;

$mark-set-background-color: rgba(78, 236, 145, 0.25);
$mark-set-hover-background-color: rgba(78, 236, 145, 0.75);
$mark-set-active-background-color: #1EBC61;

$mark-selected: #FDE3A7;
$mark-selected-hover: #FFCF4B;

$text-color: black;
$text-color-inversed: white;

// app.scss
@import "colors";

#app {

```

```

height: 100vh;
margin: auto;
display: flex;
flex-flow: column;
align-items: center;

#main-header {
    flex: 0 0 50px;
    width: 100%;
    background-color: $dark-background-color;
}

#app-body {
    flex: 1 1 auto;
    display: flex;
    flex-flow: row;
    justify-content: center;
    width: 1200px;

    > .menu {
        height: 100%;
    }
}

#app-content {
    flex: 1 1 auto;
}
}

// panels.scss
.two-side-panel {
    height: 100%;

    display: flex;
    flex-direction: row;

    .left-side {
        flex: 0 0 50%;
        overflow-y: auto;
    }

    .right-side {
        flex: 0 0 50%;
        overflow-y: auto;
    }
}

// agenda.scss

#agenda {
    height: 100%;
}

// agenda-view.scss
@import "../colors";

#agenda-view {
    height: 100%;
    display: flex;

```

```

flex-direction: column;

> .segment {
    margin: 0;
}

#agenda-list {
    flex: 0 0 auto;
    display: flex;
    flex-direction: column;
}
}

// agenda-line.scss
@import "../colors";

.agenda-item {
    flex: 0 1 auto;
    display: flex;
    flex-direction: column;
    border: 1px solid $lighter-border-color;
    border-radius: 1px;

    .agenda-item-info {
        flex: 1 1 auto;
        display: flex;
        padding: 14px 7px 7px 7px;

        span {
            flex: 0 0 auto;
            text-align: center;
            overflow: hidden;
            white-space: nowrap;
            text-overflow: ellipsis;
        }
    }

    .agenda-item-time {
    }

    .agenda-item-subject {
        width: 20%;
    }

    .agenda-item-subject-type {
        width: 7%;
    }

    .agenda-item-auditorium {
        flex-shrink: 1;
        width: 20%;
    }

    .agenda-item-lecturer {
        flex-grow: 1;
        text-align: end;
    }
}

.agenda-item-status {

```

```

        flex: 0 1 auto;
        padding: 0 0 14px 0;
    }

    &:hover {
        background-color: $light-background-hover-color;
        cursor: pointer;
    }
}

.agenda-line-date {
    text-align: center;
    padding: 5px 0;
}

// disciplines.scss
#disciplines {
    height: 100%;
}

// disciplines-view.scss
@import "../colors";

#disciplines-view {
    height: 100%;
    display: flex;
    flex-direction: column;

    .discipline-line {
        flex: 0 0 auto;
        display: flex;
        flex-direction: column;
        border: 1px solid $lighter-border-color;
        border-radius: 1px;
        padding: 14px 7px;

        .discipline-info {
            flex: 1 1 auto;
            display: flex;
            justify-content: space-between;

            > span {
                flex: 0 1 auto;
            }
        }

        .discipline-abbreviation {
            flex-shrink: 1;
            width: 20%;
        }

        .discipline-tutor {
            width: 40%;
            text-align: left;
        }

        .discipline-type {
            width: 7%;
        }
    }
}

```

```

        .discipline-hours {
    }
}

.discipline-status {
    flex: 0 1 auto;
}

&:hover {
    background-color: $light-background-hover-color;
    cursor: pointer;
}
}

// subject-page.scss
.agenda-item-page {
    display: flex;
    flex-direction: column;
    padding: 7px;

    .agenda-item-page-header {
        flex: 0 1 auto;
        display: flex;
        flex-direction: column;

        .agenda-item-page-header-subject {
            flex: 0 1 auto;
            font-size: larger;
            margin: 5px;
        }

        .agenda-item-page-header-lecturer {
            flex: 0 1 auto;
            font-size: larger;
            margin: 5px 5px 12px 5px;
        }
    }
}

// task-page.scss
.task-page {
    display: flex;
    flex-direction: column;
    padding: 7px;

    .task-page-header {
        display: flex;
        flex-direction: column;

        > span {
            flex: 1 1 auto;
            display: flex;
            justify-content: space-between;
            margin: 5px;

            .task-page-subject {
                flex: 0 1 auto;
                font-size: larger;
            }
        }
    }
}

```

```

        }

    .task-page-lecturer {
        flex: 0 1 auto;
        font-size: larger;
    }

    .task-page-task-type {
        flex: 1 1 auto;
        font-size: large;
        margin: 0 0 7px 0;
    }

    > span:last-child {
        margin: 5px;
    }
}

// tasks-line.scss
@import "../colors";

.task-line {
    font-size: 15px;
    display: flex;
    flex-wrap: wrap;

.tasks-by-type {
    display: flex;
    flex-direction: row;
    align-items: baseline;
    margin: 0 5px 0 2px;

.tasks {
    flex: 0 1 auto;
    display: flex;
    flex-wrap: wrap;

.task {
    flex: 0 0 auto;
    display: flex;
    margin: 8px 4px 0 0;

    .task-number {
        flex: 0 0 auto;
        width: 2.5em;
        text-align: right;
        padding: 1px 0;
        margin: 0 2px 0 0;
    }

    .task-mark {
        flex: 0 0 auto;
        width: 1.3em;
        text-align: center;
        border: 1px solid $light-border-color;

        &:hover {

```

```
        background-color: $light-background-hover-color;
        cursor: pointer;
    }

.task-mark-set {
    background-color: $mark-set-background-color;
    padding: 0 0 0 1px;
    display: block;

    &:hover {
        background-color: $mark-set-hover-background-color;
    }
}

.mark-selected {
    @extend .task-mark;
    background-color: $mark-selected;

    &:hover {
        background-color: $mark-selected-hover;
    }
}

}

}
}
```

ПРИЛОЖЕНИЕ Б
(рекомендуемое)
Описание схемы базы данных

```
{ N.DatabaseConfiguration
    ConfigurationId:
    RedundancyLevel: 0
    DatabaseSchema:
    { N.DatabaseSchema
        TypeDefinitions:
        *
        { N.TypeDefinition
            TypeName: Entity
            BaseTypeName: DbObject
            FieldDefinitions:
            * Id: Int64
        }
        *
        { N.TypeDefinition
            TypeName: User
            BaseTypeName: DbObject
            FieldDefinitions:
            * Email: String
            * EmailConfirmed: Boolean
            * RegistrationDateTime: DateTime
            * PasswordHash: String
            * PasswordSalt: String
            * HashAlgorithm: String
            * PasswordChangeDateTime: DateTime
            * FirstName: String
            * MiddleName: String
            * LastName: String
            * About: String
        }
        *
        { N.TypeDefinition
            TypeName: StudentRoleRequest
            BaseTypeName: Entity
            FieldDefinitions:
            * State: Byte
            * User: User
            * StudentGroup: StudentGroup
            * Subgroup: Byte
        }
        *
        { N.TypeDefinition
            TypeName: TutorRoleRequest
            BaseTypeName: Entity
            FieldDefinitions:
            * State: Byte
            * User: User
            * Department: Department
        }
        *
        { N.TypeDefinition
            TypeName: ApprovedStudentRoleRequest
            BaseTypeName: Entity
            FieldDefinitions:
```

```

        * ApprovedDateTime: DateTime
        * StudentRoleRequest: StudentRoleRequest
        * Approver: User
    }
    *
{ N.TypeDefinition
    TypeName: ApprovedTutorRoleRequest
    BaseTypeName: Entity
    FieldDefinitions:
        * ApprovedDateTime: DateTime
        * TutorRoleRequest: TutorRoleRequest
        * Approver: User
}
*
{ N.TypeDefinition
    TypeName: Student
    BaseTypeName: Entity
    FieldDefinitions:
        * User: User
        * StudentGroups: StudentGroup[]
}
*
{ N.TypeDefinition
    TypeName: Tutor
    BaseTypeName: DbObject
    FieldDefinitions:
        * User: User
        * AcademicPosition: Byte
        * AdministrativePosition: Byte
        * AcademicDegree: Byte
        * AcademicTitle: Byte
        * Department: Department @BackReference(Tutors)
        * DisciplineSections: DisciplineSection[]
        * DisciplineMaterialSets: DisciplineMaterialSet[]
}
*
{ N.TypeDefinition
    TypeName: DisciplineMaterialSet
    BaseTypeName: Entity
    FieldDefinitions:
        * DisciplineSection: DisciplineSection
        * DisciplineMaterials: DisciplineMaterial[]
}
*
{ N.TypeDefinition
    TypeName: University
    BaseTypeName: DbObject
    FieldDefinitions:
        * Abbreviation: String
        * FullName: String
        * Description: String
        * Faculties: Faculty[] @BackReference(University)
        * Departments: Department[] @BackReference(University)
        * AdministrativeDepartments: AdministrativeDepartment[] @BackReference(
            University)
        * TimeSlots: TimeSlot[]
}
*
{ N.TypeDefinition

```

```

TypeName: Faculty
BaseTypeName: DbObject
FieldDefinitions:
* Abbreviation: String
* FullName: String
* Description: String
* University: University @BackReference(Faculties)
* Specialities: Speciality[] @BackReference(Faculty)
}
*
{ N.TypeDefinition
TypeName: AdministrativeDepartment
BaseTypeName: Entity
FieldDefinitions:
* Name: String
* Description: String
* University: University @BackReference(AdministrativeDepartments)
}
*
{ N.TypeDefinition
TypeName: Department
BaseTypeName: Entity
FieldDefinitions:
* Abbreviation: String
* FullName: String
* Description: String
* University: University @BackReference(Departments)
* Specialities: Speciality[]
* Tutors: Tutor[] @BackReference(Department)
}
*
{ N.TypeDefinition
TypeName: StudentGroup
BaseTypeName: Entity
FieldDefinitions:
* Number: String
* YearOfEntrance: UInt16
* YearOfStudy: Byte
* Speciality: Speciality
}
*
{ N.TypeDefinition
TypeName: Speciality
BaseTypeName: DbObject
FieldDefinitions:
* Abbreviation: String
* FullName: String
* WayOfStudy: Byte
* Faculty: Faculty
* SpecialityPrograms: SpecialityProgram[] @BackReference(Speciality)
}
*
{ N.TypeDefinition
TypeName: SpecialityProgram
BaseTypeName: DbObject
FieldDefinitions:
* YearOfCommissioning: UInt16
* YearsToStudy: UInt16
* Disciplines: Discipline[]

```

```

    * Speciality: Speciality @BackReference(SpecialityPrograms)
}
*
{
    N.TypeDefinition
    TypeName: Discipline
    BaseTypeName: DbObject
    FieldDefinitions:
        * Abbreviation: String
        * FullName: String
        * Description: String
        * TotalHours: UInt16
        * SpecialityProgram: SpecialityProgram
        * DisciplineSections: DisciplineSection[] @BackReference(Discipline)
}
*
{
    N.TypeDefinition
    TypeName: DisciplineSection
    BaseTypeName: DbObject
    FieldDefinitions:
        * Description: String
        * SemesterNumber: Byte
        * SemesterHours: UInt16
        * TypeOfExam: Byte
        * DisciplineProgram: FileObject
        * Discipline: Discipline @BackReference(DisciplineSections)
        * Materials: DisciplineMaterial[]
}
*
{
    N.TypeDefinition
    TypeName: DisciplineMaterial
    BaseTypeName: Entity
    FieldDefinitions:
        * Title: String
        * TypeOfMaterial: Byte
        * LastUpdateTime: DateTime
        * FileObject: FileObject
        * DisciplineSection: DisciplineSection
        * Tutor: Tutor
}
*
{
    N.TypeDefinition
    TypeName: StudentTask
    BaseTypeName: Entity
    FieldDefinitions:
        * DisciplineMaterial: DisciplineMaterial
        * TaskNumber: Byte
        * VariantNumber: Byte
        * Student: Student
        * Tutor: Tutor
        * TaskResults: TaskResult[] @BackReference(StudentTask)
        * WorkflowState: Byte
        * Mark: Byte
        * MarkDate: DateTime
}
*
{
    N.TypeDefinition
    TypeName: TaskResult
    BaseTypeName: Entity
    FieldDefinitions:

```

```

    * StudentTask: StudentTask @BackReference(TaskResults)
    * DeliveryDate: DateTime
    * FileObject: FileObject
}
*
{ N.TypeDefinition
    TypeName: TimeSlot
    BaseTypeName: DbObject
    FieldDefinitions:
    * LessonNumber: Byte
    * StartTime: DateTime
    * EndTime: DateTime
}
*
{ N.TypeDefinition
    TypeName: ClassesSchedule
    BaseTypeName: Entity
    FieldDefinitions:
    * WeekNumber: Byte
    * WeekDay: Byte
    * StudentGroups: StudentGroup[]
    * TimeSlot: TimeSlot
    * ClassRoom: ClassRoom
    * DisciplineSection: DisciplineSection
    * AcademicYear: UInt16
    * Season: Byte
    * Tutor: Tutor
    * TypeOfClasses: Byte
}
*
{ N.TypeDefinition
    TypeName: ClassesEvent
    BaseTypeName: Entity
    FieldDefinitions:
    * ClassesSchedule: ClassesSchedule
    * Date: DateTime
}
*
{ N.TypeDefinition
    TypeName: StudentAttendance
    BaseTypeName: Entity
    FieldDefinitions:
    * ClassesEvent: ClassesEvent
    * Student: Student
    * Attendance: Byte
}
*
{ N.TypeDefinition
    TypeName: ClassRoom
    BaseTypeName: Entity
    FieldDefinitions:
    * University: University
    * BuildingNumber: Byte
    * Address: String
    * RoomNumber: UInt16
}
*
{ N.TypeDefinition
    TypeName: Message

```

```

BaseTypeName: Entity
FieldDefinitions:
* FromUser: User
* ToUser: User
* Text: String
* Attachment: FileObject
* Time: DateTime
}
SecondaryIndexDefinitions:
* User[+Email]!
* StudentRoleRequest[+Id]!
* TutorRoleRequest[+Id]!
* ApprovedStudentRoleRequest[+Id]!
* ApprovedTutorRoleRequest[+Id]!
* Student[+Id]!
* Faculty[+Abbreviation]!
* Speciality[+Abbreviation]!
* University[+Abbreviation]!
* AdministrativeDepartment[+Id]!
* Department[+Id]!
* StudentGroup[+Id]!
* SpecialityProgram[+YearOfCommissioning+YearsToStudy]
* Discipline[+Abbreviation]!
* DisciplineSection[+SemesterNumber]!
* DisciplineMaterial[+Id]!
* StudentTask[+Id]!
* TaskResult[+Id]!
* TimeSlot[+LessonNumber]!
* ClassesSchedule[+Id]!
* ClassesEvent[+Id]!
* StudentAttendance[+Id]!
* ClassRoom[+Id]!
* Message[+Id]!
}
Nodes:
* 0
* 1
PrimaryIndexDistributions:
*
{ N.PrimaryIndexDistribution
  Nodes:
  * 0
  * 1
}
SecondaryIndexDistributions:
*
{ N.SecondaryIndexDistribution
  IndexDefinition: User[+Email]!
  Nodes:
  * 0
  * 1
  Ranges:
  *
  { N.SecondaryIndexDistributionRange
    InclusiveUpperBound:
    * p
    MirrorNodes:
    * 0
  }
}

```

```

*
{ N.SecondaryIndexDistributionRange
  MirrorNodes:
    * 1
}
}

*
{ N.SecondaryIndexDistribution
  IndexDefinition: University[+Abbreviation]!
  Nodes:
    * 0
    * 1
  Ranges:
  *
  { N.SecondaryIndexDistributionRange
    InclusiveUpperBound:
    *
    MirrorNodes:
      * 0
  }
  *
  { N.SecondaryIndexDistribution
    MirrorNodes:
      * 1
  }
}
*

{
  N.SecondaryIndexDistribution
  IndexDefinition: Faculty[+Abbreviation]!
  Nodes:
    * 0
    * 1
  Ranges:
  *
  { N.SecondaryIndexDistributionRange
    InclusiveUpperBound:
    *
    MirrorNodes:
      * 0
  }
  *
  { N.SecondaryIndexDistribution
    MirrorNodes:
      * 1
  }
}
*

{
  N.SecondaryIndexDistribution
  IndexDefinition: Speciality[+Abbreviation]!
  Nodes:
    * 0
    * 1
  Ranges:
  *
  { N.SecondaryIndexDistributionRange
    InclusiveUpperBound:
    *
    MirrorNodes:
      * 0
}

```

```

}
*
{ N.SecondaryIndexDistributionRange
  MirrorNodes:
    * 1
  }
}
*
{ N.SecondaryIndexDistribution
  IndexDefinition: Discipline[+Abbreviation]!
  Nodes:
    * 0
    * 1
  Ranges:
  *
  { N.SecondaryIndexDistributionRange
    InclusiveUpperBound:
    *
    MirrorNodes:
      * 0
    }
  *
  { N.SecondaryIndexDistributionRange
    MirrorNodes:
      * 1
    }
  }
}
FileBufferSize: 65536
FileStorageDistributions:
* { N.FileStorageDistribution
  FileBlockSize: 4096
  Nodes:
    * 0
    * 1
  }
}

```

ПРИЛОЖЕНИЕ В

(рекомендуемое)

Конфигурационные файлы проекта

```
// webpack.config.js
module.exports = {
  entry: [
    "./src/index.tsx"
  ],
  output: {
    filename: "./dist/bundle.js"
  },
  devtool: "source-map",
  resolve: {
    extensions: [".webpack.js", ".web.js", ".ts", ".tsx", ".js", ".scss"],
    modules: ["src", "node_modules"]
  },
  devServer: {
    contentBase: ".",
    host: "localhost",
    port: 8080
  },
  module: {
    rules: [
      {
        enforce: "pre",
        test: /\.js$/,
        loader: "source-map-loader"
      },
      {
        test: /\.tsx?$/,
        loader: "ts-loader"
      },
      {
        test: /\.scss$/,
        loader: "style-loader!css-loader!sass-loader"
      },
      {
        test: /\.(eot|ttf|woff2?|otf|svg|png|jpg)$/,
        loader: "file-loader"
      }
    ]
  },
  externals: {
    "react": "React",
    "react-dom": "ReactDOM"
  }
};
```

```

// package.json
{
  "version": "1.0.0",
  "name": "univer.frontend",
  "private": true,
  "scripts": {
    "webpack": "webpack -w --display-error-details --colors",
    "start": "webpack-dev-server --hot --inline --colors"
  },
  "dependencies": {
    "dateformat": "^2.0.0",
    "mobx": "^3.1.0",
    "mobx-react": "^4.1.0",
    "react": "^15.4.2",
    "react-custom-scrollbars": "^4.0.2",
    "react-dom": "^15.4.2",
    "react-router": "^3.0.2",
    "semantic-ui-css": "2.2.9",
    "semantic-ui-react": "0.67.0"
  },
  "devDependencies": {
    "@types/dateformat": "^1.0.1",
    "@types/react": "^15.0.7",
    "@types/react-custom-scrollbars": "^2.0.0",
    "@types/react-dom": "^0.14.20",
    "@types/react-router": "^2.0.46",
    "css-loader": "^0.26.1",
    "file-loader": "^0.10.0",
    "node-sass": "^4.5.0",
    "sass-loader": "^5.0.0",
    "source-map-loader": "^0.1.6",
    "style-loader": "^0.13.1",
    "ts-loader": "^2.0.1",
    "typescript": "^2.2.1",
    "webpack": "^2.2.1",
    "webpack-dev-server": "^2.3.0"
  }
}

```