СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Практика на предприятии позволяет молодому специалисту применить полученные теоретические знания и ознакомиться со специальным оборудованием и программным обеспечением, применяемым при разработке, сопровождении и эксплуатации промышленных информационных систем. Преимуществом такого вида деятельности заключается в том, что студент находится под контролем руководителя, являющегося специалистом с обширным практическим опытом, что позволяет быстрее и лучше усваивать получаемые знания.

Основной задачей практики является получение навыков проектирования и разработки информационных систем в рамках коммерческого предприятия. Данными информационными системами могут быть любые программы, направленные на оптимизацию или автоматизацию деятельности предприятия.

Автоматизация – одно из направлений научно-технического прогресса, использующее технические средства и математические методы с целью освобождения человека от участия в процессах получения, преобразования, передачи и использования энергии, материалов, изделий или информации, либо существенного уменьшения степени этого участия или трудоёмкости выполняемых операций. Она позволяет более эффективно использовать ресурсы предприятия и конкурировать с другими предприятиями этой же сферы.

Оптимизация процессов предприятия также является одним из самых важных направлений деятельности организации. При помощи оптимизации организация может тратить меньше ресурсов для достижения большего результата. Базовым примером оптимизации на предприятии может служить улучшение логистики для уменьшение затрат на транспортировку сырья или перерасчёт использования сырья на различные товары, максимизируя прибыль. Кроме того, оптимизация может иметь не совсем материальный характер, например предприятие может затратить меньше денег для обучения сотрудников, улучшив качество обучения при помощи специальных программных комплексов.

Целью практики является разработка приложения для персонального планирования и контроля процессов обучения сотрудников предприятия. Данное приложение направлено на оптимизацию процессов обучения сотрудников, позволяя уделять меньше времени на планирование и управление обучением, в связи с чем увеличивается время на самообучение.

Работа является актуальной в тех областях, где от сотрудников или от студентов требуется постоянное повышение своей квалификации. Данное приложение позволяет контролировать большое количество проектов обучения, что очень полезно для студентов или для специалистов, работающих в области информационных технологий, так как обе эти деятельности неразрывно связаны с постоянным изучением большого количества нового материала.

Основные этапы прохождения практики:

– ознакомление с базой прохождения практики;

– ознакомление со структурой организации, перечнем решаемых задач и внутренним распорядком;

– сбор информации для выполнения индивидуального задания «Приложение для персонального планирования и контроля процессов обучения»;

– анализ методов решения задачи планирования проектов и контроля процессов обучения;

– исследование существующих программных решений для персонального планирования;

– исследование существующих программных решений для контроля процессов обучения;

– определение архитектуры программного комплекса;

– разработка структуры баз данных приложения;

– разработка системы аутентификации пользователей приложения;

– разработка серверной части приложения;

– разработка клиентской части приложения;

– модульное тестирование приложения;

– верификация работы информационной системы;

– анализ разработанного программного обеспечения и написание отчёта по преддипломной практике.

Главным этапом прохождения практики является ознакомление с предприятием и понимание внутренних производственных процессов этого предприятия.

1 ХАРАКТЕРИСТИКА ПРЕДПРИЯТИЯ ООО «Ива-Гомель-Парк»

1.1 Общие сведения о предприятии

IBA была основана в 1993 году в качестве трехстороннего партнерства между *IBM Corp*. и двумя ведущими белорусскими ИТ-предприятиями. В 1999 году *IBM* вышла из состава *IBA* как совладелец, однако она остается стратегическим деловым партнером. Компания *IBA* была призвана дополнить высокую квалификацию местных специалистов в области разработки программного обеспечения передовыми технологиями и решениями, предоставляемыми *IBM*.

В *IBA* работают сотрудники из Беларуси, Чехии, Польши, Словакии, Болгарии, Казахстана, которые имеют разносторонний опыт. Часто опытные программисты работают вместе с молодыми разработчиками над одним проектом.

34 процента сотрудников пришли в компанию за последние три года. Многие из них – выпускники университетов, которые окончили курсы *IBA Group*.

29 процента сотрудников работают в *IBA* от трёх до десяти лет.

32 процента сотрудников проработали в компании более десяти лет. Большинство из них начали свою карьеру в *IBA Group*.

Пять процентов сотрудников имеют опыт работы в области информационных технологий более 20 лет.

Компания сочетает различные организационные модели, что бы лучше удовлетворять потребности клиентов.

Структура организации представлена на рисунке 1.1.

Рисунок 1.1 – Структура организации *IBA Group*

IBA работает с организациями по всему миру. В область деятельности компании входят страны Европы, Азии, Америки, Африки и Австралии.

Список стран, с которыми сотрудничает компания изображён на рисунке 1.2.

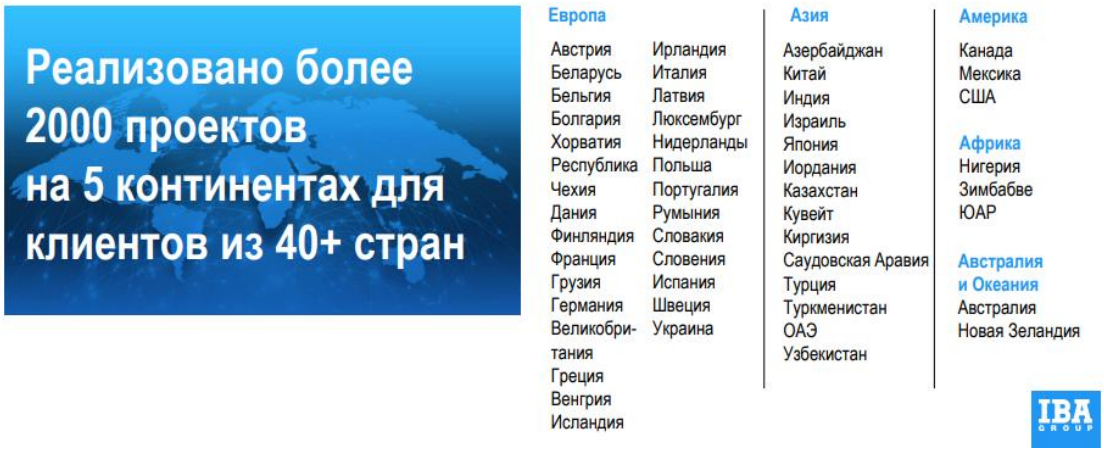

Рисунок 1.2 – Список стран, с которыми работает *IBA Group*

Компания имеет множество постоянных клиентов, таких как:

– *IBM* – один из крупнейших в мире производителей и поставщиков аппаратного и программного обеспечения. Корпорация представлена практически во всех странах мира. На конец 2020 года наибольшее число сотрудников компании работало в США и в Индии;

– *Standart Bank* – южноафриканская финансовая группа, включающая дочерние банки в 20 странах Африки, страховую компанию *Liberty* и компанию по управлению активами *Melville Douglas*;

– *Goodyear* – американская международная компания, производитель шин и других резинотехнических изделий, а также полимеров для автомобильного и промышленного рынков. Основана в 1898 году в городе Акрон в штате Огайо; главное управление находится в этом городе и в настоящее время. *Goodyear* производит шины для легковых и грузовых автомобилей, мотоциклов, гоночных машин, самолётов, сельскохозяйственной и землеройно-транспортной техники;

– *Rockwell Automation* – американский поставщик промышленной автоматизации и информационных продуктов. Бренды включают *Allen-Bradley* и программное обеспечение *Rockwell*;

– *Fujitsu* – крупная японская корпорация, производитель электроники и ИТ-компания. Работает на глобальном уровне, имеет дочерние подразделения во всём мире;

– *Hapag-Lloyd* – транснациональная немецкая транспортная компания. Она состоит из линии по доставке грузовых контейнеров *Hapag-Lloyd AG*;

– *T Mobile* – группа компаний, работающих в области мобильной связи, которые находятся в собственности немецкого телекоммуникационного холдинга *Deutsche Telekom*. Эти компании управляют *GSM*-сетями в Европе;

– *O2* – британский поставщик телекоммуникационных услуг со штаб-квартирой в Слау, Англия.

Список постоянных клиентов и даты начала работы с ними изображены на рисунке 1.3.

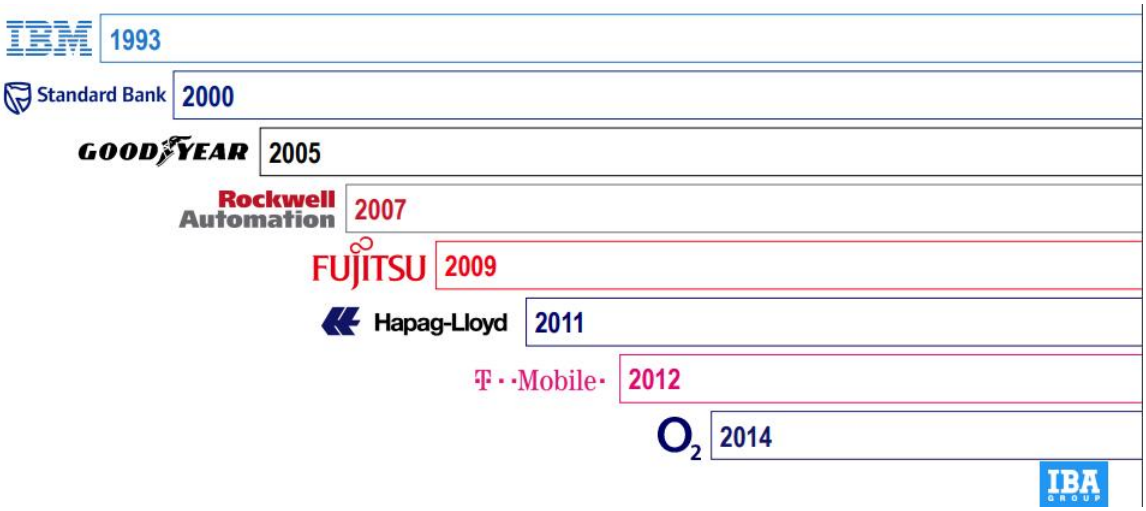

Рисунок 1.3 – Список постоянных клиентов компании *IBA Group*

Основными направлениями деятельности компании являются:
– Мейнфреймы *IBM*;
– Машинное обучение;
– Бизнес-аналитика;
– *SAP*-системы;
– Облачные решения;
– Интернет вещей.

Кроме этих направлений, компания имеет очень широкий спектр других направлений, в которых работают тысячи программистов по всему миру [1].

1.2 История предприятия

В 1993 году была создана компания *IBA* в городе Минск, Беларусь.

В 1994 году в Минске был открыт вычислительный центр *IBA*.

В 1997 году было создано СП «Информационные производственные архитектуры» – *IBA IS*.

В 1998 году компания открыла офис в Америке, а в 1999 были открыты офисы в Чехии, на Кипре и был открыт технический центр в Минске.

В 2000 году были открыты офис в Германии и студия веб-дизайна в Минске.

В 2001 году компания открыла центр разработки в городе Гомель Республики Беларусь. Этот филиал получил название «ИВА-Гомель-Парк».

После этого до 2007 года были открыты многочисленные отделения в Беларуси, Чехии и России.

В 2007 году компания стала резидентом парка высоких технологий Республики Беларусь.

В течении 8 лет компания открыла множество других офисов. В том числе были открыты офисы в Казахстане и в Украине.

В 2015 году *IBA Group* была включена в категорию «Лидеры» рейтинга «*The* 2015 *Global Outsourcing* 100». *IBA IT Park*, один из белорусских центров

разработок *IBA Group*, стал одним среди крупнейших компаний-резидентов ПВТ по итогам 2014 года. Система автоматизации оплаты в общественном транспорте Минска, созданная *IBA Group*, признана лучшим отраслевым решением года на конкурсе «*European IT & Software Excellence Awards*».

В 2016 году *IBA Group* снова попала в категорию «Лидеры» рейтинга «*The* 2016 *Global Outsourcing* 100», а также была отмечена как лучшая компания за достижения в области социальной ответственности, высокую удовлетворенность клиентов, инновации, сертификацию и победы в конкурсах и рейтингах. Также *IBA Group* была включена в список финалистов европейского конкурса ИТ-проектов «*European IT & Software Excellence Awards* 2016». *IBA Minsk*, центр разработок *IBA Group* стал пятикратным лауреатом Премии Правительства Республики Беларусь за достижения в области качества.

Далее, до 2022 года, компания также получила многочисленные награды в самых разнообразных ИТ отраслях и прочно закрепилась на глобальном рынке информационных технологий. Кроме того, *IBA* разработала большое количество программного обеспечения, призванного облегчить жизнь людей и больших корпораций [2].

1.3 Корпоративная политика

В октябре 2008 года *IBA Group* приняла единую программу корпоративной социальной ответственности (КСО). В программе КСО отмечается, что в своей повседневной деятельности компания руководствуется принципами этичного поведения, прозрачности, уважения к законам и международным нормам, а также правам человека.

В 2009 году компания *IBA* была награждена золотой медалью и дипломом победителя профессионального конкурса «Бренд года» в почетной номинации «Социально ответственный брэнд» в категории «Лучший работодатель».

В 2011 году *IBA Group* стала финалистом конкурса «2011 *European Outsourcing Association Awards*» в категории «*Award for Corporate Social Responsibility*» – награда за корпоративную социальную ответственность.

В 2012 году *IBA* была признана одной из лучших компаний для работы в Беларуси. *IBA* была выбрана победителем в категории «Лучшая ИТ-компания для работы (50+ сотрудников)» и была удостоена третьего приза в категории «Лучшая компания для работы (250+ сотрудников)».

Социальный пакет компании включает:

– бесплатное медицинское обслуживание в корпоративном медпункте и периодическое медицинское обследование;

– компенсацию части затрат на занятия спортом;

– компенсацию стоимости медицинского обслуживания в учреждениях здравоохранения и части затрат на санаторно-курортное лечение и оздоровление;

– частичную компенсацию затрат на добровольное медицинское страхование;

– бесплатную вакцинацию против гриппа;

– скидки на оздоровление детей сотрудников в летних лагерях;

– спорт (спортивные секции, внешние и внутренние чемпионаты);

– туризм (пешие экскурсии, походы, туристические фестивали, поездки выходного дня);

– корпоративные праздники;

– беспроцентные займы;

– корпоративный транспорт;

– материальную помощь (вступление в брак, рождение ребенка, декрет, смерть близкого родственника).

Кроме социального пакета компания имеет широкие возможности для повышения квалификации и профессиональной подготовки персонала.

Сотрудники *IBA* могут проходить обучение в Институте *IBA* и в центрах обучения мировых лидеров ИТ-индустрии, включая *IBM* (США, Германия, Великобритания, Словения и Россия), *HP*, *SAP*, *SAS*, *Microsoft*, *Novell*, *Check Point*, *PTC* и 1С.

Институт *IBA* предлагает курсы в области информационных технологий, бизнеса и иностранных языков (английский, немецкий, французский и китайский). Институт *IBA* является авторизованным центром тестирования *Thompson Prometric*, что позволяет сотрудникам *IBA* проходить сертификационные тесты по различным программам, включая более 70 тестов в области ИТ (*Microsoft*, *Novell*, *Cisco*, *Oracle*, *SAP*, *Symantec* и *IBM*). Около 25% сотрудников компании прошли обучение в образовательных центрах глобальных ИТ-провайдеров.

IBA Group заинтересована в развитии, поощрении и продвижении талантливых сотрудников. Поощрение включает как материальные стимулы, так и моральные. В дополнение к заработной плате, премиям и возможности обучения в ведущих ИТ-компаниях, *IBA* создала электронную доску почета. Письма с благодарностью заказчиков сотрудникам публикуются на корпоративном портале. Для молодых талантливых сотрудников создаются возможности занять ведущие должности в компании.

Через корпоративную доску объявлений сотрудники *IBA* собирают средства для детей, которым необходимо лечение за границей, подписи в знак поддержки или протеста против каких-либо событий, а также находят новые дома для брошенных животных [3].

1.4 Деятельность предприятия

Основная деятельность предприятия направлена на разработку и поддержку программного обеспечения компаний-клиентов. Однако кроме основной деятельности, компания также вкладывает большие средства в ИТ образование.

IBA Group сотрудничает с университетами Республики Беларусь, лидирующими в области подготовки специалистов по информационным технологиям: Белорусским государственным университетом (БГУ), Белорусским государственным университетом информатики и

радиоэлектроники (БГУИР), Гомельским государственным университетом имени Франциска Скорины (ГГУ) и Гомельским Государственным Техническим Университетом (ГГТУ) с целью подготовки и трудоустройства молодых специалистов.

Сотрудничество с университетами включает:

– работу ведущих специалистов в качестве преподавателей в университетах;

– обучение студентов передовым информационным технологиям, в том числе целевое обучение в соответствии с потребностями компании;

– организацию практики студентов в компаниях альянса;

– организацию совместных мероприятий: конференций, семинаров, совещаний, дней карьерного роста, ярмарок вакансий с участием сотрудников, преподавателей и студентов;

– совместные мероприятия по материально-технической, информационной, методической поддержке учебного процесса: создание совместных лабораторий, учебных классов, обеспечение общих сетевых коммуникаций.

С 2004 *IBA* является генеральным спонсором студенческих команд БГУ и БГУИР, участвующих в Международной студенческой олимпиаде по программированию «*ACM International Collegiate Programming Contest*».

С 2010 года *IBA* является главным спонсором команды БГУИР на Международной студенческой олимпиаде по программированию.

В 2012 году команды БГУ и БГУИР завоевали серебряные и бронзовые медали.

Кроме деятельности в пределах Беларуси, компания ведёт деятельность по обмену знаниями студентов разных стран.

Участвуют БГУ, БГУИР, ГГУ (Гомель), Карлов университет (Прага), Чешский технический университет (Прага), Пражский университет экономики, Масариков университет (Брно), Университет Западной Богемии (Пльзень).

С 1999 года действует первая совместная лаборатория *IBA* и БГУ «Новые информационные технологии». С 2011 года *IBA* стал центром ИТ-компетенций БГУ.

В 2008 году основано 2 совместные ИТ-лаборатории с БГУИР.

В 2011 году *IBA* и БГУ открыли Центр ИТ-компетенций, направленный на повышение уровня ИТ-образования в Белорусском государственном университете. Предоставляя преподавателям и студентам БГУ образование по современным ИТ-технологиям, Центр ИТ-компетентности способствует реализации следующих программ:

– университетские программы *IBM*, включая академическую инициативу *IBM*, запущенную в БГУ;

– образовательная программа *Microsoft*, включающая многочисленные курсы по инновациям в образовательном процессе (*IT*-Академия, инновационные центры, *MLG*, *MDAA*, *E-Learning*), лицензионные программы для преподавателей и студентов (*CASA*, *MSDN Academic Alliance*, *Faculty Connection*), а также поддержку студентов и молодых специалистов на

конкурсах и конференциях (*Imagine Cup*, *Microsoft technologies in programming theory and practice*);

– образовательные программы с использованием ландшафта *SAP*, развернутые на серверах *IBA;*

– образовательные программы для преподавателей и студентов в области информационных технологий для удовлетворения потребностей *IBA* в людских ресурсах.

IBA Group также ведёт другие виды деятельности.

Благотворительная помощь, осуществляемая на регулярной основе, выражается в предоставлении оборудования и услуг или оказании поддержки иного рода следующим организациям: центр внешкольной работы и социально-педагогический центр Советского района, детский приют, общество охотников и рыболовов, ОСВОД.

Уже более 25 лет белорусские центры разработок *IBA Group* стремятся вносить свой посильный вклад в обустройство жизни республики: принимают активное участие в общественных и благотворительных акциях, поддерживают талантливую молодежь и плодотворно сотрудничают с учреждениями образования по всей стране.

IBA регулярно принимает участие в благотворительных акциях и программах Белорусского благотворительного фонда для чернобыльских детей, Белорусского детского хосписа, поздравляет ветеранов Второй Мировой Войны, дарит новогодние подарки детям минского детского дома № 3, минского детского дома № 4, детского дома № 3 Мядельского района, социально-педагогических центров Минского Советского района и Логойской области, школы для умственно отсталых детей Пинского района.

IBA Group стремимся эффективно использовать электроэнергию и другие ресурсы. *Green IT* имеет большое значение для *IBA*.

Ради сохранения природных ресурсов и сбережения энергии *IBA* старается приобретать только то оборудование, которое действительно нужно, экономно использует бумагу, использует инструменты управления питанием для мониторов и жестких дисков, заменяет устаревшее оборудование новым и более эффективным, внедряет технологии нового поколения [4].

Также *IBA* активно занимается разработкой средств автоматизации и оптимизации рабочих процессов для своих сотрудников, так как это увеличивает их эффективность обучения и квалификацию. Именно поэтому в ходе практики было поставлено задание разработки такого программного обеспечения.

2 ОБЗОР ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНОЙ ЗАДАЧИ

2.1 Постановка индивидуальной задачи

Индивидуальным заданием, выданным руководителем практики является разработка приложения для персонального планирования проектов и контроля процессов обучения.

Шаги выполнения поставленной задачи:

– изучение требований, предъявляемых к безопасной и комфортной работе инженеров-программистов. Знакомство со спецификой этапов жизненного цикла программного продукта;

– анализ методов планирования проектов и контроля результатов обучения;

– изучение методов тестирования и классификации тестов;

– сравнительный анализ существующих программных решений для персонального планирования в том числе для обучения;

– сравнительный анализ существующих программных решений для контроля результатов обучения;

– проектирование структуры программного комплекса, базы данных для хранения информации, формирование пользовательских правил (ролевых политик) для доступа к ресурсам и функциям системы, графического интерфейса;

– разработка системы авторизации и аутентификации пользователей различных групп;

– формирование информационной базы для выполнения дипломной работы и тестирования предполагаемых к созданию программных продуктов;

– создание прототипа программного продукта в соответствии с темой дипломной работы.

Приложение должно поддерживаться всеми современными видами устройств: настольными компьютерами, ноутбуками, мобильными телефонами.

Также приложение должно предоставлять функционал аутентификации и авторизации пользователей для того, что бы каждый пользователь мог получать доступ к своим проектам и ограничить доступ другим пользователям.

Акцент приложения следует сделать на визуальной составляющей и удобстве использования, так как это самая главная характеристика такого рода приложений. Для этого приложение должно иметь только самый необходимый функционал, который должен располагаться в различных местах программы.

Организация проектов должна иметь древовидную структуру, так как такого рода организация избегает ограничения линейного представления процессов работы над проектами:

– невозможность динамичной декомпозиции задач проекта;

– невозможность сделать точную иерархию задач;

– невозможность наглядного представления большого объёма данных.

Приложение должно позволять пользователю добавлять и удобно просматривать вложения различного рода, например ссылки или таблицы. Это

позволит включать в удобную древовидную структуру проектов дополнительную информацию, представленную различными типами данных.

Приложение должно иметь понятный и удобный пользовательский интерфейс, который не будет отвлекать пользователя и одинаково удобно отображаться на различного рода устройствах.

Первой поставленной задачей было изучение методов планирования проектов и контроля обучения и сравнительная характеристика готовых инструментов и разработанного приложения.

2.2 Методы планирования проектов

Методы планирования проектов необходимо использовать, чтобы выровнять все аспекты проектов так, чтобы они соответствовали друг другу. График должен быть соразмерен времени, установленному для проекта, и все его ресурсы должны использоваться оптимальным образом. Учитывая изменчивый характер проектов и, иногда, их масштабы, их сложно спланировать, но это необходимо сделать, потому что без какого-либо плана вероятность успешного завершения проекта очень мала.

График состоит из всех действий, включенных в реализацию и выполнение проекта в течение заранее определенного периода времени проекта. График проекта помогает расставить приоритеты в работе над проектом и завершить ее упорядоченным образом. Это также помогает в назначении правильного человека для работы и в надлежащем распределении доступных ресурсов. Управление временем и корректировка в рамках проекта возможны только при наличии надлежащего графика, подготовленного для работы над проектом [5].

Планирование проекта, как правило, включает в себя различные методы, краткое описание каждого метода приводится ниже.

Первыми методами планирования проектов являются методы математического анализа: метод критического пути и метод оценки и анализа программ (*PERT*). Метод критического пути (*CPM*) и метод оценки и анализа программы (*PERT*) являются двумя наиболее часто используемыми методами для руководителей проектов. Эти методы используются для расчета времени выполнения проекта.

Древовидная диаграмма каждого проекта имеет критический путь. Метод критического пути оценивает максимальное и минимальное время, необходимое для завершения проекта. *CPM* также помогает определять критические задачи, которые должны быть включены в проект.

PERT – это способ запланировать поток задач в проекте и оценить общее время, необходимое для его выполнения. Этот метод помогает понять, как каждая задача зависит от другой. Чтобы запланировать проект с использованием *PERT*, необходимо определить виды деятельности, упорядочить их и определить основные этапы.

Следующим методом планирования проектов является моделирование. В данном методе ожидаемая продолжительность проекта рассчитывается с использованием другого набора задач в симуляции. Расписание создается на

основе предположений, поэтому его можно использовать, даже если область действия изменилась или задачи недостаточно ясны.

Далее необходимо выделить список задач. Он является одним из самых простых методов планирования проектов. Для его реализации необходимо в электронной таблице или текстовом редакторе привести список всех возможных задач, связанных с проектом. Этот метод является простым и самым популярным из всех методов. Это очень полезно при реализации небольших проектов. Но для больших проектов с многочисленными аспектами рассмотреть список задач не представляется возможным.

Также одним из самых популярных методов планирования проектов является диаграмма Ганта. Диаграмма Ганта представляет собой метод визуализации, используемый в управлении проектами. Он используется менеджерами проектов большую часть времени, чтобы получить представление о среднем времени, необходимом для завершения проекта. График Ганта – это столбчатая диаграмма, которая представляет ключевые действия в последовательности слева и в зависимости от времени. Каждая задача представлена полосой, которая отражает начало и конец действия в проекте, то есть его продолжительность.

И заключительным популярным методом планирования проектов является календарь. Большинство календарей написаны для персональных нужд работников проекта. Но часто создаются общие календари с требованиями для проекта, которые могут быть скопированы и изменены индивидуально. Календарь показывает временную шкалу для всего проекта. Основным преимуществом является то, что он может подвергаться изменениям, поскольку он является разделяемым. Несмотря на то, что это отличный метод для отслеживания проекта, у него есть определенные ограничения: нельзя назначать задачи определенным людям и затруднительно увидеть зависимости задач.

Инструменты планирования ресурсов могут помочь менеджеру проекта в распределении ресурсов и завершить проект в течение установленного промежутка времени. Они также помогают в подготовке графиков для всех проектов, которые должны быть выполнены в будущем. Изучая результаты своей команды в прошлом, препятствия, с которыми сталкивались при реализации проекта, и как преодолеть эти препятствия, менеджер может делать более точные прогнозы о процессе выполнения проекта. Планирование должно быть неотъемлемой частью управления проектом.

Большинство из этих методов применимы и для планирования персональных проектов пользователей. Проекты могут иметь различный характер, однако есть общие черты, которые позволяют пользоваться одним из методов для управления всеми своими проектами [6].

2.3 Программные решения для персонального планирования

Для всех вышеперечисленных методов планирования проектов существует многочисленное и разнообразное программное обеспечение. Одни программы

являются бесплатными, другие коммерческими, третьи предоставляют пробный период. Все эти программы отличаются и пользователь может выбирать то, что ему нравится больше, исходя из собственных предпочтений.

Первым расмотренным приложением для планирования персональными проектами является *Any.DO*. Программа *Any.DO* – один из самых популярных планировщиков среди пользователей *Android* и *iOS*. Приложение отличается удобным и простым интерфейсом и может синхронизироваться с несколькими устройствами. Для добавления заданий в приложении можно пользоваться голосовым набором – при этом включается интеллектуальный ввод, позволяющий выбрать запись из готовых вариантов.

Преимущества планировщика *Any.DO*:

– ввод текста с возможностью прикрепления к нему видеофайлов, изображений или фото – возможность, отсутствующая у большинства похожих утилит;

– простое добавление пользовательских списков из главного меню;

– удобное переключение между режимами;

– работа с разными видами данных – в том числе списками дел, покупок.

Приложение работает не только на смартфонах, но и в браузере. Интеграция с операционной системой позволяет ему выдавать сообщения прямо в строке состояния. Среди других плюсов стоит отметить многопользовательскую работу, защиту информации с помощью кода и геотеги. Базовая версия приложения бесплатна, но функциональность можно расширить, используя платную версию.

Следующим приложением для планирования является *Todoist*. Программа *Todoist* содержит целый комплекс полезных функций, позволяющих контролировать выполнение любых задач. С её помощью можно вести статистику, составлять различные списки, синхронизировать несколько устройств, на которых установлен планировщик.

Список возможностей приложения включает:

– добавление задач прямо с рабочего стола мобильного устройства – для этого достаточно использовать виджеты;

– настройку напоминаний;

– запись комментариев и родительских задач;

– контроль выполнения задач в статистике профиля.

Главные преимущества приложения - целый ряд доступных пользователю цветовых схем, автоматическое перемещение новых задач в список входящих, удобные фильтры и метки, позволяющие отсортировать задания. Еще один важный плюс – поддержка более чем 10 платформ и приложений, в которые интегрируется *Todoist* – среди них *MacOS*, *Windows*, *iOS*, *Android*, *браузеры Firefox* и *Chrome*.

Далее была рассмотрена программа *GTasks*. Планировщик *GTasks* отличается удобством использования и возможностью синхронизации с несколькими устройствами с помощью облачных сервисов. Для этого пользователю придется авторизоваться в программе через аккаунт *Google*.

Причем, несмотря на то, что приложение выпущено самой *Google*, пользоваться им могут также владельцы *iOS*.

Основные особенности программы:

– синхронизация с *Google* Календарем, из которого можно импортировать данные;

– работа в локальном режиме, если все сведения хранятся на устройстве;

– стильный дизайн и удобный интерфейс;

– голосовой ввод заданий.

Преимущества утилиты состоят в мультиплатформенности и удобном отображении информации. Списки, которые пользователь добавляет в *GTasks*, легко добавляются, скрываются и удаляются. При подключении платной версии становится доступным еще и резервное копирование, смена оформления и блокировка данных ключом.

После этого был рассмотрен визуальный инструмент планирования *Sectograph*. *Sectograph* – это практичный визуальный помощник, выдающий список дел в виде циферблата часов. Схема *Sectograph* позволяет не только узнать, какие дела запланированы на день, но и увидеть, сколько осталось до их предполагаемого начала или завершения. События подгружаются из *Google*-календаря, с которым синхронизирован планировщик.

Функциональность приложения включает:

– добавление ежедневных дел;

– оригинальный таймер поездок и длительности авиаперелетов;

– возможность планировать приемы лекарств и пищи;

– отслеживание времени, потраченного на тренировку;

– совместную работу с системой Android Wear на смарт-часах и фитнес-браслетах.

Преимуществами утилиты является встроенный в нее функциональный таймер, обратный отсчет и контроль хронометража различных действий. Эти особенности будут полезными для спортсменов, путешественников и деловых людей. При этом планировщик также может синхронизироваться с компьютерами *Apple*, *Windows* и телефонами на базе *Android* и *iOS*.

Последним популярным кандидатом является приложение *Trello*. Приложение *Trello* представляет собой продукт, созданный по тому же принципу, что и программное обеспечение для управления крупными проектами. С его помощью можно создавать задачи для коллектива, семьи и даже интернет-магазина.

Возможности *Trello*:

– создание списков задач для индивидуального и общего использования;

– приглашение в группу коллег, членов семьи и друзей;

– назначение заданий другим пользователям;

– ответы на комментарии;

– привязка карточек к координатам на карте;

– визуализация задач.

Программа поддерживается различными операционными системами – *Microsoft Windows*, *macOS*, *iOS* и *Android*. Среди ее плюсов – возможность

следить за проектами, простой и удобный интерфейс, загрузка файлов и настройка дедлайнов. При этом планировщик не требует платы за использование.

Сравнительная десятибальная характеристика приложений для персонального планирования проектами и приложения, написанного в ходе практики, приведена в таблице 2.1.

Таблица 2.1 – Характеристика приложений для планирования проектов

Параметры	*Any.DO*	*Todoist*	*GTasks*	*Sectograph*	*Trello*	Разработанное приложение
Доступность	6	10	8	10	6	10
Понятный интерфейс	5	7	6	9	7	9
Поддержка устройств	10	10	6	6	10	10
Вложения	7	0	0	0	10	8
Простота	4	9	6	8	7	10
Функционал	9	4	5	5	8	6
Визуализация	4	4	4	9	7	8
Итоги	45	44	35	47	55	61

Из результатов видно что приложения существенно отличаются друг от друга. Большинство функций приложений являются бесплатными, однако в некоторых приложениях часть функций заблокирована либо присутствует реклама. Современные приложения практически все имеют неплохой графический интерфейс, однако большое количество функционала делает его более сложным, чем он должен быть.

Тяжело совместить большое количество функционала и простоту использования приложения. Поэтому пользователь должен сам проанализировать и решить что ему подходит больше.

Часть приложений не имеет браузерной версии и доступна только в мобильных магазинах приложений, что сильно ограничивает их применение. Многие приложения предоставляют большой функционал но не содержат вложений для задач, что является большой проблемой для эффективной организации проекта.

Разработанное приложения является узконаправленным для управления персональными проектами обучения, поэтому не содержит большое количество функционала, однако имеет все необходимые функции для реализации такого рода проектов. Это позволяет ему занять свою нишу и предоставить пользователю более простой и линейный пользовательский интерфейс при том, что организация проектов в приложении имеет древовидную структуру.

2.4 Методы контроля обучения

Контроль знаний и умений является важным элементом процесса обучения. Результативность процесса обучения во многом зависит от тщательности разработки методики контроля знаний. Контроль знаний необходим при всякой системе обучения и любой организации учебного процесса. Это средство управления учебной деятельностью обучающихся. Но для того, чтобы, наряду с функцией проверки, реализовались и функции обучения, необходимо создать определенные условия, важнейшее из которых – объективность проверки знаний.

Объективность проверки знаний предполагает корректную постановку контрольных вопросов, вследствие чего появляется однозначная возможность отличить правильный ответ от неправильного. Кроме того, желательно, чтобы форма проверки знаний позволяла легко выявить результаты [7].

Существует несколько традиционных форм контроля знаний и умений:

– устный или письменный опрос;

– карточки;

– краткая самостоятельная работа;

– практическая или лабораторная работа;

– тестовые задания.

Устная проверка знаний подразумевает наличие человека, проводящего контроль. Устная проверка может быть в форме фронтальной беседы, когда проверяющий задает вопросы всем учащимся. При этом происходит непосредственный контакт этого человека с людьми. При опросе кого-либо из обучающихся все остальные должны внимательно следить за ответом, поправляя и дополняя его. Устная фронтальная проверка не позволяет установить всю глубину усвоенных понятий, но зато в течение короткого времени проверяющий человек уточняет, насколько люди усвоили основные представления об изучаемом материале или объекте, умеют ли обобщать и систематизировать знания, устанавливать связи.

Работе с карточками придается особое значение, так как такая проверка знаний дает возможность дифференцированно подойти к обучающимся, проверить знания большого количества людей.

Карточки, которые предлагаются учащимся, могут быть очень разными по содержанию, объему, оформлению. Кроме того, следует сделать карточки для сложного, среднего и простого уровней, что позволяет использовать «зону ближайшего развития» каждого человека.

Письменная проверка знаний – распространенная форма контроля знаний и умений обучающихся. Она представляет собой перечень вопросов, на которые обучающиеся должны дать незамедлительные и краткие ответы. Время на каждый ответ строго регламентировано и достаточно мало, поэтому сформулированные вопросы должны быть четкими и требовать однозначных, не требующих долгого размышления, ответов. Именно краткость ответов отличает его от остальных форм контроля. С помощью письменной проверки можно проверить ограниченную область знаний обучающихся: буквенные

обозначения, названия единиц, определения, формулировки, связь между величинами, формулировки научных фактов. Именно эти знания могут быть проверены в быстрых и кратких ответах. Письменная проверка не позволяет проверить умения, которыми овладели люди при изучении той или иной темы. Таким образом, быстрота проведения письменной проверки является одновременно как его достоинством, так и недостатком, т.к. ограничивает область проверяемых знаний. Однако эта форма контроля снимает часть нагрузки с остальных форм, а также может быть с успехом применена в сочетании с другими формами контроля.

При кратковременной самостоятельной работе обучающимся также задается некоторое количество вопросов, на которые предлагается дать свои обоснованные ответы. В качестве заданий могут выступать теоретические вопросы на проверку усвоенных знаний; задачи, на проверку умения выполнить расчеты по заданию; задания по моделированию (воспроизведению) конкретных ситуаций, соответствующих технологическим понятиям. В самостоятельной работе могут быть охвачены все виды деятельности кроме создания понятий, т.к. это требует большего количества времени. При этой форме контроля обучающиеся обдумывают план своих действий, формулируют и записывают свои мысли и решения. Понятно, что кратковременная самостоятельная работа требует гораздо больше времени, чем предыдущие формы контроля, и количество вопросов может быть не более трёх, а иногда самостоятельная работа состоит и из одного задания.

Практическая или лабораторная работа – достаточно необычная форма контроля, она требует от обучающихся не только наличия знаний, но еще и умений применять эти знания в новых ситуациях, сообразительности. Лабораторная работа активизирует познавательную деятельность людей. Практическую лабораторную работу целесообразно комбинировать с такими формами контроля, как письменная проверка или тест. Такая комбинация может достаточно полно охватить знания и умения обучающихся при минимальных затратах времени, а также снять при этом трудность длинных письменных высказываний.

В тестовых работах предлагается несколько, обычно четыре или пять, вариантов ответов на вопрос, из которых надо выбрать правильный. Эта форма контроля тоже имеет свои преимущества, неслучайно это одна из наиболее распространенных форм контроля. Обучающиеся не теряют времени на формулировку ответов и их запись, что позволяет охватить большее количество материала за то же время.

Несмотря на все очевидные достоинства, тестовые задания имеют ряд недостатков. Главный из них – это трудность формулирования вариантов ответов на вопросы при их составлении. Если ответы подобраны без достаточного логического обоснования, большинство обучающихся очень легко выбирают требуемый ответ, исходя не из имеющихся у них знаний, а только лишь из простейших логических умозаключений и жизненного опыта. Поэтому бывает трудно или даже невозможно составить удачный тест без теоретической подготовки. Следует также отметить, что тестовые задания дают

возможность проверить ограниченную область знаний обучающихся, оставляя в стороне деятельность по созданию объектов труда, воспроизведению конкретных действий, соответствующих практическим навыкам и т.п. По результатам выполнения тестов нельзя проверить умения обучающихся решать комбинированные задачи, а также способности построения логически связанного ответа в устной форме.

Итак, эффективность контроля знаний и умений во многом зависит от умения правильно организовать обучение и грамотно выбрать ту или иную форму проведения контроля. И для правильной организации контроля обучения, также как и для организации проектов, существует большое количество разработанных программ [8].

2.5 Программные решения для контроля обучения

Первой популярной программой для контроля обучения является *Study Smarter*. В этом приложении можно создавать обучающие карточки и учебные заметки в очень быстрые сроки. Также оно даёт доступ к общим учебным материалам и учебникам от ведущих издателей. Кроме того в этом приложении можно настроить свой индивидуальный учебный план.

Из особенностей приложения можно выделить:

– возможность создавать карточки для запоминания;
– функционал по созданию и прохождению тестов;
– возможность сохранять и просматривать конспекты лекций;
– создание учебных заметок и пособий;
– проектирование своего плана обучения;
– возможность устанавливать таймеры для напоминания об обучении;
– функционал создания учебных групп пользователей;
– возможность загружать и создавать учебники.

Из минусов приложения необходимо выделить поддержку приложения только на мобильных устройствах. А также приложение не имеет поддержки русского языка.

Следующее приложение для обучения является более специализированным, так как предназначено для контроля обучения программистов. Это приложение называется *Solo Learn*. Оно позволяет людям, которые хотят обучиться программированию сделать это проще и веселее. Приложение поддерживается всеми платформами, переведено на русский язык и также имеет привлекательный и простой интерфейс пользователя.

Это приложение сконцентрировано на практических навыках, так как пользуясь им человек начинает писать первый код сразу же после начала обучения. Также приложение имеет очень хорошо организованные подзадачи, что позволяет пользователю концентрироваться на небольших частях обучения.

Приложение содержит ряд курсов, в конце которых пользователь получает сертификат о прохождении и готовый проект в резюме. Эти курсы направлены как на профессионалов, которые хотят изучить новые технологии, так и на студентов, которым необходимо, например, сдать экзамен.

Из особенностей приложения необходимо выделить следующее:

– направленность на практику;

– чёткое направление обучения – программирование;

– современный и удобный пользовательский интерфейс;

– кроссплатформенность;

– коммуникации с людьми в процессе обучения;

– своя песочница для работы с кодом;

– интерактивность;

– функционал достижений.

Минусы приложения следуют из его плюсов. Оно является очень узконаправленным и подходит только для обучения связанного с программированием. Однако явных минусов приложение не имеет.

Следующим приложением является *Khan Academy*. Это приложение позволяет получить доступ к большой базе знаний из любой точки мира. Это бесплатный сервис, работающий на благотворительности и не требующий покупки каких-либо услуг.

В данном приложении можно заниматься как самообучением, так и обучением с учителем. Для этого учителю необходимо зарегистрировать свой класс и добавить в него учеников. Кроме учителя и ученика в приложении также есть роль родителя. Родитель может смотреть результаты обучения и просматривать программу обучения.

Из особенностей приложения можно выделить:

– приложения является полностью бесплатным;

– содержит большую базу знаний по многим школьным направлениям;

– поддерживается на любых устройствах;

– имеет поддержку русского языка и переведено на множество других языков;

– удобный и простой пользовательский интерфейс, который будет понятен людям любого возраста.

Минусом приложения является его направленность только на школьные дисциплины, однако благодаря этому всё, что связано с этими дисциплинами, имеет отличную организацию.

Следующие два приложения реализуют концепцию ментальных карт. Ментальная карта или диаграмма связей – техника структуризации концепций с использованием графической записи в виде диаграммы. Популяризована британским психологом *Tony Buzan*.

Ментальная карта реализуется в виде древовидной схемы, на которой изображены слова, идеи, задачи или другие понятия, связанные ветвями, отходящими от центрального понятия или идеи.

Ментальные карты могут использоваться как:

– инструменты управления знаниями;

– при обучении или самообучении;

– для записи результатов мозгового штурма.

Приложения *Mindomo* и *Mind Meister* имеют очень широкий спектр применения. Их можно использовать для преподавания и обучения, для работы,

для личных целей и т.д. Создавая ассоциативные карты, обучающиеся изучают информацию, определяют важные моменты и решают, как они связаны с уже имеющимися данными. Этот процесс отлично развивает критическое мышление.

Благодаря ментальным картам легче понять, как понятия соотносятся друг с другом и научиться находить новые связи и закономерности. Также легче проводить собственные исследования и структурировать эссе. Карту можно трансформировать в реальный план действий и приступить к его исполнению.

Кроме обучения, ассоциативные карты можно интегрировать в рабочий процесс, чтобы провести мозговой штурм, стратегическое планирование и использовать нестандартное мышление, а также перейти на качественно новый уровень решения проблем.

Эти два приложения имеют много похожих особенностей и практически не отличаются, однако всё же есть пару моментов, которые необходимо выделить:

– *Mindomo* дороже своего конкурента, однако имеет больший рейтинг удовлетворённости пользователей;

– *Mindomo* интегрируется с большим количеством приложений и поддерживает большее количество языков;

– *Mindomo* имеет более широкий набор функций.

Результаты сравнения приложений для контроля процесса обучения представлены в таблице 2.2.

Таблица 2.2 – Характеристика приложений для контроля обучения

Параметры	*Study Smarter*	*Solo Learn*	*Khan Academy*	*Mindomo*	*Mind Meister*	Разработанное приложение
Доступность	6	6	10	6	7	10
Понятный интерфейс	8	10	10	7	7	9
Поддержка устройств	6	10	10	10	10	10
Интеграция	0	4	4	10	6	0
Простота	8	8	8	6	5	10
Функционал	6	7	6	8	7	3
Визуализация	6	7	5	10	10	8
Итоги	40	52	53	57	52	50

Из таблицы видно, что приложение *Mindomo* является очень хорошим вариантом для контроля процессов обучения. Разработанное приложение предоставляет средний по меркам похожих приложений функционал для контроля процессов обучения.

2.6 Архитектура программного комплекса

Разработанный программный комплекс имеет микросервисную архитектуру. Микросервисная архитектура – распространенный подход к разработке программного обеспечения, когда приложение разбивается на небольшие автономные компоненты(микросервисы) с четко определенными интерфейсами. Эта архитектура часто ставится в противопоставление с монолитной архитектурой приложений.

Монолиты – это приложения, построенные как единое целое, где вся логика по обработке запросов помещается внутрь одного процесса. Разумеется, монолиты могут иметь модульную структуру – содержать отдельные классы, функции, области имён. Но связи между этими модулями настолько сильны, что изменение каждого из них неизбежно отражается на работе приложения в целом [9]. Ниже приведены преимущества микросервисной архитектуры относительно монолитной.

Первым неоспоримым преимуществом такой архитектуры является простота развертывания приложения. Можно развертывать только изменяющиеся микросервисы, независимо от остальной системы, что позволяет производить обновления чаще и быстрее.

Следующим преимуществом является оптимальность масштабирования. Можно расширять только те сервисы, которые в этом нуждаются, то есть сервисы с наименьшей производительностью, оставляя работать остальные части системы на менее мощном оборудовании.

Далее необходимо отметить устойчивость к сбоям. Отказ одного сервиса не приводит к остановке системы в целом. Когда же ошибка исправлена, необходимое изменение можно развернуть только для соответствующего сервиса – вместо повторного развертывания всего приложения. Правда, для этого еще на этапе проектирования микросервисов потребуется тщательно продумать связи между ними для достижения максимальной независимости друг от друга, а также заложить возможность корректного оповещения пользователя о временной недоступности определенного сервиса без ущерба для всей системы.

Также микросервисная архитектура позволяет применять более широкий спектр технологий для приложения. Можно подбирать различные наборы технологий, оптимальные для решения задач, стоящих перед отдельными сервисами.

Для микросервисной архитектуры нет необходимости в большой команде разработчиков. При разработке микросервисов команды принято закреплять за конкретными бизнес-задачами (и сервисами, соответственно). Такие команды, как правило, показывают большую эффективность, а управлять ими легче.

Микросервисы легко заменять при необходимости, в отличие от модулей монолитов.

И заключительным преимуществом микросервисов является независимость моделей данных. Каждый микросервис, как правило, использует

собственное хранилище данных – поэтому изменение модели данных в одном сервисе не влияет на работу остальных [10].

Разработанное приложение имеет пользовательский интерфейс, написанный на веб фреймворке *Angular*, который взаимодействует с двумя сервисами.

Первый сервис предназначен для авторизации и аутентификации пользователей. Пользовательские данные хранятся в нереляционной базе данных *MongoDB*.

Второй сервис является ресурсным сервисом. Этот сервис отвечает за хранение данных о проектах пользователей, их обработку и поставку клиенту. Этот сервис хранит в реляционной базе данных *MSSQL Server*.

Графическое представление архитектуры приложения изображено на рисунке 2.1.

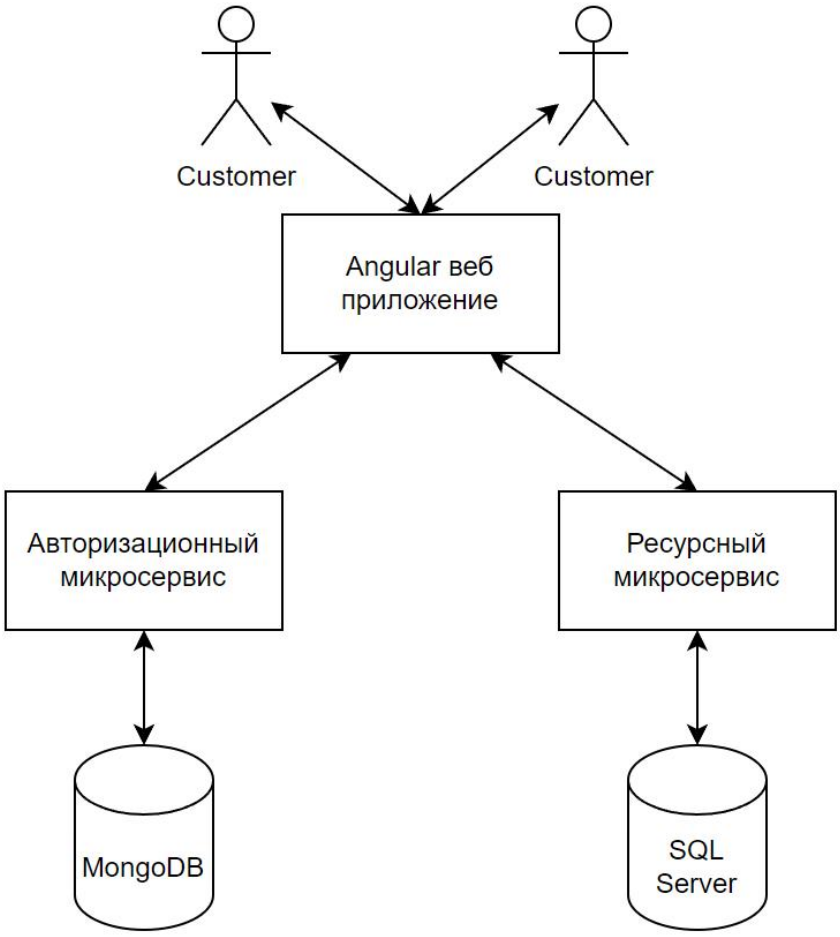

Рисунок 2.1 – Архитектура приложения

Авторизационный микросервис не взаимодействует с ресурсным микросервисом напрямую. Для того, что бы ресурсный микросервис знал какой пользователь к нему обращается, в приложении происходит постоянный обмен авторизационным токеном, хранящим основную пользовательскую информацию, полученную авторизационным сервисом из его базы данных. Всю остальную информацию ресурсный микросервис берет из своей базы данных.

2.7 Структуры баз данных

Авторизационная база данных является нереляционной и реализована при помощи технологии *MongoDB*. В этой базе данных вся информация хранится в виде коллекций документов. В данном случае документом является сущность пользователя с его полями. Представление сущности пользователя изображено на рисунке 2.2.

Рисунок 2.2 – Сущность пользователя

Коллекция пользователей хранит документы о пользователях, которые содержат следующие поля:

– *_id* – строковый идентификатор пользователя;

– *Email* – почта пользователя;

– *PasswordHash* – хеш пароля пользователя;

– *Roles* – массив чисел, представляющих роли пользователя.

Ресурсная база данных имеет реляционную структуру. Она реализована с помощью *MS SQL Server*. Для указания связи между пользователем и сущностью в ресурсной базе данных используется поле *accountId*. Это строковое поле и оно совпадает с идентификатором пользователя в базе данных пользователей.

Например, это поле используется для указания связи между пользователями и их профилями. Каждый пользователь имеет свой профиль, который возвращается ему при обращении в базу данных с его идентификатором.

Также данное идентификационное поле используется для указания связей между пользователями и их областями проектов и между пользователями и их вложениями, так как у каждого пользователя свои наборы этих сущностей. Остальные сущности базы данных не зависят от идентификатора пользователя, однако имеют связи с зависящими сущностями. Это хорошо с точки зрения хранения информации, однако немного замедляет получение данных из базы.

Сущности базы данных и их связи изображены на рисунке 2.3.

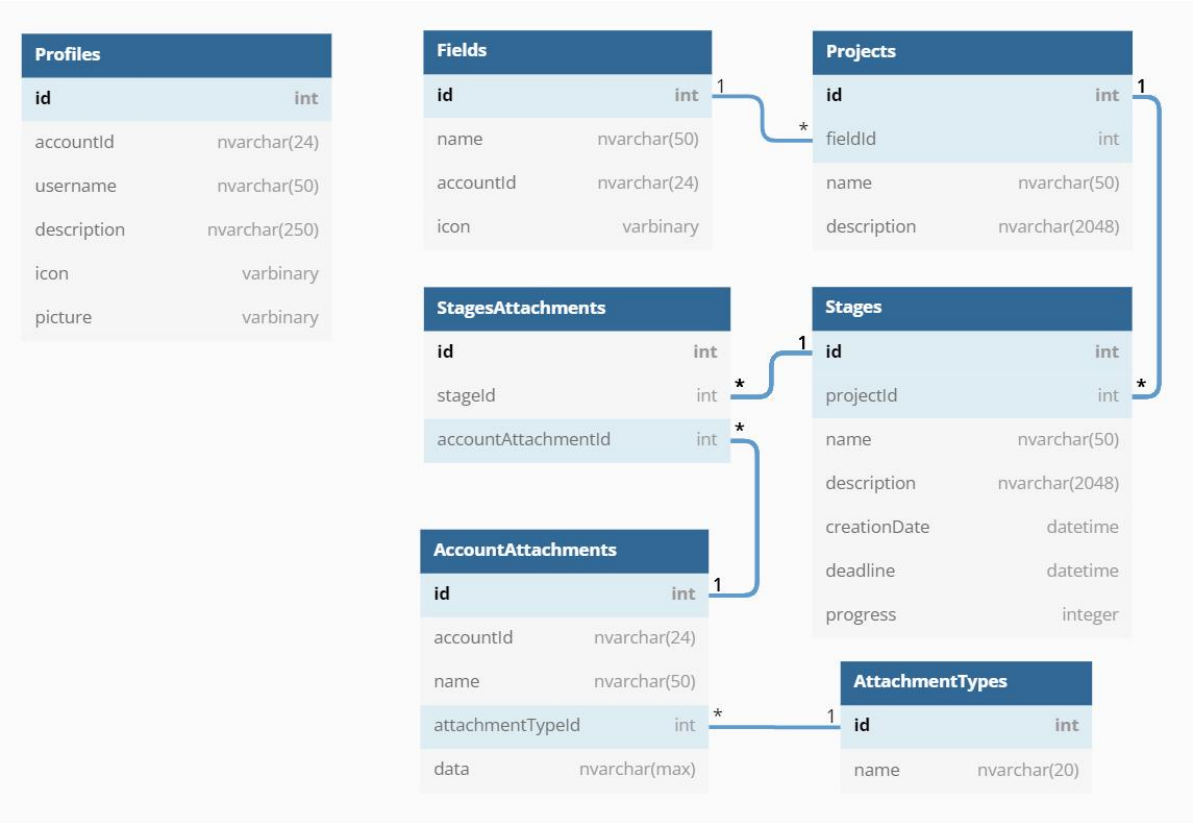

Рисунок 2.3 – Связи сущностей базы данных

В результате проектирования базы данных были получены следующие сущности:

– «Профиль пользователя»;
– «Область проектов»;
– «Проект»;
– «Стадия»;
– «Вложение в стадии»;
– «Вложение»;
– «Тип вложения».

В сущности «Профиль пользователя» определены следующие атрибуты: «Идентификатор пользователя», «Имя профиля», «Описание профиля», «Иконка профиля» и «Изображение профиля». В качестве уникального идентификатора служит «Идентификатор профиля». Более подробное описание каждого атрибута приведено в таблице 2.3.

Таблица 2.3 – Описание атрибутов сущности «Профиль пользователя»

Атрибуты	Описание домена	Тип данных
Идентификатор профиля	Уникальный инкрементируемый идентификатор. Является первичным ключом	Целочисленный

Продолжение таблицы 2.3

Атрибуты	Описание домена	Тип данных
Идентификатор пользователя	Содержит идентификатор пользователя, связанного с этим профилем	Строка
Имя профиля	Содержит имя профиля пользователя	Строка
Описание профиля	Содержит описание профиля пользователя	Строка
Иконка профиля	Содержит небольшое изображение, представляющее иконку профиля пользователя	Массив байт
Изображение профиля	Содержит изображение профиля пользователя	Массив байт

В сущности «Область проектов» определены следующие атрибуты: «Название области проектов», «Идентификатор пользователя» и «Иконка области проектов». В качестве уникального идентификатора служит «Идентификатор области проектов». Более подробное описание каждого атрибута приведено в таблице 2.4.

Таблица 2.4 – Описание атрибутов сущности «Область проектов»

Атрибуты	Описание домена	Тип данных
Идентификатор области проектов	Уникальный инкрементируемый идентификатор. Является первичным ключом	Целочисленный
Название области проектов	Содержит наименование области проектов	Строка
Идентификатор пользователя	Содержит идентификатор пользователя, связанного с этой областью проектов	Строка
Иконка области проектов	Содержит изображение области проектов	Массив байт

В сущности «Проект» определены следующие атрибуты: «Идентификатор области проектов», «Название проекта» и «Описание проекта». В качестве уникального идентификатора служит «Идентификатор проекта». Более подробное описание каждого атрибута приведено в таблице 2.5.

Таблица 2.5 – Описание атрибутов сущности «Проект»

Атрибуты	Описание домена	Тип данных
Идентификатор проекта	Уникальный инкрементируемый идентификатор. Является первичным ключом	Целочисленный
Идентификатор области проектов	Содержит идентификатор области проектов. Является внешним ключом	Целочисленный
Название проекта	Содержит название проекта	Строка
Описание проекта	Содержит описание проекта	Строка

В сущности «Стадия» определены следующие атрибуты: «Идентификатор проекта», «Название стадии», «Описание стадии», «Дата создания», «Последний срок сдачи» и «Прогресс выполнения». В качестве уникального идентификатора служит «Идентификатор стадии». Более подробное описание каждого атрибута приведено в таблице 2.6.

Таблица 2.6 – Описание атрибутов сущности «Стадия»

Атрибуты	Описание домена	Тип данных
Идентификатор стадии	Уникальный инкрементируемый идентификатор. Является первичным ключом	Целочисленный
Идентификатор проекта	Содержит идентификатор области проектов. Является внешним ключом	Целочисленный
Название стадии	Содержит название стадии	Строка
Описание стадии	Содержит описание стадии	Строка
Дата создания	Содержит дату создания стадии	Дата
Последний срок сдачи	Содержит дату ожидаемого завершения проекта	Дата
Прогресс выполнения	Содержит число от нуля до ста, обозначающее прогресс выполнения проекта	Целочисленный

В сущности «Вложение» определены следующие атрибуты: «Идентификатор пользователя», «Название вложения», «Идентификатор типа вложения» и «Данные вложения». В качестве уникального идентификатора служит «Идентификатор вложения». Более подробное описание каждого атрибута приведено в таблице 2.7.

Таблица 2.7 – Описание атрибутов сущности «Вложение»

Атрибуты	Описание домена	Тип данных
Идентификатор вложения	Уникальный инкрементируемый идентификатор. Является первичным ключом	Целочисленный
Идентификатор пользователя	Содержит идентификатор пользователя, связанного с этим вложением	Целочисленный
Название вложения	Содержит название вложения	Строка
Идентификатор типа вложения	Содержит идентификатор типа вложения. Является внешним ключом	Целочисленный
Данные	Содержит данные пользовательского вложения	Строка

В сущности «Вложение в стадии» определены следующие атрибуты: «Идентификатор стадии», «Идентификатор вложения». В качестве уникального идентификатора служит «Идентификатор вложения в стадии». Более подробное описание каждого атрибута приведено в таблице 2.8.

Таблица 2.8 – Описание атрибутов сущности «Вложение в стадии»

Атрибуты	Описание домена	Тип данных
Идентификатор вложения в стадии	Уникальный инкрементируемый идентификатор. Является первичным ключом	Целочисленный
Идентификатор стадии	Содержит идентификатор стадии. Является внешним ключом	Целочисленный
Идентификатор вложения	Содержит идентификатор стадии. Является внешним ключом	Целочисленный

В сущности «Тип вложения» определен аттрибут «Название типа вложения». В качестве уникального идентификатора служит «Идентификатор

типа вложения». Более подробное описание каждого атрибута приведено в таблице 2.9.

Таблица 2.9 – Описание атрибутов сущности «Тип вложения»

Атрибуты	Описание домена	Тип данных
Идентификатор типа вложения	Уникальный инкрементируемый идентификатор. Является первичным ключом	Целочисленный
Название типа вложения	Содержит наименование типа вложения	Строка

Все таблицы базы данных были приведены в третью нормальную форму для более эффективного хранения и изменения постоянно меняющейся пользовательской информации.

2.8 Система аутентификации пользователей

Система аутентификации пользователей представляет собой отдельный микросервис, который хранит данные о пользователях и предоставляет токены доступа к приложению.

Токены имеют тип *JWT*. *JSON Web Token* (*JWT*) – это открытый стандарт (*RFC* 7519) для создания токенов доступа, основанный на формате *JSON*. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

В качестве алгоритма шифрования используется *HmacSha256*. *HMACSHA256* – это тип хэш-алгоритма с ключом, созданный на основе хэш-функции *SHA*-256 и используемый в качестве кода аутентификации сообщений на основе хэша (*HMAC*).

Благодаря такой технологии авторизации и аутентификации пользователей микросервисы становятся менее зависимыми друг от друга и реализуется подход проектирования сервера без сохранения состояния пользователя.

2.9 Серверная часть приложения

Кроме авторизационного микросервиса в состав серверной части также входит ресурсный сервис. Он занимается непосредственно функционалом приложения:

– получение данных из базы данных;
– обработка и поставка данных;
– взаимодействие с пользовательским веб клиентом при помощи *REST*;
– обработка ошибок.

Оба реализованных микросервиса написаны при помощи технологии *ASP.NET* 6. Это версия фреймворка работает с платформой *.NET* шестой версии и языком *C#* десятой версии. Эти технологии являются одними из самых эффективных и современных, что позволяет быстро разрабатывать высоконагруженные веб приложения.

Для доступа к базе данных *MongoDB* авторизационный сервис использует *MongoDB* драйвер для языка *C#*. Этот драйвер является оболочкой над стандартными командами *MongoDB*. При помощи него можно быстро и безошибочно манипулировать коллекциями и документами в нереляционной базе данных. Основным классом для взаимодействия с этой базой данных является *BsonDocument*. По сути этот класс является представлением *JSON* документа в двоичном формате, с которым взаимодействует *MongoDB*.

Над драйвером был реализован шаблон репозиторий для разделения ответственности и более удобного тестирования. Этот репозиторий манипулирует моделью приложения, представляющей собой пользователя.

Кроме модели базы данных сервис также содержит модели, с которыми он взаимодействует с клиентским приложением. Это сделано для скрытия от клиента информации о хранении сущности в базе данных. Такими моделями являются модель токена приложения и модель авторизационной формы пользователя. Модель токена приложения авторизационный сервер возвращает при входе пользователя в систему. Модель формы авторизации сервис получает, когда к нему обращается клиентское приложение с запросом на авторизацию или регистрацию пользователя.

Ресурсный сервис взаимодействует с реляционной базой данных *SQL Server* при помощи *ORM* технологии *Entity Framework*. Ресурсный сервис содержит все модели сущностей базы данных и контекст базы данных для обращения к ней. Над контекстом реализованы шаблоны репозиторий и *UnitOfWork*. Это также сделано для удобного тестирования сервиса и разделения ответственности его классов.

Также ресурсный сервер содержит информацию о миграциях базы данных приложения. Это необходимо для контроля изменения сущностей базы данных и их отношений.

При помощи инструмента *AutoMapper* приложение понятно преобразует сущности базы данных в сущности, которые сервис принимает от клиента и посылает ему. Для работы этого инструмента необходимо разработать профили преобразования сущностей и внедрить их в приложение.

Вся бизнес логика микросервисов выделена в специальные классы, называемые службами. Эти службы обращаются к репозиториям, получая необходимые данные, обрабатывает их и отдают контроллерам для возвращения клиенту.

Кроме того ресурсный микросервис содержит классы-инструменты, которые содержат бизнес логику, не связанную с непосредственным получением и обработкой данных. Эти инструменты нужны для выделения сложной бизнес логики в отдельные классы и их применения в службах.

Микросервисы взаимодействуют с клиентским приложением при помощи протокола *HTTP* и архитектуры *REST*. Для реализации непосредственного взаимодействия с клиентом используются классы контроллеры. Они принимают запросы пользователей, обращаются к службам приложения и возвращают необходимые данные обратно клиенту. Если служба завершила выполнения метода с ошибкой, контроллер решает что необходимо вернуть клиенту. Чаще всего это краткая информация об ошибке или конкретное указание почему запрос завершился с ошибкой.

2.10 Клиентская часть приложения

Клиентская часть приложения написана при помощи веб фреймворка *Angular*. *Angular* – открытая и свободная платформа для разработки веб-приложений, написанная на языке *TypeScript*, разрабатываемая командой из компании *Google*, а также сообществом разработчиков из различных компаний. Этот фреймворк является одним из самых популярных и используемых. Он имеет чёткую архитектуру и использует типизируемый язык, что позволяет делать меньше ошибок в ходе разработки.

Структурными единицами веб приложения являются:

– файлы конфигурации;

– компоненты;

– сервисы;

– модели.

В файлах конфигурации содержится вся информация о клиентском приложении: адреса микросервисов, зависимости, версия и т.д.

Компоненты приложения – это классы, которые содержат ссылки на код гипертекстовой разметки, стили для этой разметки и реализуют логику взаимодействия с пользователем. Они являются частями веб приложения и часто их взаимодействие между собой представляет графовую структуру.

Эти компоненты обращаются к сервисам или службам. Эти службы содержат бизнес логику веб приложения: код отправки запросов к серверу, код валидации данных, код обработки данных и так далее.

Эти службы манипулируют моделями. Модели – это классы, хранящие информацию о сущностях приложения. Эти модели по своим аттрибутам аналогичны моделям, которые сервер отправляет клиенту.

Разработанное приложение содержит две модели для авторизации пользователя: модель авторизационной формы и модель токена. Также приложение содержит пять моделей для реализации основного функционала приложения: вложение, область проектов, профиль, проект, стадия.

На каждую из этих моделей присутствует служба, которая получает и отправляет эти модели при взаимодействии с сервером.

Службы используются в компонентах приложения. Можно выделить несколько компонентов верхнего уровня приложения. Корневым компонентом является сам компонент приложения *app*. В него входят компоненты, представляющие основные модули веб приложения:

– *auth* – компонент для авторизации и регистрации пользователей;

– *footer* – компонент для реализации нижней части страницы;

– *header* – компонент для реализации верхней части страницы;

– *main* – компонент приложения, в котором содержится вся основная логика взаимодействия с пользователем;

– *nav* – компонент навигации в приложении.

Модули веб приложения взаимодействуют друг с другом при помощи специальных привязок данных и событий. Приложение является одностраничным приложением. Весь контент приложения формируется при помощи *JavaScript*.

2.11 Модульное тестирование приложения

Для того, что бы убедиться что такая сложная система будет работать как ожидалось, необходимо периодически проводить её тестирование. Проводить тестирование вручную не является возможным из-за постоянно возрастающей сложности системы. Решением является написание модульных тестов.

Практически любая современная технология позволяет писать модульные тесты, так как без них невозможно продолжать разработку продукта.

Для реализации модульных тестов в *C#* принято использовать *xUnit*. *xUnit.net* – это бесплатный инструмент модульного тестирования с открытым исходным кодом для *.NET*, написанный первоначальным автором *NUnit*.

Эта технология позволяет провести быстрые асинхронные тесты для различных модулей кода, слабо связанных между собой.

Чаще всего вместе с *xUnit* используется технология *Moq*. Она позволяет заменить доступ к источнику данных на описание возвращаемых данных в случае вызова определенного метода. Это необходимо для того, что бы у каждого теста был свой контекст исполнения и они не пересекались друг с другом и не влияли на результаты выполнения друг друга.

Angular также предоставляет функционал для тестирования работы компонентов. При генерации компонентов вместе с ними создаются файлы разметки, стилей и тестов. То есть в этом фреймворке сразу подразумевается, что программист будет тестировать то, что он разработал.

Эти тесты работают похожим образом, как и модульные тесты на сервере, однако они обладают более широким контекстом исполнения и выполняются вместе с остальными компонентами приложения.

2.12 Верификация работы приложения

Однако модульных тестов недостаточно для верификации работы приложения. Чаще всего разрабатывается дополнительная документация по приложению. По этой документации проводятся ручные тесты или пишутся автоматизированные тесты, моделирующие взаимодействие реального пользователя с системой.

Для взаимодействия с системой пользователь должен зарегистрироваться или авторизироваться. Внешний вид окна авторизации изображён на рисунке Д.1.

После вхождения в систему пользователь видит главную страницу приложения. Внешний вид главной страницы приложения изображён на рисунке Д.2.

Далее пользователь может зайти в свой профиль и отредактировать необходимую информацию. Внешний вид профиля пользователя изображён на рисунке Д.3.

После этого пользователь может зайти во вкладку проекты и добавить необходимые ему области проектов. Области проектов изображены на рисунке Д.4.

Далее пользователь может нажать на область проектов и выбрать проект, с которым он хочет взаимодействовать. Выбранная область проектов и проект изображены на рисунке Д.5.

Пользователь может добавить удалить и редактировать область проектов, проект и стадию проекта. Также пользователь может раскрыть стадию проекта и просмотреть её информацию и вложения. Внешний вид раскрытой стадии изображён на рисунке Д.6.

При нажатии на кнопку вложений открывается список вложений пользователя. Внешний вид окна вложений изображён на рисунке Д.7.

В данном окне пользователь может добавить вложение в стадию, редактировать, удалить или создать новое вложение. Внешний вид окна создания нового вложения изображён на рисунке Д.8.

Верификация работы приложения показала, что всё работает так как нужно и все функции реализованы.

ЗАКЛЮЧЕНИЕ

Во время прохождения практики были изучены основные сведения о структуре предприятия. Были получены практические знания и навыки работы в организации.

Во время практики была собрана вся необходимая информация для выполнения индивидуального задания «Приложение для персонального планирования и контроля процессов обучения».

Были проанализированы методы решения задач планирования проектов и контроля процессов обучения. Кроме того, было проведено исследование готовых программных продуктов, которые реализуют функционал для реализации этих целей.

Итогом практики стало приложение для персонального планирования и контроля процессов обучения. Разработанное приложение призвано оптимизировать и упростить процессы обучения сотрудников ИТ-предприятий и людей, постоянно изучающих новые области.

Вначале была спроектирована архитектура программного комплекса. Была выбрана микросервисная архитектура, что позволило написать расширяемую и слабосвязанную систему. Было решено выделить два микросервиса и веб клиент. Первый микросервис отвечает за авторизацию клиентов, второй отвечает за взаимодействие с ними.

Далее были разработаны структуры баз данных приложения. Было решено для авторизационного сервера использовать нереляционную базу данных *MongoDB*. Для ресурсного сервера была использована реляционная база данных *SQL Server*.

После этого было принято решение об авторизационной системе. Она была реализована при помощи токенов доступа. Благодаря этому сервер, взаимодействующий с пользователем не хранит состояния авторизации.

После этого были разработаны серверы. Сначала был разработан авторизационный сервер. После этого был разработан ресурсный сервер. Были написаны тесты и проведено ручное тестирование пользовательского *API*.

Далее, при помощи фреймворка *Angular*, было разработано клиентское приложение, осуществляющее взаимодействие с пользователем и отправляющее запросы к микросервисам.

В конце прохождения практики была произведена верификация программного обеспечения и написан отчёт.

Благодаря практике, знания, полученные в университете, были обобщены и спроецированы на рабочий процесс, что поспособствовало лучшему их закреплению.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. IBA Gomel Park [Электронный ресурс]. – Режим доступа: https://iba.by/about-iba-group/iba-development-center/ibagomel/. – Дата доступа: 14.04.2022.
2. История предприятия IBA [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%93%D1%80%D1%83%D0%BF%D0%BF%D0%B0_IBA. – Дата доступа: 14.04.2022.
3. Корпоративная социальная ответственность IBA [Электронный ресурс]. – Режим доступа: https://iba.by/about-iba-group/csr/. – Дата доступа: 14.04.2022.
4. Деятельность IBA [Электронный ресурс]. – Режим доступа: https://iba.by/about-iba-group/executive-summary/. – Дата доступа: 14.04.2022.
5. Бизнес-планирование: Учебное пособие / Под ред. В.З. Черняка, Г.Г. Чараева. – М.: Юнити, 2016. – 591 с.
6. Афитов, Э.А. Планирование на предприятии: Учебник / Э.А. Афитов. – М.: Инфра-М, 2018. – 672 с.
7. Лапыгин, Ю.Н. Методы активного обучения: Учебник и практикум / Ю.Н. Лапыгин. – Люберцы: Юрайт, 2016. – 248 с.
8. Мицкевич, Н.И. Методы активного обучения взрослых: Учебно-методическое пособие / Н.И. Мицкевич. – Мн.: РИВШ, 2012. – 72 с.
9. Кристиан, Хорсдал Микросервисы на платформе. NET / Хорсдал Кристиан. – М.: Питер, 2018. – 442 с.
10. Сэм, Ньюмен Создание микросервисов. Руководство / Ньюмен Сэм. – М.: Питер, 2016. – 145 с.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг основных классов авторизационного сервера

```
using System.ComponentModel.DataAnnotations;
using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;

namespace Auth.Models
{
    public class Account
    {
        [BsonRepresentation(BsonType.ObjectId)]
        public string? Id { get; set; }

        [EmailAddress]
        public string Email { get; set; } = null!;

        public string PasswordHash { get; set; } = null!;

        public Role[] Roles { get; set; } = null!;
    }

    public enum Role
    {
        User,
        Admin
    }
}

using Auth.Data.Interfaces;
using Auth.Models;
using MongoDB.Bson;
using MongoDB.Driver;

namespace Auth.Data.Implementations
{
    public class MongoAccountRepository : IAccountRepository
    {
        private IMongoCollection<Account> accounts;

        public MongoAccountRepository(string connection)
        {
            var mongoConnection = new MongoUrlBuilder(connection);
            MongoClient client = new MongoClient();
            IMongoDatabase db = client.GetDatabase(mongoConnection.DatabaseName);
            accounts = db.GetCollection<Account>("accounts");
        }

        public async Task<Account> GetByIdAsync(string id)
        {
            return await accounts.Find(new BsonDocument("_id", new ObjectId(id)))
                .FirstOrDefaultAsync();
        }

        public async Task<Account> GetByEmailAsync(string email)
        {
            return await accounts.Find(new BsonDocument("Email", email))
```

```
                .FirstOrDefaultAsync();
        }

        public async Task CreateAsync(Account a)
        {
            await accounts.InsertOneAsync(a);
        }
    }
}

using Auth.Models;

namespace Auth.Data.Interfaces
{
    public interface IAccountRepository
    {
        public Task<Account> GetByIdAsync(string id);

        public Task<Account> GetByEmailAsync(string email);

        public Task CreateAsync(Account a);
    }
}

using Auth.Data;
using Auth.Services;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Auth.UIModels;

namespace Auth.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthController : ControllerBase
    {
        private readonly AuthService authService;

        public AuthController(AuthService authService)
        {
            this.authService = authService;
        }

        [Route("signIn")]
        [HttpPost]
        public async Task<IActionResult> SignIn([FromBody]UIModels.AuthData request)
        {
            if(ModelState.IsValid)
            {
                try
                {
                    var account = await authService.SignInAsync(request);
                    return Ok(authService.GenerateJWT(account));
                }
                catch (Exception exc)
```

```
                {
                    return BadRequest(exc.Message);
                }
            }
            else
            {
                return BadRequest();
            }
        }

        [Route("signUp")]
        [HttpPost]
             public  async  Task<IActionResult>  SignUp([FromBody]  UIModels.AuthData
request)
        {
            if(ModelState.IsValid)
            {
                try
                {
                    var account = await authService.SignUpAsync(request);
                    return Ok(authService.GenerateJWT(account));
                }
                catch(Exception exc)
                {
                    return BadRequest(exc.Message);
                }
            }
            else
            {
                return BadRequest();
            }
        }
    }
}

using Auth.Data;
using Auth.Data.Interfaces;
using Auth.Models;
using Auth.UIModels;
using Common;
using Microsoft.Extensions.Options;
using Microsoft.IdentityModel.Tokens;
using MongoDB.Bson;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace Auth.Services
{
    public class AuthService
    {
        private readonly IOptions<AuthOptions> authOptions;
        private readonly IAccountRepository rep;
        private readonly PasswordHasherService passwordHasher;

        /// <summary>
```

```
        /// User authentication service
        /// </summary>
        /// <param name="authOptions">Authentication options</param>
        /// <param name="rep">Repository with accounts data</param>
        /// <param name="passwordHasher">Password hasher</param>
        public AuthService(IOptions<AuthOptions> authOptions,
            IAccountRepository rep,
            PasswordHasherService passwordHasher)
        {
            this.authOptions = authOptions;
            this.rep = rep;
            this.passwordHasher = passwordHasher;
        }

        /// <summary>
        /// Get user account by authentication data
        /// </summary>
        /// <param name="auth">Authentication data</param>
        /// <returns>User account</returns>
        public async Task<Account> SignInAsync(UIModels.AuthData auth)
        {
            var account = await rep.GetByEmailAsync(auth.Email);

               if (account == null) throw new ArgumentException("Неверный почтовый
адрес");
             if (!passwordHasher.Verify(auth.Password, account.PasswordHash)) throw
new ArgumentException("Неверный пароль");

            return account;
        }

        /// <summary>
        /// Create and get user account by authentication data
        /// </summary>
        /// <param name="auth">Authentication data</param>
        /// <returns>New user's account</returns>
        public async Task<Account> SignUpAsync(UIModels.AuthData auth)
        {
            var account = await rep.GetByEmailAsync(auth.Email);

             if (account != null) throw new ArgumentException("Пользователь с такой
почтой уже зарегистрирован");

            account = new Account
            {
                Id = new BsonObjectId(ObjectId.GenerateNewId()).ToString(),
                Email = auth.Email,
                PasswordHash = passwordHasher.Hash(auth.Password),
                Roles = new Role[] { Role.User }
            };
            await rep.CreateAsync(account);

            return account;
        }

        /// <summary>
        /// Generate JSON Web Token for user's account
        /// </summary>
        /// <param name="account">User's account</param>
```

```
        /// <returns>JSON Web Token</returns>
        public Token GenerateJWT(Account account)
        {
                                if(string.IsNullOrWhiteSpace(account.Id)) throw new
ArgumentException("Токен можно сгенерировать только для аккаунта с
идентификатором.");

            var authParams = authOptions.Value;

            var securityKey = authParams.GetSymmetricSecurityKey();
                        var credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);

            var claims = new List<Claim>()
            {
                new Claim(JwtRegisteredClaimNames.Email, account.Email),
                new Claim(JwtRegisteredClaimNames.Sub, account.Id.ToString())
            };

            foreach (var role in account.Roles)
            {
                claims.Add(new Claim("role", role.ToString()));
            }

            var token = new JwtSecurityToken(authParams.Issuer,
                authParams.Audience,
                claims,
                expires: DateTime.Now.AddSeconds(authParams.TokenLifetime),
                signingCredentials: credentials);

            return new Token {
                AccessToken = new JwtSecurityTokenHandler().WriteToken(token)
            };
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Auth.Services
{
    public class PasswordHasherService
    {
        /// <summary>
        /// Size of salt
        /// </summary>
        private const int saltSize = 16;

        /// <summary>
        /// Size of hash
        /// </summary>
        private const int HashSize = 20;

        /// <summary>
        /// Creates a hash from a password
```

```
        /// </summary>
        /// <param name="password">The password</param>
        /// <param name="iterations">Number of iterations</param>
        /// <returns>The hash</returns>
        public string Hash(string password, int iterations)
        {
            // Create salt
            byte[] salt = RandomNumberGenerator.GetBytes(saltSize);

            // Create hash
            var pbkdf2 = new Rfc2898DeriveBytes(password, salt, iterations);
            var hash = pbkdf2.GetBytes(HashSize);

            // Combine salt and hash
            var hashBytes = new byte[saltSize + HashSize];
            Array.Copy(salt, 0, hashBytes, 0, saltSize);
            Array.Copy(hash, 0, hashBytes, saltSize, HashSize);

            // Convert to base64
            var base64Hash = Convert.ToBase64String(hashBytes);

            // Format hash with extra information
            return string.Format("$MAGIK$V1${0}${1}", iterations, base64Hash);
        }

        /// <summary>
        /// Creates a hash from a password with 10000 iterations
        /// </summary>
        /// <param name="password">The password</param>
        /// <returns>The hash</returns>
        public string Hash(string password)
        {
            return Hash(password, 10000);
        }

        /// <summary>
        /// Checks if hash is supported
        /// </summary>
        /// <param name="hashString">The hash</param>
        /// <returns>Is it hash supported</returns>
        public bool IsHashSupported(string hashString)
        {
            return hashString.Contains("$MAGIK$V1$");
        }

        /// <summary>
        /// Verifies a password against a hash
        /// </summary>
        /// <param name="password">The password</param>
        /// <param name="hashedPassword">The hash</param>
        /// <returns>Is it password verified</returns>
        public bool Verify(string password, string hashedPassword)
        {
            // Check hash
                                if   (!IsHashSupported(hashedPassword))   throw   new
NotSupportedException("The hashtype is not supported");

            // Extract iteration and Base64 string
```

```
                    var splittedHashString = hashedPassword.Replace("$MAGIK$V1$",
"").Split('$');
            var iterations = int.Parse(splittedHashString[0]);
            var base64Hash = splittedHashString[1];

            // Get hash bytes
            var hashBytes = Convert.FromBase64String(base64Hash);

            // Get salt
            var salt = new byte[saltSize];
            Array.Copy(hashBytes, 0, salt, 0, saltSize);

            // Create hash with given salt
            var pbkdf2 = new Rfc2898DeriveBytes(password, salt, iterations);
            byte[] hash = pbkdf2.GetBytes(HashSize);

            // Get result
            for (var i = 0; i < HashSize; i++)
            {
                if (hashBytes[i + saltSize] != hash[i])
                {
                    return false;
                }
            }
            return true;
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace Auth.UIModels
{
    public class AuthData
    {
        [EmailAddress]
        public string Email { get; set; } = null!;

        [MinLength(5)]
        public string Password { get; set; } = null!;
    }
}

namespace Auth.UIModels;

public class Token
{
    public string AccessToken { get; set; } = null!;
}

namespace Auth
{
    public class Program
    {
        public static void Main(string[] args)
```

```
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}

using Auth.Data;
using Auth.Data.Implementations;
using Auth.Data.Interfaces;
using Auth.Services;
using Common;

namespace Auth
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

         // This method gets called by the runtime. Use this method to add services
to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllers();

            services.AddCors(options =>
            {
                options.AddDefaultPolicy(builder =>
                {
                    builder.AllowAnyOrigin()
                    .AllowAnyMethod()
                    .AllowAnyHeader();
                });
            });

            services.AddSingleton<PasswordHasherService>();
                                    services.AddScoped<IAccountRepository>(f   =>   new
MongoAccountRepository(Configuration.GetConnectionString("Default")));

            var authOptionsConfiguration = Configuration.GetSection("Auth");
            services.Configure<AuthOptions>(authOptionsConfiguration);

            services.AddScoped<AuthService>();
        }

        // This method gets called by the runtime. Use this method to configure the
HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
```

```
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();

            app.UseCors();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Листинг основных классов ресурсного сервера

```
using Microsoft.EntityFrameworkCore;
using Resource.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Resource.Data
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions opts) : base(opts){}

        public DbSet<AccountAttachment> AccountAttachments { get; set; } = null!;
        public DbSet<AttachmentType> AttachmentTypes { get; set; } = null!;
        public DbSet<Profile> Profiles { get; set; } = null!;
        public DbSet<Project> Projects { get; set; } = null!;
        public DbSet<Field> Fields { get; set; } = null!;
        public DbSet<Stage> Stages { get; set; } = null!;
        public DbSet<StageAttachment> StagesAttachments { get; set; } = null!;
    }
}

namespace Resource.Exceptions;

public class ApplicationException : Exception {
    public ApplicationException(string? message) : base(message)
    {}
}

using AutoMapper;
using Resource.UIModels;

namespace Resource.MapperProfiles;

public class AttachmentMapperProfile : Profile {
    public AttachmentMapperProfile()
    {
        CreateMap<Models.AccountAttachment, AttachmentUI>()
            .ForMember(
                dest => dest.Id,
                opt => opt.MapFrom(src => src.Id)
            )
            .ForMember(
                dest => dest.Name,
                opt => opt.MapFrom(src => src.Name)
            )
            .ForMember(
                dest => dest.AttachmentTypeId,
                opt => opt.MapFrom(src => src.AttachmentTypeId)
            )
            .ForMember(
                dest => dest.Data,
                opt => opt.MapFrom(src => src.Data)
```

```
            );
    }
}

using Resource.Models;

namespace Resource.Data.Interfaces
{
    public interface IProjectsRepository {
        public Task<IEnumerable<Project>> GetAsync(int fieldId);
        public Task<Project?> FirstOrDefaultAsync(int projectId);
        public Task CreateAsync(Project project);
        public Task UpdateAsync(Project project);
        public Task DeleteAsync(Project project);
    }
}

namespace Resource.Data.Interfaces;

public interface IUnitOfWork {
    public IProfilesRepository Profiles { get; }
    public IFieldsRepository Fields { get; }
    public IProjectsRepository Projects { get; }
    public IStagesRepository Stages { get; }
    public IAccountAttachmentsRepository AccountAttachments{ get; }
    public IStagesAttachmentsRepository StagesAttachments { get; }
}

using Microsoft.EntityFrameworkCore;
using Resource.Data.Interfaces;
using Resource.Models;

namespace Resource.Data.MSImplementations;

public class MSProjectsRepository : IProjectsRepository
{
    private readonly AppDbContext context;

    public MSProjectsRepository(AppDbContext context)
    {
        this.context = context;
    }
    public async Task CreateAsync(Project project)
    {
        await context.Projects.AddAsync(project);
        await context.SaveChangesAsync();
    }

    public async Task DeleteAsync(Project project)
    {
        context.Projects.Remove(project);
        await context.SaveChangesAsync();
    }

    public async Task<Project?> FirstOrDefaultAsync(int projectId)
    {
        return await context.Projects.Where(p => p.Id == projectId)
            .Include(p => p.Stages)
            .FirstOrDefaultAsync(p => p.Id == projectId);
```

```
    }

    public async Task<IEnumerable<Project>> GetAsync(int fieldId)
    {
        return await context.Projects.Where(p => p.FieldId == fieldId)
            .Include(p => p.Stages)
            .ToListAsync();
    }

    public async Task UpdateAsync(Project project)
    {
        context.Projects.Update(project);
        await context.SaveChangesAsync();
    }
}

using Resource.Data.Interfaces;

namespace Resource.Data.MSImplementations;

public class MSUnitOfWork : IUnitOfWork {
    private readonly AppDbContext context;

    public MSUnitOfWork(AppDbContext context)
    {
        this.context = context;
    }

    private MSProfilesRepository? profiles;
              public    IProfilesRepository    Profiles    =>    profiles    ??=    new
MSProfilesRepository(context);
    private MSFieldsRepository? fields;
    public IFieldsRepository Fields => fields ??= new MSFieldsRepository(context);
    private MSProjectsRepository? projects;
              public    IProjectsRepository    Projects    =>    projects    ??=    new
MSProjectsRepository(context);
    private MSStagesRepository? stages;
    public IStagesRepository Stages => stages ??= new MSStagesRepository(context);
    private MSAccountAttachmentsRepository? accAttachments;
     public IAccountAttachmentsRepository AccountAttachments => accAttachments ??=
new MSAccountAttachmentsRepository(context);
    private MSStagesAttachmentsRepository? stagesAttachments;
     public IStagesAttachmentsRepository StagesAttachments => stagesAttachments ??=
new MSStagesAttachmentsRepository(context);
}

using System.Security.Claims;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Resource.Services;
using Resource.UIModels;

namespace Resource.Controllers;

[Route("api/[controller]")]
[ApiController]
[Authorize]
public class ProjectsController : ControllerBase {
    private readonly ProjectsService projectsService;
```

```
    public ProjectsController(ProjectsService projectsService)
    {
        this.projectsService = projectsService;
    }

    [HttpGet("{fieldId}")]
    [Route("")]
    public async Task<IActionResult> Get(int fieldId) {
                        var   accountId   =   User.Claims.Single(c   =>   c.Type   ==
ClaimTypes.NameIdentifier).Value;
        try {
            return Ok(await projectsService.GetProjectsAsync(accountId, fieldId));
        }
        catch(ApplicationException exc) {
            return BadRequest(exc.Message);
        }
    }

    [HttpGet("single/{projectId}")]
    [Route("")]
    public async Task<IActionResult> GetSingle(int projectId) {
                        var   accountId   =   User.Claims.Single(c   =>   c.Type   ==
ClaimTypes.NameIdentifier).Value;
        try {
            return Ok(await projectsService.GetProjectAsync(accountId, projectId));
        }
        catch(ApplicationException exc) {
            return BadRequest(exc.Message);
        }
    }

    [HttpPost("{fieldId}")]
    [Route("")]
       public  async  Task<IActionResult>  Create(int  fieldId,  [FromBody]ProjectUI
project) {
        if(ModelState.IsValid) {
                            var  accountId  =  User.Claims.Single(c  =>  c.Type  ==
ClaimTypes.NameIdentifier).Value;
            try {
                        await  projectsService.CreateProjectAsync(accountId,  fieldId,
project);
                return Ok(project);
            }
            catch(ApplicationException exc) {
                return BadRequest(exc.Message);
            }
        }
        else {
            return BadRequest();
        }
    }

    [HttpPut("{projectId}")]
    [Route("")]
      public  async  Task<IActionResult>  Update(int  projectId,  [FromBody]ProjectUI
project) {
        if(projectId != project.Id) return BadRequest();
        if(ModelState.IsValid) {
```

```
                        var accountId = User.Claims.Single(c => c.Type ==
ClaimTypes.NameIdentifier).Value;
            try {
                await projectsService.UpdateProjectAsync(accountId, project);
                return Ok();
            }
            catch(ApplicationException exc) {
                return BadRequest(exc.Message);
            }
        }
        else {
            return BadRequest();
        }
    }

    [HttpDelete("{id}")]
    [Route("")]
    public async Task<IActionResult> Delete(int id) {
        if(ModelState.IsValid) {
                        var accountId = User.Claims.Single(c => c.Type ==
ClaimTypes.NameIdentifier).Value;
            try {
                await projectsService.DeleteProjectAsync(accountId, id);
                return Ok();
            }
            catch(ApplicationException exc) {
                return BadRequest(exc.Message);
            }
        }
        else {
            return BadRequest();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace Resource.Models
{
    public class Project
    {
        public int Id { get; set; }

        public int FieldId { get; set; }

        [StringLength(50)]
        public string Name { get; set; } = null!;

        [StringLength(2048)]
        public string Description { get; set; } = null!;

        public Field? Field { get; set; }

        public IEnumerable<Stage>? Stages { get; set; }
```

```
    }
}

using AutoMapper;
using Resource.Data.Interfaces;
using Resource.Models;
using Resource.Tools;
using Resource.UIModels;

namespace Resource.Services;

public class ProjectsService {
    private readonly IUnitOfWork uof;
    private readonly UserAccessValidator accessValidator;
    private readonly IMapper mapper;

      public ProjectsService(IUnitOfWork uof, UserAccessValidator accessValidator,
IMapper mapper)
    {
        this.uof = uof;
        this.accessValidator = accessValidator;
        this.mapper = mapper;
    }

     public async Task CreateProjectAsync(string accountId, int fieldId, ProjectUI
project) {
        await accessValidator.ValidateAndGetFieldAsync(accountId, fieldId);
        project.Color = ColorEvaluator.DEFAULT_COLOR;
        Project newProject = new Project {
            FieldId = fieldId,
            Name = project.Name,
            Description = project.Description
        };
        await uof.Projects.CreateAsync(newProject);
        project.Id = newProject.Id;
    }

      public async Task<IEnumerable<ProjectUI>> GetProjectsAsync(string accountId,
int fieldId) {
        await accessValidator.ValidateAndGetFieldAsync(accountId, fieldId);
                                  return    mapper.Map<IEnumerable<ProjectUI>>(await
uof.Projects.GetAsync(fieldId));
    }

    public async Task<ProjectUI> GetProjectAsync(string accountId, int projectId) {
                                            return     mapper.Map<ProjectUI>(await
accessValidator.ValidateAndGetProjectAsync(accountId, projectId));
    }

    public async Task UpdateProjectAsync(string accountId, ProjectUI project) {
                                           var     projectToEdit     =     await
accessValidator.ValidateAndGetProjectAsync(accountId, project.Id);
        projectToEdit.Name = project.Name;
        projectToEdit.Description = project.Description;
        await uof.Projects.UpdateAsync(projectToEdit);
    }

    public async Task DeleteProjectAsync(string accountId, int projectId) {
```

```
         var project = await accessValidator.ValidateAndGetProjectAsync(accountId,
projectId);
        await uof.Projects.DeleteAsync(project);
    }
}

using Resource.Models;

namespace Resource.Tools;

public static class ColorEvaluator {
    public static string DEFAULT_COLOR = "#23a5d588";

    public static string GetProjectColor(Project project) {
             if(project.Stages == null || project.Stages.Count() == 0) return
DEFAULT_COLOR;
         var unfinishedStagesDifferences = project.Stages.Where(s => s.Progress !=
100).Select(stage => GetActExpDifference(stage));
                          if(unfinishedStagesDifferences.Count()   ==   0)   return
GetColorFromDifference(1);
        return GetColorFromDifference(unfinishedStagesDifferences.Average());
    }

    public static string GetStageColor(Stage stage) {
        return GetColorFromDifference(GetActExpDifference(stage));
    }

    private static string GetColorFromDifference(double difference) {
        //r = 255 0 0
        //g = 0 255 0
        //y = 255 255 0

        int r, g;
        double bas = 255 * 2 * (1 / (1 + (Math.Abs(difference - 0.5) * 2)));
        r = (int)(bas * (1 - difference));
        g = (int)(bas * difference);

        return $"rgba({r}, {g}, 75, 0.5)";
    }

    /// <summary>
    /// Return difference between expected progress and actual progress for project
stage
    /// </summary>
    /// <param name="stage">Project stage</param>
     /// <returns>Difference between expected progress and actual progress in range
from 0(You have done nothing and there is a deadline) to 1(You finished this
stage)</returns>
    private static double GetActExpDifference(Models.Stage stage) {
        if(stage.Progress == 100) return 1;
        double timeSpent = (DateTime.Now - stage.CreationDate).TotalSeconds;
        double totalTime = (stage.Deadline - stage.CreationDate).TotalSeconds;
        double expectedProgress = timeSpent / totalTime;
        if(expectedProgress > 1) expectedProgress = 1;
        double actualProgress = stage.Progress / 100.0;
         double difference = Math.Pow(((actualProgress - expectedProgress) + 1) / 2,
0.7); //we use power for biasing color to green side
        return difference;
    }
```

```
}

using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace Resource.Tools;

public class PictureConverter
{
    /// <summary>
    /// Create icon from image
    /// </summary>
    /// <param name="image">Binary image that should be converted</param>
    /// <param name="iconSize">Size of the resulting icon</param>
    /// <returns>Icon</returns>
    public byte[] CreateIconFromImage(byte[] image, int iconSize=128) {
        Bitmap bmp = new Bitmap(ByteToImage(image));
        bmp = ResizeImage(CropImage(bmp), iconSize, iconSize);
        return ImageToByte(bmp);
    }

    /// <summary>
    /// Restrict image height and width
    /// </summary>
    /// <param name="image">Binary image that should be restricted</param>
    /// <param name="maxWidth">New max width for image</param>
    /// <param name="maxHeight">New max height for image</param>
    /// <returns>Restricted image</returns>
     public byte[] RestrictImage(byte[] image, int maxWidth = 512, int maxHeight =
512) {
        Bitmap bmp = new Bitmap(ByteToImage(image));
        if(bmp.Width > maxWidth) {
            double ratio = (double)maxWidth / bmp.Width;
            int newHeight = (int)(bmp.Height * ratio);
            bmp = ResizeImage(bmp, maxWidth, newHeight);
        }
        if(bmp.Height > maxHeight) {
            double ratio = (double)maxHeight / bmp.Height;
            int newWidth = (int)(bmp.Width * ratio);
            bmp = ResizeImage(bmp, newWidth, maxHeight);
        }
        return ImageToByte(bmp);
    }

    /// <summary>
    /// Convert image to byte array
    /// </summary>
    /// <param name="img">Image</param>
    /// <returns>Byte array representing image</returns>
    public byte[] ImageToByte(Image img)
    {
        using var stream = new MemoryStream();
        img.Save(stream, ImageFormat.Png);
        return stream.ToArray();
    }

    /// <summary>
    /// Convert byte array to image
```

```
    /// </summary>
    /// <param name="img">Byte array representing image</param>
    /// <returns>Image</returns>
    public Image ByteToImage(byte[] img) {
        using var stream = new MemoryStream(img);
        return Image.FromStream(stream);
    }

    private Bitmap CropImage(Image orgImg)
    {
        var width = orgImg.Width;
        var height = orgImg.Height;
        var centerW = width / 2;
        var centerH = height / 2;

        Rectangle sRect = default;
        Rectangle destRect = default;

        if (width > height)
        {
            sRect = new Rectangle(centerW - centerH, 0, height, height);
            destRect = new Rectangle(0, 0, height, height);
        }
        else
        {
            sRect = new Rectangle(0, centerH - centerW, width, width);
            destRect = new Rectangle(0, 0, width, width);
        }

        var cropImage = new Bitmap(destRect.Width, destRect.Height);
        using (var graphics = Graphics.FromImage(cropImage))
        {
            graphics.DrawImage(orgImg, destRect, sRect, GraphicsUnit.Pixel);
        }
        return cropImage;
    }

    private Bitmap ResizeImage(Image image, int width, int height)
    {
        var destRect = new Rectangle(0, 0, width, height);
        var destImage = new Bitmap(width, height);

                                  destImage.SetResolution(image.HorizontalResolution,
image.VerticalResolution);

        using (var graphics = Graphics.FromImage(destImage))
        {
            graphics.CompositingMode = CompositingMode.SourceCopy;
            graphics.CompositingQuality = CompositingQuality.HighQuality;
            graphics.InterpolationMode = InterpolationMode.HighQualityBicubic;
            graphics.SmoothingMode = SmoothingMode.HighQuality;
            graphics.PixelOffsetMode = PixelOffsetMode.HighQuality;

            using (var wrapMode = new ImageAttributes())
            {
                wrapMode.SetWrapMode(WrapMode.TileFlipXY);
                graphics.DrawImage(image, destRect, 0, 0, image.Width, image.Height,
GraphicsUnit.Pixel, wrapMode);
            }
```

```
        }

        return destImage;
    }
}

using Microsoft.EntityFrameworkCore;
using Resource.Data;
using Resource.Data.Interfaces;

namespace Resource.Tools;

public class UserAccessValidator {
    private readonly IUnitOfWork uof;

    /// <summary>
    /// Validate access of user to data in DB
    /// </summary>
    /// <param name="uof">Data storage</param>
    public UserAccessValidator(IUnitOfWork uof)
    {
        this.uof = uof;
    }

    /// <summary>
    /// Validate if user owns field
    /// </summary>
    /// <param name="accountId">User accoutn ID</param>
    /// <param name="fieldId">Field ID</param>
    /// <returns>Field if user owner of that field</returns>
     public async Task<Models.Field> ValidateAndGetFieldAsync(string accountId, int
fieldId) {
        var field = await uof.Fields.FirstOrDefaultAsync(fieldId);
            if(field == null) throw new ApplicationException("Нет такой области
проектов");
            if(field.AccountId != accountId) throw new ApplicationException("Эта
область проектов принадлежит другому пользователю");
        return field;
    }

    /// <summary>
    /// Validate if user owns project
    /// </summary>
    /// <param name="accountId">User account ID</param>
    /// <param name="projectId">Project ID</param>
    /// <returns>Project if user owner of this project</returns>
     public async Task<Models.Project> ValidateAndGetProjectAsync(string accountId,
int projectId) {
        var project = await uof.Projects.FirstOrDefaultAsync(projectId);
        if(project == null) throw new ApplicationException("Нет такого проекта");
        await ValidateAndGetFieldAsync(accountId, project.FieldId);
        return project;
    }

     public async Task<Models.Stage> ValidateAndGetStageAsync(string accountId, int
stageId) {
        var stage = await uof.Stages.FirstOrDefaultAsync(stageId);
        if(stage == null) throw new ApplicationException("Нет такого этапа");
        await ValidateAndGetProjectAsync(accountId, stage.ProjectId);
```

```
        return stage;
    }

                            public       async       Task<Models.AccountAttachment>
ValidateAndGetAttachmentAsync(string accountId, int attachmentId) {
                                                 var      attachment      =      await
uof.AccountAttachments.FirstOrDefaultAsync(attachmentId);
             if(attachment  ==  null)  throw  new  ApplicationException("Нет  такого
вложения");
          if(attachment.AccountId != accountId) throw new ApplicationException("Это
вложение принадлежит другому пользователю");
        return attachment;
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace Resource.UIModels
{
    public class ProjectUI
    {
        public int Id { get; set; }

        [StringLength(50)]
        public string Name { get; set; } = null!;

        [StringLength(2048)]
        public string Description { get; set; } = null!;

        public string? Color { get; set; } = null!;
    }
}

namespace Resource
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}

using Common;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
```

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Resource.Data;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Resource.Services;
using Resource.Tools;
using Resource.Data.Interfaces;
using Resource.Data.MSImplementations;

namespace Resource
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

         // This method gets called by the runtime. Use this method to add services
to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllers();

            var authOptions = Configuration.GetSection("Auth").Get<AuthOptions>();

            services.AddCors(options =>
            {
                options.AddDefaultPolicy(builder =>
                {
                    builder.AllowAnyOrigin()
                        .AllowAnyMethod()
                        .AllowAnyHeader();
                });
            });

            services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
                .AddJwtBearer(options =>
                {
                    options.RequireHttpsMetadata = true;
                                        options.TokenValidationParameters  =  new
Microsoft.IdentityModel.Tokens.TokenValidationParameters
                    {
                        ValidateIssuer = true,
                        ValidIssuer = authOptions.Issuer,

                        ValidateAudience = true,
                        ValidAudience = authOptions.Audience,
```

```
                    ValidateLifetime = true,

                    IssuerSigningKey = authOptions.GetSymmetricSecurityKey(),
                    ValidateIssuerSigningKey = true
                };
            });

        services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());

        services.AddDbContext<AppDbContext>(opts =>
        {
            opts.UseSqlServer(Configuration.GetConnectionString("Default"));
        });

        services.AddTransient<PictureConverter>();

        services.AddScoped<IUnitOfWork, MSUnitOfWork>();
        services.AddScoped<UserAccessValidator>();

        services.AddScoped<ProfilesService>();
        services.AddScoped<FieldsService>();
        services.AddScoped<ProjectsService>();
        services.AddScoped<StagesService>();
        services.AddScoped<AttachmentsService>();
    }

    // This method gets called by the runtime. Use this method to configure the
HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.UseRouting();
        app.UseCors();

        app.UseAuthentication();
        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
}
```

ПРИЛОЖЕНИЕ В
(обязательное)
Листинг основных компонентов клиентского приложения

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import {FormsModule, ReactiveFormsModule} from "@angular/forms";
import { HeaderComponent } from './header/header.component';
import { MainComponent } from './main/main.component';
import { FooterComponent } from './footer/footer.component';
import { NavComponent } from './nav/nav.component';
import { HomeComponent } from './main/home/home.component';
import { ProfileComponent } from './main/profile/profile.component';
import { AuthComponent } from './auth/auth.component';
import {AUTH_API_URL, RESOURCE_API_URL, tokenGetter} from "./config/app-injection-
tokens";
import {environment} from "../environments/environment";
import {HttpClientModule} from "@angular/common/http";
import {JwtModule} from "@auth0/angular-jwt";
import { ProfileEditComponent } from './main/profile/profile-edit/profile-
edit.component';
import { ProfileFriendsComponent } from './main/profile/profile-friends/profile-
friends.component';
import { ProfileStatisticComponent } from './main/profile/profile-
statistic/profile-statistic.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FieldsComponent } from './main/projects-manager/fields/fields.component';
import { AddFieldComponent } from './main/projects-manager/fields/add-field/add-
field.component';
import { EditFieldComponent } from './main/projects-manager/fields/edit-field/edit-
field.component';
import { ProjectsManagerComponent } from './main/projects-manager/projects-
manager.component';
import { ProjectComponent } from './main/projects-
manager/project/project.component';
import { StagesComponent } from './main/projects-manager/stages/stages.component';
import { EditProjectComponent } from './main/projects-manager/project/edit-
project/edit-project.component';
import { AddStageComponent } from './main/projects-manager/project/add-stage/add-
stage.component';
import { EditStageComponent } from './main/projects-manager/stages/edit-stage/edit-
stage.component';
import { AttachmentsComponent } from './main/attachments/attachments.component';
import { EditAttachmentComponent } from './main/attachments/edit-attachment/edit-
attachment.component';
import { CreateAttachmentComponent } from './main/attachments/create-
attachment/create-attachment.component';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    MainComponent,
    FooterComponent,
    NavComponent,
    HomeComponent,
    ProjectsManagerComponent,
```

```
    ProfileComponent,
    AuthComponent,
    ProfileEditComponent,
    ProfileFriendsComponent,
    ProfileStatisticComponent,
    FieldsComponent,
    AddFieldComponent,
    EditFieldComponent,
    ProjectComponent,
    StagesComponent,
    EditProjectComponent,
    AddStageComponent,
    EditStageComponent,
    AttachmentsComponent,
    EditAttachmentComponent,
    CreateAttachmentComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule,
    ReactiveFormsModule,
    JwtModule.forRoot({
      config: {
        tokenGetter,
        allowedDomains: environment.allowedDomains
      }
    }),
    BrowserAnimationsModule
  ],
  providers: [
    {
      provide: AUTH_API_URL,
      useValue: environment.authApi
    },
    {
      provide: RESOURCE_API_URL,
      useValue: environment.resourceApi
    }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }

export class Auth {
  constructor(public email:string,
              public password:string) {
  }
}

export class Token {
  constructor(public accessToken:string) {
  }
}

export class Attachment {
    constructor(public name:string,
                public data:string,
                public attachmentTypeId:number,
```

```
                public id?:number) {}

    //attachment type id: 1-Link, 2-Table
}

export class Field {
    constructor(public id:number,
        public name:string,
        public icon:string) {}
}

export class Profile {
  constructor(public username: string,
              public description: string,
              public picture: string) {
  }
}

export class Project {
    constructor(public id: number|undefined,
        public name: string,
        public description: string,
        public color: string|undefined = undefined) {}

    //project type id: 1-Private, 2-Readonly, 3-Copyable, 4-Public
}

import { Attachment } from "./attachment";

export class Stage {
    constructor(public name:string,
                public description:string,
                public deadline:Date,
                public attachments:Attachment[],
                public creationDate?:Date,
                public id?:number,
                public progress?:number,
                public color?:string) {}
}

import { HttpClient } from '@angular/common/http';
import { Inject, Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { RESOURCE_API_URL } from 'src/app/config/app-injection-tokens';
import { Attachment } from 'src/models/resource/attachment';
import { Stage } from 'src/models/resource/stage';

@Injectable({
  providedIn: 'root'
})
export class AttachmentsService {

  constructor(@Inject(RESOURCE_API_URL)private url:string,
    private http:HttpClient) { }

    getAttachments(): Observable<Attachment[]> {
      return this.http.get<Attachment[]>(`${this.url}api/attachments`);
    }
```

```
    getAttachmentsForStage(stage: Stage): Observable<Attachment[]> {
                                                                        return
this.http.get<Attachment[]>(`${this.url}api/attachments/stage/${stage.id}`);
    }

    createAttachment(attach: Attachment): Observable<Attachment> {
      return this.http.post<Attachment>(`${this.url}api/attachments`, attach);
    }

    addAttachmentToStage(stage: Stage, attach: Attachment): Observable<any> {
                                                                        return
this.http.post<Attachment>(`${this.url}api/attachments/stage/${stage.id}`,
attach.id);
    }

    editAttachment(attach: Attachment): Observable<Attachment> {
        return  this.http.put<Attachment>(`${this.url}api/attachments/${attach.id}`,
attach);
    }

            deleteAttachmentFromStage(stage:   Stage,   attachment:   Attachment):
Observable<any> {
                                                                        return
this.http.delete<any>(`${this.url}api/attachments/stage/${stage.id}?attachmentId=${
attachment.id}`);
    }

    deleteAttachment(attach: Attachment): Observable<any> {
      return this.http.delete<any>(`${this.url}api/attachments/${attach.id}`);
    }
}

import {Inject, Injectable} from '@angular/core';
import {Auth} from "../../models/auth/auth";
import {Observable} from "rxjs";
import {tap} from "rxjs/operators";
import {Token} from "../../models/auth/token";
import {HttpClient} from "@angular/common/http";
import {
  AUTH_API_URL,
  tokenGetter,
  tokenKiller,
  tokenSetter
} from "../../app/config/app-injection-tokens";
import {JwtHelperService} from "@auth0/angular-jwt";

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  constructor(private http: HttpClient,
              @Inject(AUTH_API_URL) private url: string,
              private jwtHelper: JwtHelperService) {}

  public isAuthenticated(): boolean {
    let token = tokenGetter();
    // @ts-ignore
    return token && !this.jwtHelper.isTokenExpired(token);
  }
```

```
  public signIn(auth:Auth): Observable<Token> {
    return this.http.post<Token>(`${this.url}api/auth/signIn`, auth)
      .pipe(
        tap(token => {
          tokenSetter(token.accessToken);
        })
      )
  }

  public signUp(auth:Auth): Observable<Token> {
    return this.http.post<Token>(`${this.url}api/auth/signUp`, auth)
      .pipe(
        tap(token => {
          tokenSetter(token.accessToken);
        })
      )
  }

  public signOut() {
    tokenKiller();
  }
}

import { HttpClient } from '@angular/common/http';
import { Inject, Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { RESOURCE_API_URL } from 'src/app/config/app-injection-tokens';
import { Field } from 'src/models/resource/field';

@Injectable({
  providedIn: 'root'
})
export class FieldsService {

  constructor(@Inject(RESOURCE_API_URL)private url:string,
              private http:HttpClient) { }

  getFields():Observable<Field[]> {
    return this.http.get<Field[]>(`${this.url}api/fields`);
  }

  addField(field:Field):Observable<number> {
    return this.http.post<number>(`${this.url}api/fields`, field);
  }

  editField(field:Field):Observable<any> {
    return this.http.put(`${this.url}api/fields`, field);
  }

  deleteField(field:Field):Observable<any> {
    return this.http.delete(`${this.url}api/fields/${field.id}`);
  }
}

import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
```

```
})
export class ModalService {

  constructor() { }

  private currentModalName: string = "";
  private modalChild: string = "";

  public openModal(modalName:string) {
    this.currentModalName = modalName;
  }

  public isModalOpened(modalName:string) {
    return this.currentModalName == modalName;
  }

  public closeModal() {
    this.currentModalName = "";
  }

  public openChildModal(childName:string) {
    this.modalChild = childName;
  }

  public isChildOpened(childName:string) {
    return this.modalChild == childName;
  }

  public closeChild() {
    this.modalChild = "";
  }
}

import {Inject, Injectable} from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {RESOURCE_API_URL} from "../../app/config/app-injection-tokens";
import {Observable} from "rxjs";
import {Profile} from "../../models/resource/profile";

@Injectable({
  providedIn: 'root'
})
export class ProfilesService {
  constructor(private http: HttpClient,
              @Inject(RESOURCE_API_URL) private url: string) { }

  getProfile(): Observable<Profile> {
    return this.http.get<Profile>(`${this.url}api/profiles/`);
  }

  editProfile(profile: Profile, isPictureEdited: boolean): Observable<any> {
    if(!isPictureEdited) {
      profile = {...profile};
      profile.picture = "";
    }
    return this.http.put(`${this.url}api/profiles/`, profile);
  }

  validateProfile(profile?: Profile): string | undefined {
```

```
    if(!profile) return "Профиль пользователя должен присутствовать";
    if(profile.username.length > 50 ) return "Максимальная длина имени 50 символов";
     if(profile.description.length > 200 ) return "Максимальная длина описания 200
символов";
    return undefined;
  }
}

import { HttpClient } from '@angular/common/http';
import { Inject, Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { RESOURCE_API_URL } from 'src/app/config/app-injection-tokens';
import { Field } from 'src/models/resource/field';
import { Project } from 'src/models/resource/project';

@Injectable({
  providedIn: 'root'
})
export class ProjectsService {

  constructor(@Inject(RESOURCE_API_URL)private url:string,
    private http:HttpClient) { }

  getProjects(field: Field): Observable<Project[]> {
    return this.http.get<Project[]>(`${this.url}api/projects/${field.id}`);
  }

  getProject(projectId: number): Observable<Project> {
    return this.http.get<Project>(`${this.url}api/projects/single/${projectId}`);
  }

  addProject(field: Field, project:Project): Observable<Project> {
    return this.http.post<Project>(`${this.url}api/projects/${field.id}`, project);
  }

  editProject(project:Project):Observable<Project> {
    return this.http.put<Project>(`${this.url}api/projects/${project.id}`, project);
  }

  deleteProject(project: Project): Observable<any> {
    return this.http.delete<any>(`${this.url}api/projects/${project.id}`);
  }
}

import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class ProfileRoutingService {
  private currentWindow:string = "statistic";

  constructor() {}

  default() {
    this.currentWindow = "statistic";
  }

  routeToEdit() {
```

```
    this.currentWindow = "edit";
  }

  routeToFriends() {
    this.currentWindow = "friends";
  }

  routeToStatistic() {
    this.currentWindow = "statistic";
  }

  isEdit():boolean {
    return this.currentWindow == "edit";
  }

  isFriends():boolean {
    return this.currentWindow == "friends";
  }

  isStatistic():boolean {
    return this.currentWindow == "statistic";
  }
}

import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class AppRoutingService {
  private currentPage:string = "projects";

  constructor() {}

  routeToHomepage() {
    this.currentPage = "homepage";
  }

  routeToProjects() {
    this.currentPage = "projects";
  }

  routeToProfile() {
    this.currentPage = "profile";
  }

  isHomepage():boolean {
    return this.currentPage == "homepage";
  }

  isProjects():boolean {
    return this.currentPage == "projects";
  }

  isProfile():boolean {
    return this.currentPage == "profile";
  }
}
```

```
import { HttpClient } from '@angular/common/http';
import { Inject, Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { RESOURCE_API_URL } from 'src/app/config/app-injection-tokens';
import { Project } from 'src/models/resource/project';
import { Stage } from 'src/models/resource/stage';

@Injectable({
  providedIn: 'root'
})
export class StagesService {

  constructor(@Inject(RESOURCE_API_URL)private url:string,
              private http:HttpClient) { }

  getStages(project:Project): Observable<Stage[]> {
    return this.http.get<Stage[]>(`${this.url}api/stages/${project.id}`);
  }

  addStage(project:Project, stage:Stage): Observable<Stage> {
    return this.http.post<Stage>(`${this.url}api/stages/${project.id}`, stage);
  }

  editStage(stage:Stage):Observable<Stage> {
    return this.http.put<Stage>(`${this.url}api/stages/${stage.id}`, stage);
  }

  deleteStage(stage:Stage): Observable<any> {
    return this.http.delete<any>(`${this.url}api/stages/${stage.id}`);
  }
}

// This file can be replaced during build by using the `fileReplacements` array.
// `ng build` replaces `environment.ts` with `environment.prod.ts`.
// The list of file replacements can be found in `angular.json`.

export const environment = {
  production: false,
  authApi: 'http://192.168.100.77:7001/',
  resourceApi: 'http://192.168.100.77:7002/',
  allowedDomains: ['192.168.100.77:7001', "192.168.100.77:7002"]
};

/*
 * For easier debugging in development mode, you can import the following file
   *   to   ignore   zone   related   error   stack   frames   such   as   `zone.run`,
`zoneDelegate.invokeTask`.
 *
 * This import should be commented out in production mode because it will have a
negative impact
 * on performance if an error is thrown.
 */
// import 'zone.js/plugins/zone-error';  // Included with Angular CLI.

import { Component, OnInit } from '@angular/core';
import {FormBuilder, FormControl, FormGroup, Validators} from "@angular/forms";
import {AuthService} from "../../services/auth/auth.service";
import {Auth} from "../../models/auth/auth";
```

```
@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html',
  styleUrls: ['./auth.component.css']
})
export class AuthComponent implements OnInit {
  public authGroup: FormGroup = new FormGroup({
    email: new FormControl('', [Validators.email, Validators.required]),
    password: new FormControl('', [Validators.minLength(5), Validators.required])
  });
  public error = "";

  constructor(private authService: AuthService) {}

  ngOnInit(): void {
  }

  signIn() {
    if(this.authGroup.valid) {
                    this.authService.signIn(new   Auth(this.authGroup.value.email,
this.authGroup.value.password))
        .subscribe(res => {

        }, err => {
          this.error = err.error;
        });
    }
  }

  signUp() {
    if(this.authGroup.valid) {
                    this.authService.signUp(new   Auth(this.authGroup.value.email,
this.authGroup.value.password))
        .subscribe(res => {

        }, err => {
          this.error = err.error;
        });
    }
  }
}

import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';
import { Observable } from 'rxjs';
import { Attachment } from 'src/models/resource/attachment';
import { Stage } from 'src/models/resource/stage';
import { AttachmentsService } from 'src/services/attachment/attachments.service';
import { ModalService } from 'src/services/modal/modal.service';

@Component({
  selector: 'app-attachments',
  templateUrl: './attachments.component.html',
  styleUrls: ['./attachments.component.css']
})
export class AttachmentsComponent implements OnInit {

  @Input() public stage?:Stage;
       @Output()   public   attachmentAdded:   EventEmitter<Attachment>   =   new
EventEmitter<Attachment>();
```

```
      @Output()  public  attachmentEdited:  EventEmitter<Attachment>  =  new
EventEmitter<Attachment>();
      @Output()  public  attachmentDeleted:  EventEmitter<Attachment>  =  new
EventEmitter<Attachment>();
  public accountAttachments?: Attachment[];
  public attachToEdit?: Attachment;

  constructor(public modalService: ModalService,
              public attachmentsService: AttachmentsService) { }

  ngOnInit(): void {
    this.getAttachments();
  }

  getAttachments() {
    this.attachmentsService.getAttachments()
      .subscribe(res => {
        this.accountAttachments = res;
      }, err => {

      })
  }

  createAttachmentModal() {
    this.modalService.openChildModal("create-attachment");
  }

  createAttachment(attach:Attachment) {
    this.attachmentsService.createAttachment(attach)
      .subscribe(res => {
        this.accountAttachments?.push(res);
      }, err => {

      })
  }

  editAttachmentModal(attach: Attachment) {
    this.attachToEdit = attach;
    this.modalService.openChildModal("edit-attachment");
  }

  editAttachment(attach:Attachment) {
    this.attachmentsService.editAttachment(attach)
    .subscribe(res => {
      this.attachmentEdited.emit(res);
      let i = this.accountAttachments?.indexOf(this.attachToEdit!);
      this.accountAttachments![i!] = res;
    }, err => {

    });
  }

  deleteAttachment(attach: Attachment) {
    if(confirm("Вы точно хотите удалить это вложение?")) {
      this.attachmentsService.deleteAttachment(attach)
        .subscribe(res => {
          this.attachmentDeleted.emit(attach);
            this.accountAttachments = this.accountAttachments!.filter(a => a.id !=
attach.id);
```

```
        }, err => {

        });
    }
  }

  addAttachmentToStage(attach: Attachment) {
    this.attachmentsService.addAttachmentToStage(this.stage!, attach)
      .subscribe(res => {
        this.attachmentAdded.emit(attach);
      }, err => {
        alert("Вложение уже находится в этой стадии");
      });
  }
}

import { Component, OnInit } from '@angular/core';
import {ProfilesService} from "../../../services/profile/profiles.service";
import {Profile} from "../../../models/resource/profile";
import {DomSanitizer, SafeResourceUrl} from "@angular/platform-browser";
import    {    ProfileRoutingService    }    from    'src/services/routing/profile-
routing.service';

@Component({
  selector: 'app-profile',
  templateUrl: './profile.component.html',
  styleUrls: ['./profile.component.css']
})
export class ProfileComponent implements OnInit {
  currentProfile?: Profile;
  currentProfileImageUrl?: SafeResourceUrl;

  constructor(private profileService: ProfilesService,
              private sanitizer: DomSanitizer,
              public profileRoutingService: ProfileRoutingService) { }

  ngOnInit(): void {
    this.getProfile();
  }

  getProfile() {
    this.profileService.getProfile()
      .subscribe(res => {
        this.currentProfile = res;
        this.getPictureSource();
      }, err => {
        console.log(err)
      });
  }

  getPictureSource() {
    this.currentProfileImageUrl = this.sanitizer.bypassSecurityTrustResourceUrl(
      ` ${this.currentProfile?.picture}`);
  }
}

import { Component, OnInit, Output, ViewChild } from '@angular/core';
import { Project } from 'src/models/resource/project';
import { Stage } from 'src/models/resource/stage';
```

```
import { FieldsService } from 'src/services/field/fields.service';
import { ProjectsService } from 'src/services/project/projects.service';
import { FieldsComponent } from './fields/fields.component';
import { StagesComponent } from './stages/stages.component';

@Component({
  selector: 'app-projects-manager',
  templateUrl: './projects-manager.component.html',
  styleUrls: ['./projects-manager.component.css']
})
export class ProjectsManagerComponent implements OnInit {

  @ViewChild(FieldsComponent)private fields?:FieldsComponent;
  @ViewChild(StagesComponent)private stages?:StagesComponent;

  public currentProject?:Project;

  constructor(private projectsService: ProjectsService) { }

  ngOnInit(): void {
  }

  onCurrentProjectChanged(project:Project) {
    this.currentProject = project;
  }

  onCurrentProjectDeleted(item:any) {
    this.fields?.removeProjectFromList(this.currentProject!);
    this.currentProject = undefined;
  }

  onStageAdded(stage:Stage) {
    this.stages?.addStageToList(stage);
    this.fields?.getCurrentFieldProjects();
    this.reloadProject();
  }

  onStageEditedOrDeleted() {
    this.reloadProject();
  }

  reloadProject() {
    this.projectsService.getProject(this.currentProject?.id!)
      .subscribe(res => {
        this.currentProject!.name = res.name;
        this.currentProject!.description = res.description;
        this.currentProject!.color = res.color;
      }, err => {

      })
  }
}

import { Component, EventEmitter, Input, OnDestroy, OnInit, Output } from
'@angular/core';
import { DomSanitizer, SafeResourceUrl } from '@angular/platform-browser';
import { Field } from 'src/models/resource/field';
import { ModalService } from 'src/services/modal/modal.service';
import { FieldsService } from 'src/services/field/fields.service';
```

```
import { ProjectsService } from 'src/services/project/projects.service';
import { Project } from 'src/models/resource/project';

@Component({
  selector: 'app-fields',
  templateUrl: './fields.component.html',
  styleUrls: ['./fields.component.css']
})
export class FieldsComponent implements OnInit {
  public defaultProjectColor = "#23a5d588";

  public fields?:Field[];

  public currentField?: Field;

  public currentFieldProjects: Project[] = new Array<Project>();

      @Output()private   currentProjectChanged:   EventEmitter<Project>   =   new
EventEmitter<Project>();

  @Input()public currentProject?: Project;

  public isFieldsShown: boolean = true;

  constructor(private sanitizer: DomSanitizer,
    public modalService: ModalService,
    private fieldsService: FieldsService,
    private projectsService: ProjectsService) { }

  ngOnInit(): void {
    this.getFields();
  }

  selectField(field: Field) {
    this.currentField = field;
    this.getCurrentFieldProjects();
    this.hideFields();
  }

  showFields() {
    this.isFieldsShown = true;
  }

  hideFields() {
    this.isFieldsShown = false;
  }

  getUrlFromIcon(icon:string) {
    if(icon == "") return "assets/puzzle.jpg";
    return  this.sanitizer.bypassSecurityTrustResourceUrl(
        ` ${icon}`);
  }

  getFields() {
    this.fieldsService.getFields()
      .subscribe(res => {
        this.fields = res;
      },err => {
        alert("Не удалось получить данные с сервера");
```

```
      console.log(err);
    });
}

addFieldModal() {
  this.modalService.openModal('add-field');
}

addField(field: Field) {
  this.fieldsService.addField(field)
    .subscribe(res => {
      field.id = res;
      this.fields!.push(field);
    }, err => {
      console.log(err);
      alert("Добавление не удалось");
    });
}

editFieldModal() {
  this.modalService.openModal('edit-field');
}

editField(field: Field) {
  this.fieldsService.editField(field)
    .subscribe(res => {

    }, err => {
      console.log(err);
      alert("Изменение не удалось");
    });
}

deleteCurrentField() {
  if(confirm("Вы уверены что хотите удалить эту область проектов?")) {
    this.fieldsService.deleteField(this.currentField!)
      .subscribe(res => {
        this.fields = this.fields?.filter(f => f !== this.currentField!);
        this.currentField = undefined;
        this.showFields();
      }, err => {
        console.log(err);
        alert("Удаление не удалось");
      });
  }
}

getCurrentFieldProjects() {
  this.currentFieldProjects = new Array<Project>();
  this.projectsService.getProjects(this.currentField!)
    .subscribe(res => {
      this.currentFieldProjects = res;
    }, err => {
      console.log(err);
      alert("Получение проектов не удалось");
    })
}

addNewProject() {
```

```
      let newProject = new Project(undefined, "Новый проект", "Мой новый проект",
undefined);
    this.projectsService.addProject(this.currentField!, newProject)
      .subscribe(res => {
        newProject = res;
        this.currentFieldProjects.push(newProject);
      }, err => {
        console.log(err);
        alert("Не удалось добавить проект");
      })
  }

  changeProject(project: Project) {
    this.currentProject = project;
    this.currentProjectChanged.emit(this.currentProject);
  }

  removeProjectFromList(project:Project) {
       this.currentFieldProjects = this.currentFieldProjects.filter(p => p.id !=
project.id);
  }
}

import { Component, ElementRef, EventEmitter, Input, OnChanges, OnInit, Output,
SimpleChanges, ViewChild } from '@angular/core';
import { Project } from 'src/models/resource/project';
import { Stage } from 'src/models/resource/stage';
import { ModalService } from 'src/services/modal/modal.service';
import { ProjectsService } from 'src/services/project/projects.service';
import { StagesService } from 'src/services/stage/stages.service';

@Component({
  selector: 'app-project',
  templateUrl: './project.component.html',
  styleUrls: ['./project.component.css']
})
export class ProjectComponent implements OnInit, OnChanges {

  @Input() public currentProject?: Project;
       @Output()    public    currentProjectDeleted:    EventEmitter<any>    =    new
EventEmitter<any>();
  @Output() public stageAdded: EventEmitter<Stage> = new EventEmitter<Stage>();

  @ViewChild("projectHeader") public projectHeader?: ElementRef;

  public descriptionShown: boolean = false;

  constructor(private projectsService: ProjectsService,
              private stagesService: StagesService,
              public modalService: ModalService) { }

  ngOnChanges(changes: SimpleChanges): void {
    this.reloadBackground();
  }

  ngOnInit(): void {
  }

  addStageModal() {
```

```
    this.modalService.openModal('add-stage');
  }

  addStage(stage: Stage) {
    this.stagesService.addStage(this.currentProject!, stage)
      .subscribe(res => {
        this.stageAdded.emit(res);
      }, err => {
        console.log(err);
        alert("Изменение не удалось");
      });
  }

  editCurrentProjectModal() {
    this.modalService.openModal('edit-project');
  }

  editProject(project: Project) {
    this.projectsService.editProject(project)
      .subscribe(res => {

      }, err => {
        console.log(err);
        alert("Изменение не удалось");
      });
  }

  deleteCurrentProject() {
    if(confirm("Вы уверены что хотите удалить этот проект?")) {
      this.projectsService.deleteProject(this.currentProject!)
        .subscribe(res => {
          this.currentProjectDeleted.emit();
        }, err => {
          console.log(err);
          alert("Удаление не удалось");
        });
    }
  }

  reloadBackground() {
    console.log("ok");
                              this.projectHeader?.nativeElement.setAttribute('style',
`background:${this.currentProject?.color}`);
  }
}

import { Component, EventEmitter, Input, OnChanges, OnInit, Output, SimpleChanges }
from '@angular/core';
import { Attachment } from 'src/models/resource/attachment';
import { Project } from 'src/models/resource/project';
import { Stage } from 'src/models/resource/stage';
import { AttachmentsService } from 'src/services/attachment/attachments.service';
import { ModalService } from 'src/services/modal/modal.service';
import { StagesService } from 'src/services/stage/stages.service';

@Component({
  selector: 'app-stages',
  templateUrl: './stages.component.html',
  styleUrls: ['./stages.component.css']
```

```
})
export class StagesComponent implements OnInit, OnChanges {
  @Input() public currentProject?: Project;
  @Output() public stageEditedOrDeleted: EventEmitter<any> = new EventEmitter<any>()
  public stages?:Stage[];
  public stageToEdit?:Stage;
  public stageToAddAttachment?:Stage;

  public openedStages: Stage[] = new Array<Stage>();

  constructor(private stagesService: StagesService,
              private attachmentsService: AttachmentsService,
              public modalService: ModalService) { }

  ngOnChanges(changes: SimpleChanges): void {
    if(changes["currentProject"]) this.getStages();
  }

  ngOnInit(): void {
    this.getStages();
  }

  isStageOpened(stage:Stage) {
    return this.openedStages.find(s => s.id == stage.id) != undefined;
  }

  openStage(stage:Stage) {
    this.openedStages.push(stage);
  }

  closeStage(stage:Stage) {
    this.openedStages = this.openedStages.filter(s => s.id != stage.id);
  }

  getStages() {
    this.stagesService.getStages(this.currentProject!)
      .subscribe(res => {
        this.stages = res;
      }, err => {
        console.log(err);
        alert(err);
      });
  }

  addStageToList(stage:Stage) {
    this.stages?.push(stage);
  }

  editStageModal(stage:Stage) {
    this.stageToEdit = stage;
    this.modalService.openModal('edit-stage');
  }

  editStage(stage: Stage) {
    this.stagesService.editStage(stage)
      .subscribe(res => {
        let i = this.stages?.findIndex(s => s.id == stage.id)!;
        this.stages![i] = res;
        this.stageEditedOrDeleted.emit();
```

```
      }, err => {
        console.log(err);
        alert("Изменение не удалось");
      });
  }

  deleteStage(stage:Stage) {
    if(confirm("Вы уверены что хотите удалить эту стадию?")) {
      this.stagesService.deleteStage(stage)
        .subscribe(res => {
          this.stages = this.stages?.filter(s => s.id != stage.id);
          this.stageEditedOrDeleted.emit();
        }, err => {
          console.log(err);
          alert("Удаление не удалось");
        });
    }
  }

  addAttachmentModal(stage:Stage) {
    this.stageToAddAttachment = stage;
    this.modalService.openModal("attachments");
  }

  getTableFromAttachment(attach: Attachment):string[][] {
    return JSON.parse(attach.data);
  }

  addAttachmentToList(attach:Attachment) {
    this.stageToAddAttachment?.attachments.push(attach);
  }

  editAttachments(attach:Attachment) {
    for(let stage of this.stages!) {
      let current = stage.attachments.filter(a => a.id == attach.id)[0];
      if(current) {
        current.name = attach.name;
        current.data = attach.data;
      }
    }
  }

  deleteAttachmentFromStages(attach:Attachment) {
    for(let stage of this.stages!) {
      stage.attachments = stage.attachments.filter(a => a.id != attach.id);
    }
  }

  deleteAttachmentFromStage(stage:Stage, attach:Attachment) {
    this.attachmentsService.deleteAttachmentFromStage(stage, attach)
      .subscribe(res => {
        stage.attachments = stage.attachments.filter(a => a.id != attach.id);
      }, err => {

      })
  }
}
```

ПРИЛОЖЕНИЕ Г
(обязательное)
Охрана труда на рабочем месте инженера программиста

Охрана труда – система законодательных актов, социально-экономических, организационных, технических, гигиенических и лечебно-профилактических мероприятий и средств, обеспечивающих безопасность, сохранение здоровья и работоспособности человека в процессе труда. Научно-технический прогресс внёс серьёзные изменения в условия производственной деятельности работников умственного труда. Их труд стал более интенсивным, напряжённым, требующим значительных затрат умственной, эмоциональной и физической энергии.

Вся деятельность в области охраны труда регламентирована действующим законодательством Республики Беларусь, санитарными нормами и правилами, гигиеническими нормативами, предписаниями надзорных органов.

К комплексу мероприятий в области охраны труда относятся:

– проведение производственного лабораторного контроля за условиями труда на рабочих местах;

– разработка инструкций по охране труда и ознакомление с ними персонала;

– модернизация рабочих мест и технологического оборудования;

– создание безопасных условий труда.

Рабочее место – это часть пространства, в котором инженер осуществляет трудовую деятельность, и проводит большую часть рабочего времени. Рабочее место, хорошо приспособленное к трудовой деятельности инженера, правильно и целесообразно организованное, в отношении пространства, формы, размера обеспечивает ему удобное положение при работе и высокую производительность труда при наименьшем физическом и психическом напряжении.

При правильной организации рабочего места производительность труда инженера возрастает на значение от восьми до 20 процентов.

При организации рабочего места программиста на предприятии были соблюдены следующие основные условия:

– оптимальное размещение оборудования, входящего в состав рабочего места;

– достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;

– хорошее естественное и искусственное освещение для выполнения поставленных задач;

– уровень акустического шума не превышал допустимого значения.

Главными элементами рабочего места программиста являются письменный стол и кресло. Основным рабочим положением является положение сидя.

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает чёткий порядок и

постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне лёгкой досягаемости рабочего пространства.

Моторное поле – пространство рабочего места, в котором могут осуществляться двигательные действия человека.

Максимальная зона досягаемости рук – это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

Важным моментом является также рациональное размещение на рабочем месте документации, канцелярских принадлежностей, что должно обеспечить работающему удобную рабочую позу, наиболее экономичные движения и минимальные траектории перемещения, работающего и предмета труда на данном рабочем месте.

При разработке оптимальных условий труда программиста необходимо учитывать освещённость, шум и микроклимат.

Помещения для работы программиста должны иметь естественное и искусственное освещение.

Площадь на одно рабочее место с видео-дисплейным терминалом (ВДТ) и ПЭВМ для взрослых пользователей должна составлять не менее 6,0 $м^2$, а объем не менее 20,0 $м^3$.

Искусственное освещение в помещениях эксплуатации ВДТ и ПЭВМ должно осуществляться системой общего равномерного освещения. В административно-общественных помещениях, в случаях преимущественной работы с документами, допускается применение системы комбинированного освещения.

В качестве источников света при искусственном освещении должны применяться преимущественно люминесцентные лампы.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учётом его количества и конструктивных особенностей (размер ВДТ и ПЭВМ, клавиатуры и др.), характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ВДТ и ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления.

Тип рабочего стула (кресла) должен выбираться в зависимости от характера и продолжительности работы.

Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надёжную фиксацию.

Экран видеомонитора должен находиться от глаз пользователя на оптимальном расстоянии 600-700 мм, но не ближе 500 мм с учётом размеров алфавитно-цифровых знаков и символов.

Высота рабочей поверхности стола должна регулироваться в пределах 680-800 мм, при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии не менее чем 300 мм от края, обращённого к пользователю или на специальной, регулируемой по высоте рабочей поверхности, отделённой от основной столешницы.

Помимо требований к организации рабочего места СанПиН 9-131 РБ 2000 устанавливает требования к микроклимату рабочей зоны: влажности, температуре, скорости потока воздуха и пр. Основной принцип нормирования микроклимата – создание оптимальных условий для теплообмена тела человека с окружающей средой.

К работе с ПК допускаются работники, не имеющие медицинских противопоказаний, прошедшие инструктаж по вопросам охраны труда, с группой по электробезопасности не ниже I.

Женщины со времени установления беременности и в период кормления грудью к выполнению всех видов работ, связанных с использованием ПК, не допускаются.

Работники должны соблюдать правила внутреннего трудового распорядка, установленные на предприятии, не допускать нарушений трудовой и производственной дисциплины.

Рабочий стол с учётом характера выполняемой работы должен иметь достаточный размер для рационального размещения монитора (дисплея), клавиатуры, другого используемого оборудования и документов, поверхность, обладающую низкой отражающей способностью.

Клавиатура располагается на поверхности стола таким образом, чтобы пространство перед клавиатурой было достаточным для опоры рук работника (на расстоянии не менее чем 300 мм от края, обращённого к работнику).

Чтобы обеспечивалось удобство зрительного наблюдения, быстрое и точное считывание информации, плоскость экрана монитора располагается ниже уровня глаз работника предпочтительно перпендикулярно к нормальной линии взгляда работника (нормальная линия взгляда – 15 град. вниз от горизонтали).

ПРИЛОЖЕНИЕ Д
(обязательное)
Графический интерфейс программного комплекса

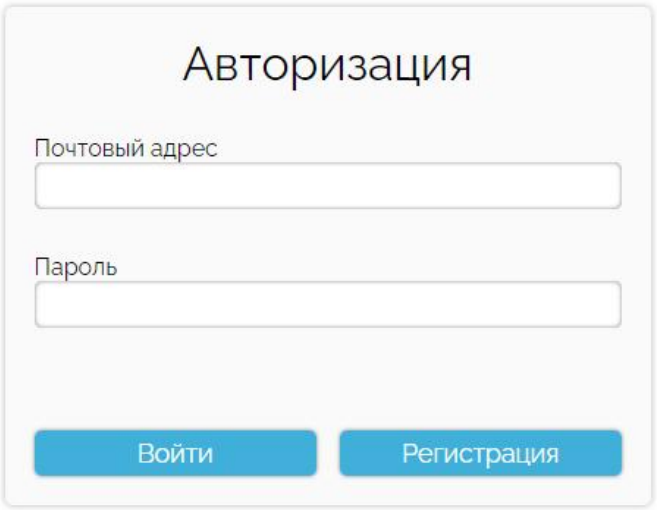

Рисунок Д.1 – Внешний вид окна авторизации

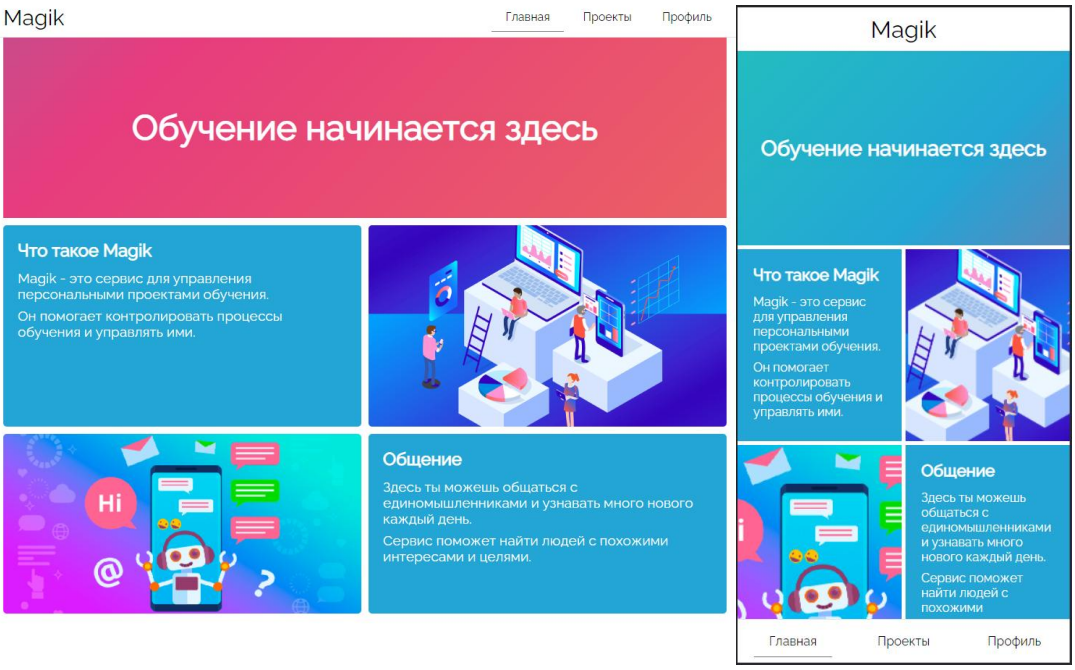

Рисунок Д.2 – Внешний вид главного окна приложения

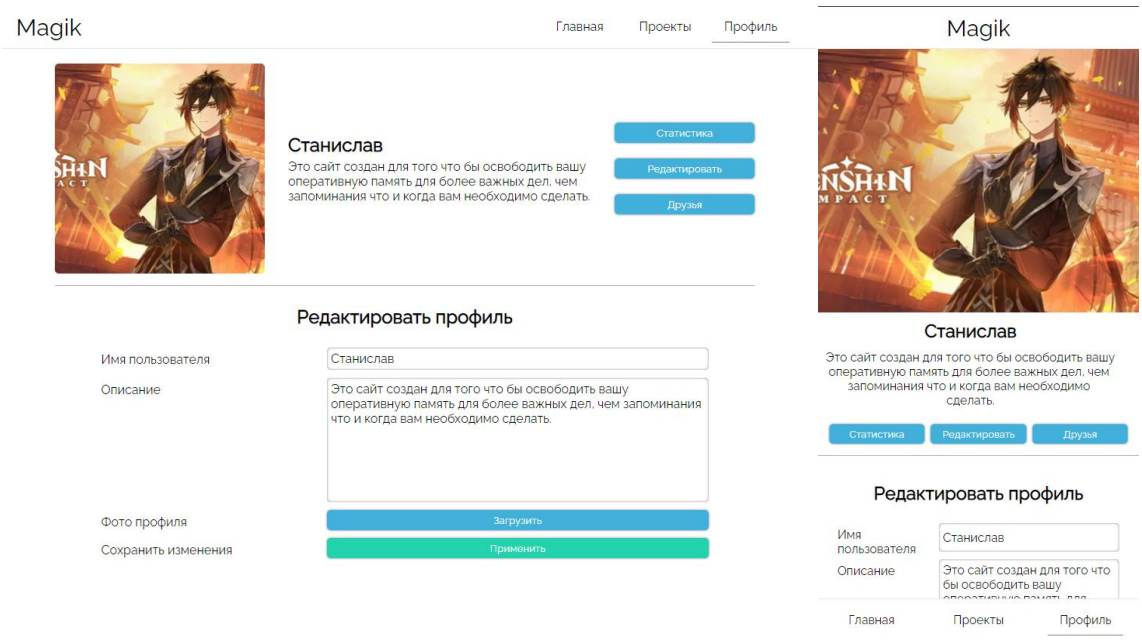

Рисунок Д.3 – Внешний вид профиля пользователя

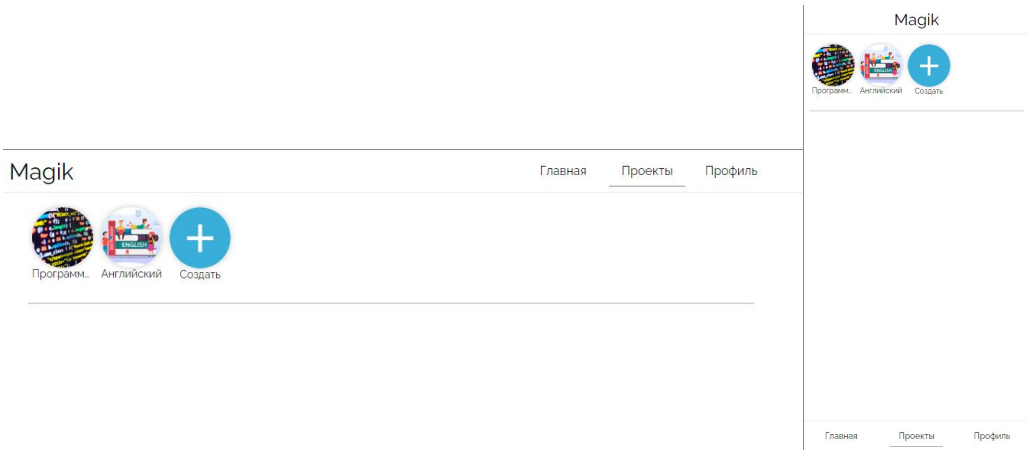

Рисунок Д.4 – Внешний вид областей проектов приложения

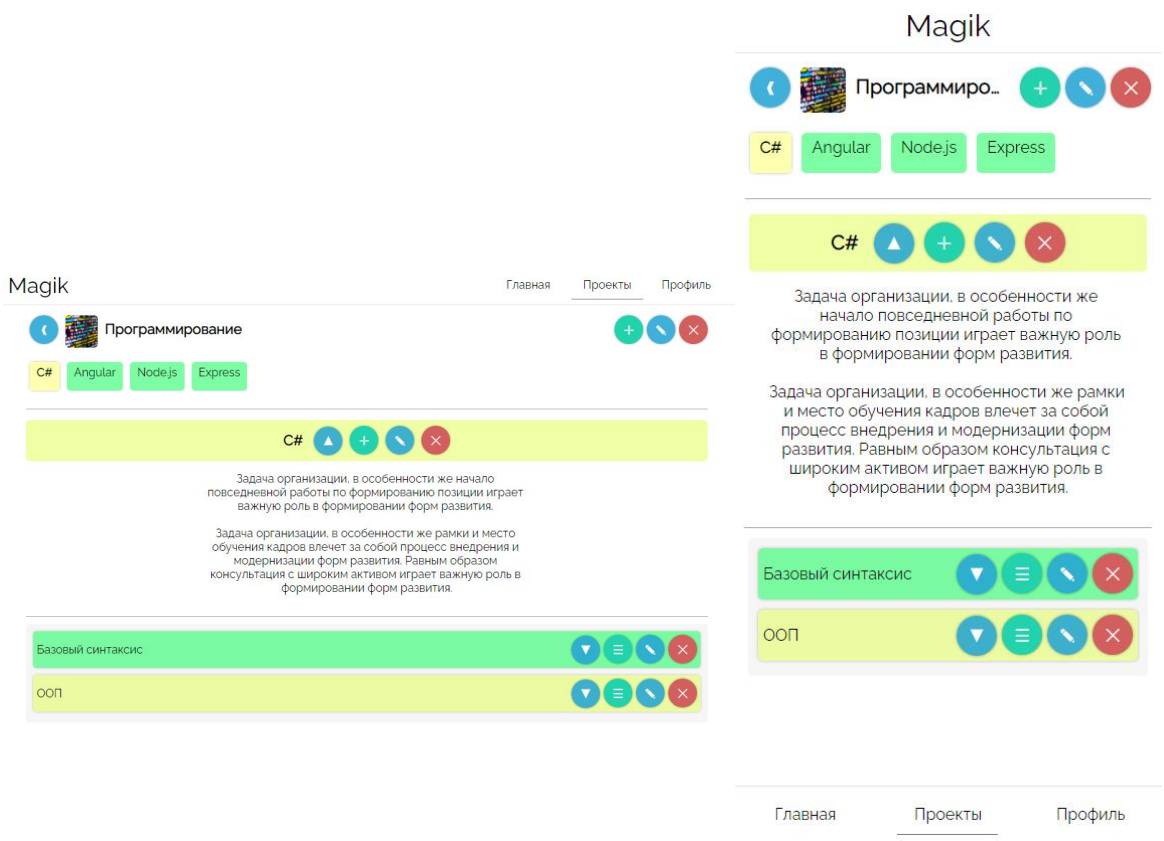

Рисунок Д.5 – Внешний вид выбранной области проектов и проекта

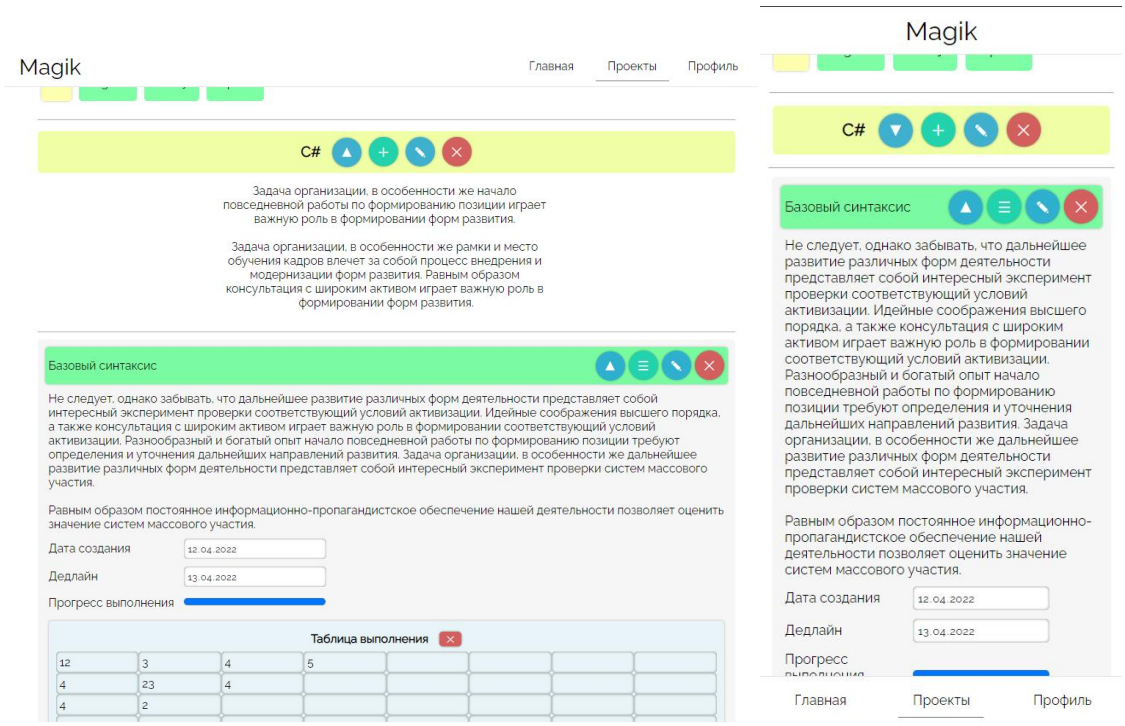

Рисунок Д.6 – Внешний вид раскрытой стадии проектов

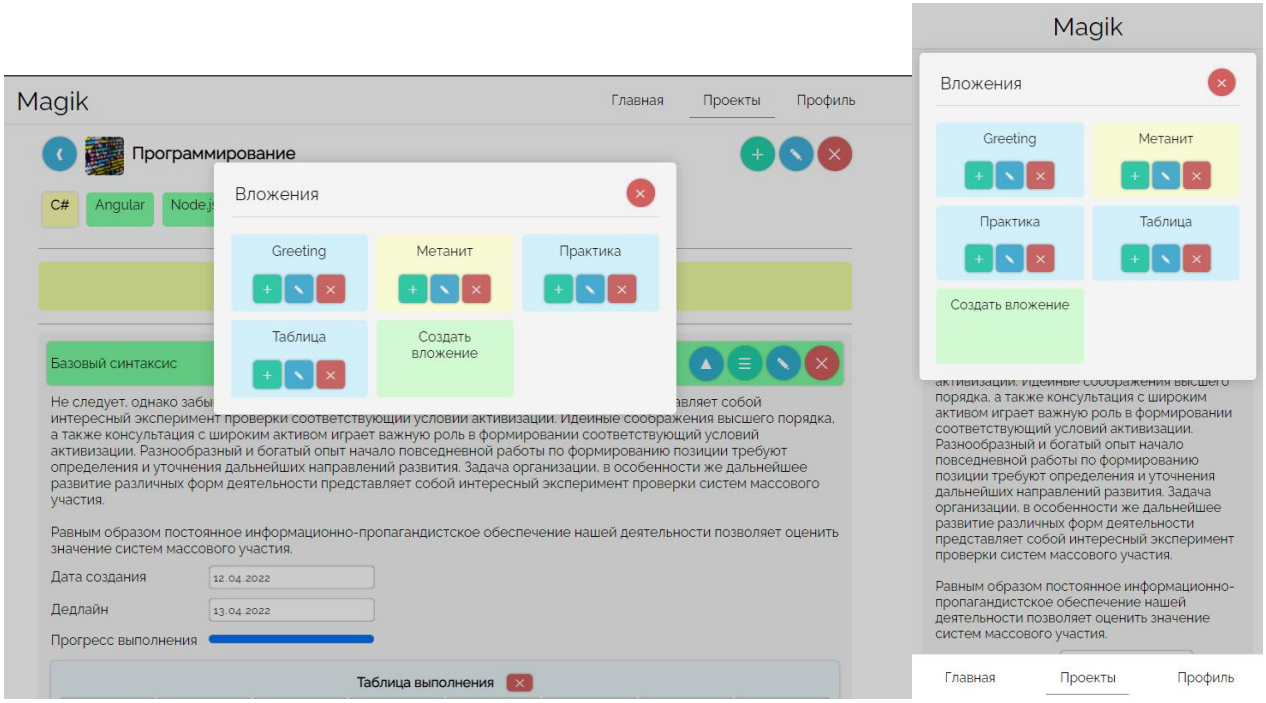

Рисунок Д.7 – Внешний вид окна вложений

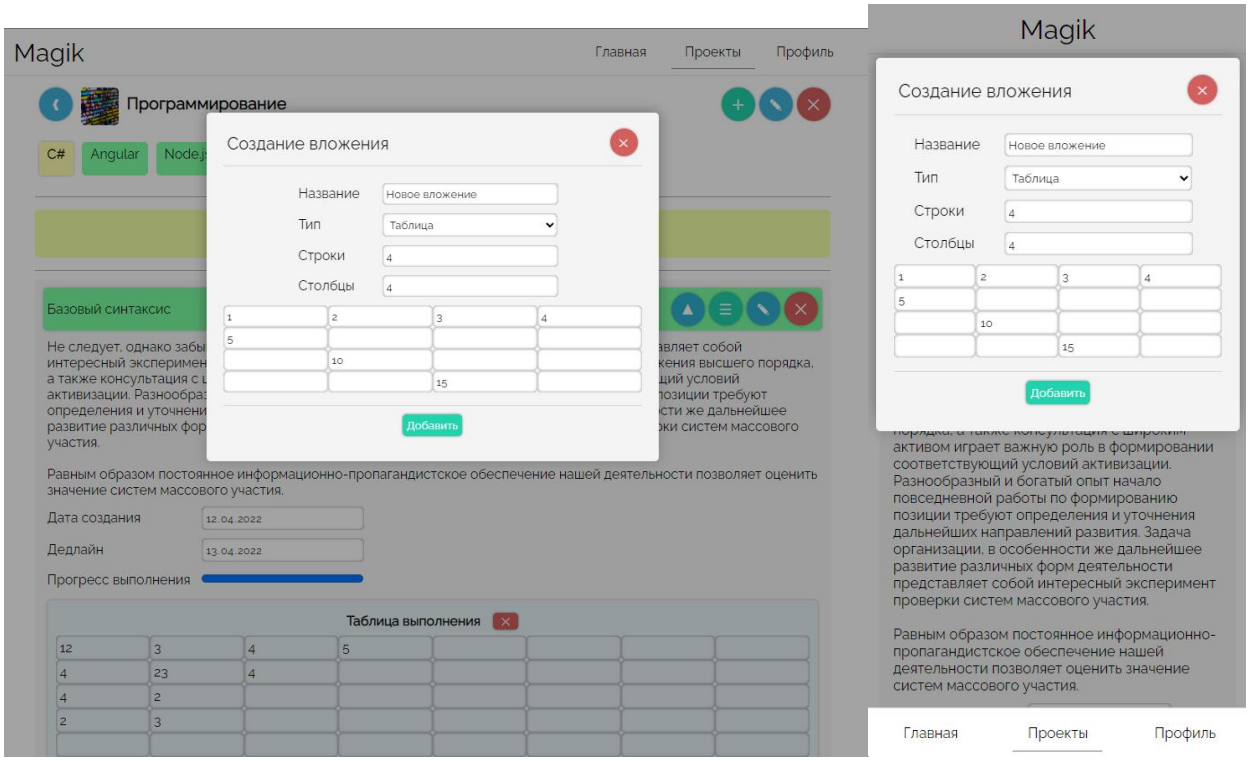

Рисунок Д.8 – Внешний вид окна создания вложения