

Пермский филиал федерального государственного автономного образовательного
учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Магистерская школа НИУ ВШЭ – Пермь

Бизнес-аналитика
(ранее Информационная аналитика в управлении предприятием)
Магистратура
38.04.05 «Бизнес-информатика»

О Т Ч Е Т

по проектной практике

Проект

**Создание Open-source проекта для решения учебных
практикоориентированных задач программной инженерии**

Выполнил студент гр. ИАУП-22-1
Михайлов Александр Витальевич

(подпись)

Проверил:

Доктор педагогических наук,
Профессор кафедры информационных технологий в бизнесе.
Плотникова Евгения Григорьевна

(подпись)

(дата)

Пермь, 2023 год

Содержание

Общее описание проекта.....	4
Введение	5
Глава 1. Анализ аналогов, источников и предметной области в контексте программного обучения ИТ-специалистов.....	7
1.1. Анализ предметной области.....	7
1.1.1. Описание предметной области.....	7
1.1.2. Анализ организационной структуры	11
1.1.3. Анализ заинтересованных сторон.....	12
1.1.4. Анализ бизнес-процессов.....	17
1.1.5. Поддержка бизнес-процессов	19
1.1.6. Используемая инфраструктура.....	20
1.1.7. Требования к разрабатываемому продукту.....	21
1.2. Анализ существующих онлайн калькуляторов	24
1.3. Предметная область реализации онлайн калькуляторов	30
1.4. Требования к онлайн калькулятору.....	34
1.4.1. Функциональные требования	34
1.4.2. Требования к программной документации	35
1.4.3. Требования к эргономике	35
1.4.3. Требования к хостингу	35
1.4.4. Требования к информационной и программной совместимости	35
1.4.5. Требования к клиентскому программному обеспечению	36
1.4.6. Требования к маркировке и упаковке.....	36
1.4.7. Прочие требования	36
1.5. Прототип системы.....	36
1.5.1. Главная страница	36
1.5.2. Раздел «Определитель матрицы».....	38
1.5.3. Раздел «Числа Фибоначчи»	39
1.5.4. Раздел «НОД и НОК двух чисел»	40
1.5.5. Раздел «Расход топлива для поездки на заданное расстояние».....	41
1.5.6. Раздел «Проверка числа на простоту».....	42

1.5.7. Раздел «Проверка ряда чисел на совершенность»	43
1.5.8. Контроль ввода входных данных и соответствие прототипа заявленным требованиям.....	45
Глава 2. Проектирование онлайн калькулятора	46
2.1. Проектирование API	46
2.2. Проектирование серверной части онлайн калькулятора.....	48
2.2.1 Проектирование структуры серверной части онлайн калькулятора	48
2.2.2 Проектирование взаимодействия элементов серверной части онлайн калькулятора.....	54
Глава 3. Разработка онлайн калькулятора.....	63
3.1. Разработка серверной части онлайн калькулятора	66
3.1.1. Структура проекта и конфигурационные файлы программного модуля ...	67
3.1.2. Разработка ядра Онлайн-калькулятора.....	69
3.1.3. Разработка API Онлайн-калькулятора.....	72
3.1.4. Создание docker-образа для серверной части Онлайн-калькулятора.....	74
3.1.5. Разработка алгоритмов для Онлайн-калькулятора с участием студентов..	76
3.2. Разработка клиентской части онлайн калькулятора	81
3.3. DevOps.....	83
Глава 4. Тестирование	87
4.1. План тестирования	87
4.2. Реализация плана тестирования.....	90
4.3. Модульное тестирование.....	91
4.4. Проведение тестирования.....	96
Заключение	98
Результат проекта	102
Список использованных источников	104
Приложение А	108
Приложение Б.....	110
Приложение В	111

Общее описание проекта

Инициатор: НИУ ВШЭ – Пермь, кафедра информационных технологий в бизнесе.

Руководитель проекта: Плотникова Евгения Григорьевна.

Тип проекта: прикладной.

Место работы по проекту: Высшая Школа Экономики, Кафедра информационных технологий в бизнесе, Пермь.

Содержательная часть отчета, включающая описание хода выполнения проектного задания, описание результатов проекта (продукта), а также описание использованных в проекте способов и технологий, представлена далее в отчете. Описание ролей каждого участника проектной команды и оценка индивидуальных результатов выполнения проекта, сформированных / развитых компетенций приведены в разделе Заключение. Результаты проекта представлены в завершающей части отчета.

Введение

В настоящее время сфера программной инженерии активно развивается и требует от специалистов не только высокой квалификации, но и навыков работы в команде над реальными проектами. Однако учебные задачи, предлагаемые в рамках курсов по программной инженерии, часто недостаточно практико-ориентированы и не дают студентам достаточного опыта для успешного участия в проектах в реальном мире.

Цель данной работы заключается в разработке программного продукта «Онлайн-калькулятор» в MVP версии. Для достижения цели были поставлены следующие задачи: провести анализ материалов по участию студентов в IT-проектах, разработать архитектуру расширяемого программного продукта для использования в образовательных целях, разработать программный продукт и оформить документацию к нему, провести тестирование программного продукта, развернуть программный продукт в рабочей среде (production) и отладочной среде (development).

Объектом исследования являются учебные задачи программной инженерии, способы их постановки и платформы для их решения. Предметом исследования является открытый проект по разработке программного продукта, а также анализ эффективности его использования в учебных целях.

Несмотря на то, что проекты с открытым исходным кодом и разработка образовательных программ имеют достаточную степень разработанности, данная работа представляет собой новую разработку, которая позволяет студентам получить более практический опыт в области программной инженерии и разработки программных продуктов.

Теоретическая значимость данной работы заключается в том, что создание open-source проекта для решения учебных практико-ориентированных задач программной инженерии позволит студентам получить практические навыки в программной инженерии, а также даст возможность принять участие в реальном проекте, что поможет им получить опыт работы в команде и научиться работать с системами контроля версий и инструментами разработки. Кроме того, результаты данного проекта могут быть использованы в качестве учебного материала для обучения студентов программной инженерии и повышения качества образования в данной области.

Прикладная ценность данной работы заключается в создании готового программного продукта, который может быть использован студентами в учебных целях и расширен для создания других приложений. Кроме того, открытый доступ к исходному коду и документации может привлечь внимание разработчиков, что может привести к дальнейшей разработке проекта и улучшению качества образования в области программной инженерии.

Глава 1. Анализ аналогов, источников и предметной области в контексте программного обучения ИТ-специалистов

Рынок труда в ИТ-отрасли в 2020-2023 является динамичным и быстро развивающимся. Согласно отчету Burning Glass Technologies [1], в 2020 году спрос на ИТ-специалистов в США составил более 12 миллионов вакансий, что на 24% больше, чем в 2019 году. Данный отчет также показывает, что наибольший спрос был на специалистов в области программирования, аналитики данных, информационной безопасности и облачных технологий. В данной главе работы представлен анализ предметной области, приведен анализ существующих аналогов, рассмотрена область реализации онлайн калькуляторов, сформулированы функциональные требования к проекту и продемонстрирован разработанный прототип сайта.

1.1. Анализ предметной области

Анализ предметной области является важным этапом при разработке программы обучения ИТ-специалистов. В данном случае рассмотрены основные понятия и термины, связанные с программной инженерией и командной работой в ИТ. Помимо этого, представлена информация по компетенциям, ожиданиям работодателей, востребованным технологиям и навыкам для ИТ-профессий, а также о состоянии рынка труда в ИТ на текущий момент. Все это позволит более глубоко понять предметную область и сформировать программу обучения, соответствующую современным требованиям рынка.

1.1.1. Описание предметной области

В работе рассматривается бизнес-процесс проведения семинарских занятий по курсу «Алгоритмы и структуры данных» [2] в НИУ Высшая школа экономики в Перми.

Семинар — это форма занятия, служащая подведению итогов самостоятельной работы студентов и старших школьников, способствует более глубокому усвоению и обобщению изученного, отработке навыков применения знаний. Проводится в форме собеседования, конференций, защиты разработанных проектов, решения ситуационных задач, фрагментов обучающих игр, пресс-конференций и т.д. [3].

«Алгоритмы и структуры данных» — учебный курс, ориентированный на изучение базовых алгоритмов и структур данных, используемых при решении задач поиска,

хранения и обработки информации, а также при проектировании и разработке средств реализации прикладных информационных технологий [2].

Семинарские занятия проходят еженедельно в компьютерных классах пермского кампуса Высшей школы экономики. Занятия проводятся согласно расписания, составляемого специалистом кафедры по учебно-методической работе. Преподаватель проводит занятия, выдает задания, консультирует, принимает и оценивает решения. В занятиях участвуют студенты второго курса направлений «Бизнес-информатика» и «Программная инженерия». Подготовку кадров в области информационных технологий в экономике и бизнесе по направлениям подготовки «Бизнес-информатика» и «Программная инженерия» обеспечивает кафедра информационных технологий в бизнесе. Кафедра входит в состав факультета экономики, менеджмента и бизнес-информатики НИУ ВШЭ – Пермь.

Для семинарских занятий студенты объединяются в команды по трое и совместно выполняют задания. На каждом занятии студентам предлагается задание, которое они должны выполнить в составе команды в течении недели до следующего занятия. Так как значительная часть заданий связана с написанием программного кода, взаимодействие преподавателя и студентов организовано посредством системы контроля версий Git и проекта на облачном сервисе GitHub. GitHub — это облачная платформа для хостинга и совместной работы над программными проектами, основанная на системе контроля версий Git [4]. Еженедельно в проекте публикуются задания и в течении следующей недели студенты добавляют в проект свои решения. На сервисе GitHub организованы автоматическая проверка и тестирование программного кода из решений, представленных студентами. Если решение не проходит автоматическую проверку GitHub автоматически уведомляет об этом автора решения. Если решение прошло автоматическую проверку, то его проверяет преподаватель и либо принимает задание и выставляет оценку, либо публикует вопросы и замечания к решению и возвращает его на доработку, о чем автор решения автоматически уведомляется. Автор может добавить к решению комментарии и пояснения, исправить указанные недочеты. Проект располагается на сервере GitHub. Преподаватель и студенты используют интегрированную среду работы, установленную на собственный персональный компьютер или ноутбук. Среда разработки имеет встроенные инструменты системы

контроля версий и взаимодействия с проектом на GitHub. Среда разработки доступна для студентов и преподавателей ВШЭ по студенческой лицензии.

Проведению курса занятий предшествует процесс подготовки, инициируемый назначением преподавателя на курс. В ходе подготовки осуществляется выбор тем для занятий и по каждой теме готовятся задания. Задания предполагают программную реализацию определенного алгоритма либо решение некоторой задачи с использованием алгоритма. Задание сопровождается эталонным решением, с которым соотносятся решения, представленные студентами. Для заданий, предполагающих программную реализацию алгоритма, предусматривается набор автоматических тестов для проверки правильности решения.

Для моделирования предметной области проведения практических занятий в данной работе используется ArchiMate — язык моделирования архитектуры предприятия, разрабатываемый консорциумом The Open Group, базирующийся на стандарте IEEE 1471 (рекомендуемые методы описания архитектуры программных систем), предназначенный для описания, анализа и визуализации архитектуры предприятия [5]. В ходе проведения анализа предполагается построить, с помощью языка ArchiMate, модель предметной области и разработать представления модели, характеризующие предметную область с различных точек зрения на различных слоях архитектуры. Представление для модели ArchiMate является ее выделенной частью, которая иллюстрирует определенный перечень вопросов и предназначена для некоторого круга стейкхолдеров. Возможности Archimate по моделированию различных слоев архитектуры предприятия представлены на рис. 1.1.

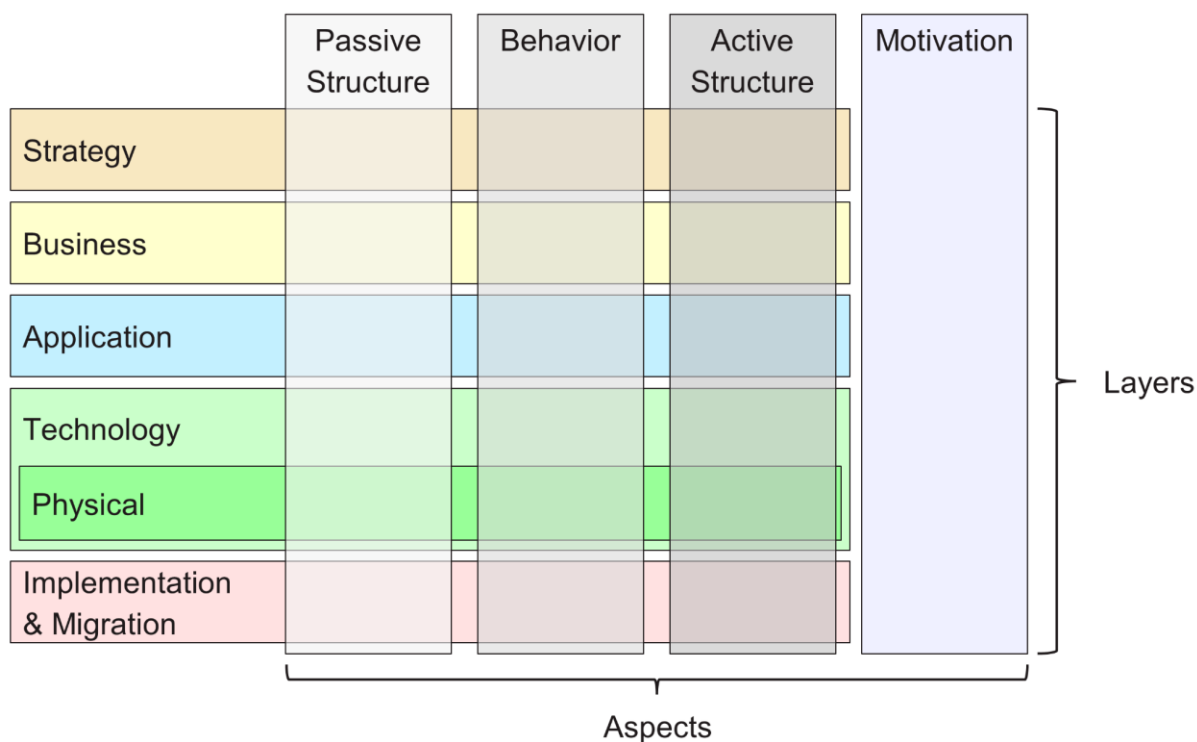


Рисунок 1.1. Структура фреймворка Archimate

В ходе анализа и моделирования:

1. Проведен анализ организационной структуры, связанной с предметной областью и разработано представление Оргструктура на бизнес-уровне Archimate.
2. Выявлены стейкхолдеры, их интересы и движущие силы, разработано представление Мотивация, отображающее цели и задачи работы.
3. Проведен анализ бизнес-процессов предметной области, разработано представление Процессы на бизнес-уровне Archimate, иллюстрирующее предлагаемые изменения.
4. Разработано представление Приложения, на соответствующем уровне Archimate, иллюстрирующее какие приложения используются для поддержки бизнес-процессов с учетом предлагаемых изменений.
5. Разработано представление Технологии, на технологическом и физическом уровнях Archimate, представляющее необходимую инфраструктуру для функционирования описанных приложений.
6. Сформулированы требования к разрабатываемому продукту, с точки зрения проведения практических занятий со студентами.

1.1.2. Анализ организационной структуры

Рассматриваемые семинарские занятия проходят в НИУ Высшая школа экономики в Перми. В процессе организации и проведения занятий непосредственно задействована кафедра информационных технологий в бизнесе и ее сотрудники – преподаватели, специалист по учебно-методической работе и заведующий кафедрой. Кафедра входит в состав факультета экономики, менеджмента и бизнес-информатики. В состав факультета входят направления подготовки бакалавров - «Бизнес-информатика» и «Программная инженерия». По каждому направлению сформированы несколько групп студентов. Для работы на семинарских занятиях студенты объединяются в команды.

Помимо бакалавров на факультете осуществляется подготовка магистров, в том числе по направлению Бизнес-аналитика. Студенты магистратуры также объединяются в группы для работы по некоторым предметам и реализации группового проекта. Кроме того, некоторые студенты магистратуры выступают в роли преподавателей и проводят семинарские занятия для бакалавров по различным дисциплинам.

Для разработки представления Организационная структура выбрана точка зрения Organization Viewpoint. Согласно спецификации языка ArchiMate 3.1 данная точка зрения представляет внутреннюю структуру организации и позволяет выделить компетенции, полномочия и ответственность в организации. Разработанный вид Оргструктура представлен на рис. 1.2. Представляемая информация предназначена для широкого круга заинтересованных сторон, в том числе архитектора, менеджеров и сотрудников. Вид разрабатывается для целей проектирования, принятия решений и информирования заинтересованных сторон.

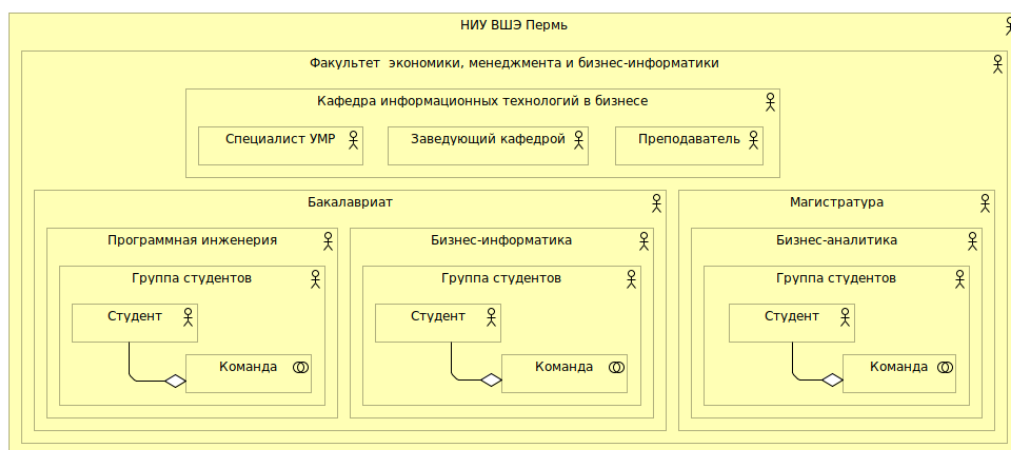


Рисунок 1.2. Представление Оргструктура

1.1.3. Анализ заинтересованных сторон

На основе описания предметной области и анализа организационной структуры выделены заинтересованные стороны - студент, преподаватель и заведующий кафедрой информационных технологий в бизнесе. В процессе анализа литературы по теме организации образовательного процесса, изучена статья Калевко В.В. «Управление образовательной программой вузов в контексте подготовки конкурентоспособных разработчиков программного обеспечения» [7], в которой рассматриваются взаимоотношения образовательной организации с внешней средой. Иллюстрация указанных взаимоотношений представлена на рис. 1.3.



Рисунок 1.3. Взаимоотношения образовательной организации с внешней средой

Применительно к текущей работе, заинтересованные стороны - преподаватель и заведующий кафедрой представляют систему управления образовательной программой, а студент - это бывший абитуриент и будущий подготовленный специалист. Кроме того, необходимо выделить в качестве заинтересованной стороны работодателя, так как именно от работодателей исходит потребность в подготовке специалистов. Потребность работодателей конкретизируется в ожидаемых знаниях, навыках, компетенциях, которыми должен обладать выпускник после освоения образовательной программы. Таким образом, в контексте текущей работы выделены заинтересованные стороны - студент, преподаватель, заведующий кафедрой и работодатель, изображенные на

представлении Мотивация (рис. 1.4). Для представления выбрана точка зрения Мотивация, которая иллюстрирует заинтересованные стороны, связанные с ними движущие силы, основные цели изменений, планируемые результаты, применяемые принципы, требования и ограничения.

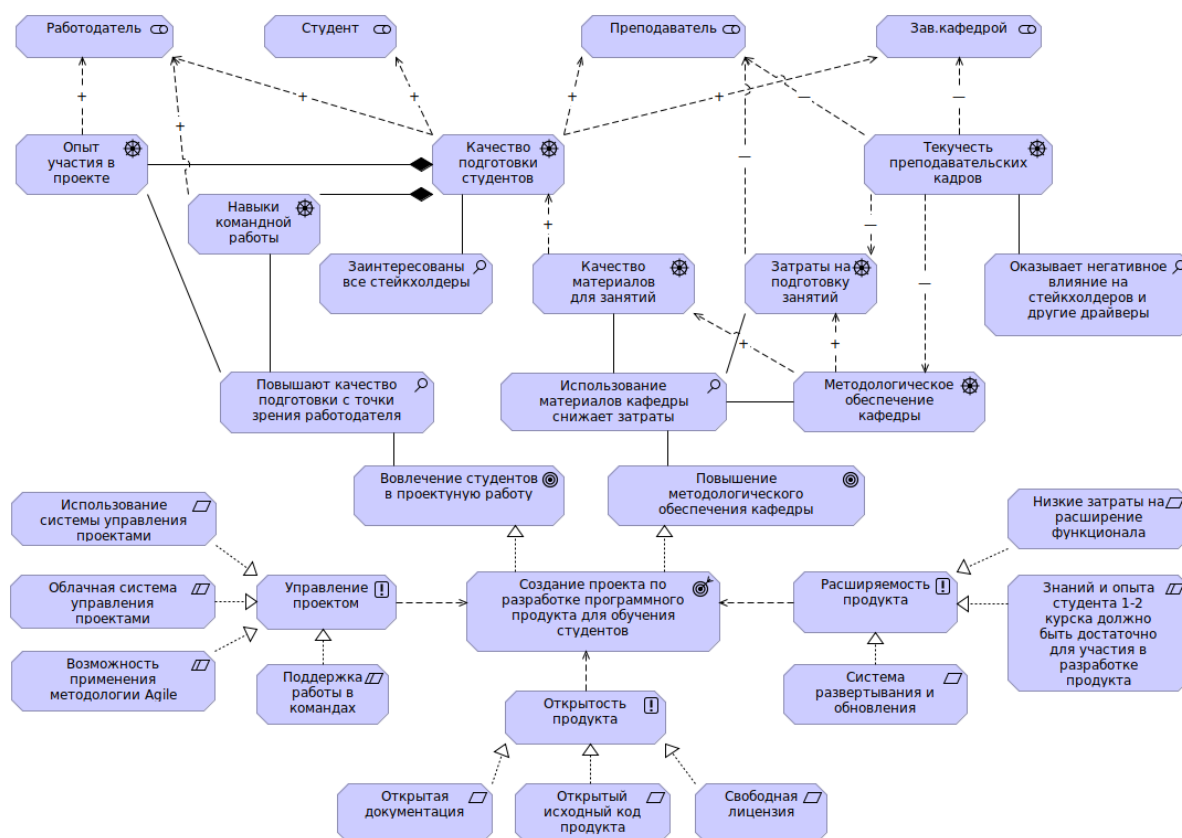


Рисунок 1.4. Представление Мотивация

На представление мотивация вынесены драйверы - движущие силы способствующие изменениям в организации. Основным драйвером, влияющим на все заинтересованные стороны, является качество подготовки студентов. В контексте данной работы можно выделить два критерия качества подготовки, которые являются важными для работодателя, это опыт участия в проектах и навыки командной работы. В статье журнала БИТ «Как и чему учить будущих звезд ИТ?» [7] многие из опрашиваемых представителей ИТ-индустрии отмечали важность проектной и командной работы для подготовки кадров ИТ-отрасли. Так Юлия Шикова, директор учебного центра «Сетевая Академия ЛАНИТ», отметила, что опыт реализации проектов повышает конкурентоспособность выпускников на рынке труда. По мнению Алены Дядченко, руководителя разработки компании Granatum Solutions, подготовка ИТ-специалистов в

России «мало соответствует последним трендам и технологиям» вследствие чего, при трудоустройстве важную роль играет опыт, в том числе опыт разработки, например, в open-source проектах. По словам Андрея Бороздюхина, директора саратовского филиала компании Bell Integrator, для ИТ-специалистов необходимо уметь работать в команде и участвовать в коллективной разработке. Схожие ожидания работодателей можно проследить и в иностранных источниках, например, по результатам опроса работодателей Национальной ассоциацией колледжей и работодателей США [8] наиболее важными качествами являются навыки решения проблем и умение работать в команде. В статье «Анализ пробелов в навыках после пандемии COVID» [9] исследуются востребованность работодателями различных навыков и возможности учебных программ политехнического колледжа Торонто (Humber College) по подготовке конкурентоспособных специалистов. На основе собранных данных, в качестве наиболее востребованных указаны навыки коммуникации, а навыки командной работы заняли четвертое место в рейтинге.

В качестве драйвера для заведующего кафедрой и преподавателя выделена текучесть кадров - семинарские занятия, как правило, проводят молодые преподаватели, зачастую студенты магистратуры, большинство из которых не задерживается на преподавательской работе. Таким образом, важную роль играет методологическое обеспечение кафедры - наличие теоретических и практических материалов для проведения занятий снижает затраты преподавателя на подготовку к занятиям, повышает качество материалов для занятий, что в свою очередь способствует повышению качества подготовки студентов. Кроме того, важно иметь возможность улучшать методологическое обеспечение кафедры и сохранять наработки для дальнейшего улучшения и переиспользования.

С учетом описанных драйверов и их оценок сформулированы цели планируемых изменений - вовлечение студентов в командную работу и повышение методологического обеспечения кафедры. Поставленные цели планируется достигнуть посредством создания проекта по разработке программного продукта для обучения студентов. Проект по разработке программного продукта является реальным конечным результатом планируемых изменений. Для реализации проекта определены принципы - свойства, которыми он должен обладать, эти принципы реализуются набором требований и ограничений.

Так например, принцип открытости продукта реализуется требованиями наличия свободной лицензии, предполагается использование лицензии MIT [10] для всех материалов проекта, и соответственно требованиями открытого исходного кода и документации - предполагается использование для проекта публичных репозиторий, размещаемых на хостинге для программных проектов.

Еще одним принципом для проекта является расширяемость. Так как программный продукт предназначен для практических занятий, важно, чтобы его архитектура позволяла студентам расширять функциональность, посредством решения учебных задач. Принцип расширяемости реализуется требованиями низких затрат на расширение и ограничением о достаточности знаний и опыта студентов для участия в разработке. Важным требованием для расширяемости является наличие системы развертывания и обновления продукта, которая предполагает применение методологии DevOps – набора практик и программных инструментов, направленных на повышение эффективности и скорости разработки систем, а также обеспечение непрерывности процесса разработки и развертывания программного обеспечения [11]. В статье «Оптимальный подход к разработке программного обеспечения с использованием современных методологий и технических средств» [12] автор описывает современные требования, как к процессу разработки программного обеспечения, так и к инфраструктуре, в частности автор указывает на необходимость применения быстрых и удобных инструментов для установки, поддержки и обновления разрабатываемого программного обеспечения. Применение методологии DevOps, включающей автоматизацию процесса тестирования, сборки и поставки продукта, позволит упростить участие студентов в разработке и снизить требования к специфическим навыкам студентов. Аналогичный опыт применения методологии DevOps описывается в работе «DEVOPS как механизм быстрого погружения в ИТ-проект» [13] - студенты привлекались к участию в разработке веб-приложения «Электронные сервисы вуза» для Омского государственного университета, автоматизация процессов сборки приложения студенты смогли эффективно выполнить поставленные задачи и получили опыт участия в реальном проекте.

Третьим принципом является управление проектом, в связи с тем, что учебные задачи по разработке продукта планируется выполнять в рамках проекта, развивая тем

самым у студентов навыки проектной работы. Принцип реализуется требованием использования системы управления проектами. Также имеется ряд ограничений, например, система управления должна быть облачной, использоваться по модели SaaS (программное обеспечение как услуга), чтобы исключить проблемы и затраты, связанные с установкой и настройкой системы. Студенты выполняют задания в командах, следовательно система должна поддерживать возможность командной работы. Работа в командах и взаимодействие с другими командами в ходе проектной работы позволит студентам получить важный практический опыт командной и проектной работы. В рамках анализа предметной области были изучены статьи по теме организации проектной работы студентов, например, «Модель формирования социальной компетентности у будущих программистов в условиях коллаборативного электронного обучения» [14]. В этой статье авторы предлагают использовать гибкую методологию Agile и технологию Scrum для управления учебным проектом. Для управления проектом авторы предлагают использовать отечественную платформу Devprom ALM [15], а для хостинга проекта платформу GitHub. В работе «Методологии управления программными проектами в подготовке IT-специалистов» [16] автор отмечает важность изучения студентами вопросов управления программными проектами и в качестве наиболее эффективных технологий управления программными продуктами выделяет Agile и Scrum. Также в работе упоминается проект нидерландского педагога Вилли Вейнандса EduScrum - система организации обучения, основанная на методологии Scrum, фреймворк, в рамках которого учителя и обучающиеся решают комплексные задачи высокой сложности и стремятся к достижению максимально значимых образовательных целей продуктивно и творчески [17]. Перспективным направлением использования Scrum в вузах в статье [16] представляется выполнение студентами сквозных междисциплинарных проектов с привлечением студентов старших курсов для работы в роли скрам-мастеров. В работе «Научить студента думать: Scrum как метод продуктивного обучения в учебном заведении» [18] рассматриваются возможности использования Scrum, как метода продуктивного обучения. Продуктивное обучение предполагает создание в ходе образовательной деятельности некоторого продукта - идеи, схемы, модели, текста и т.д. [18]. Методология Scrum, по мнению автора, является одной из наиболее эффективных в подготовке студентов к обучению на протяжении всей жизни и формирования

востребованных навыков. В работе также рассматривается опыт применения фреймворка EduScrum. В работе «Использование методологии AGILE в командной разработке проекта «Экология» [19] с применением системы контроля версий GIT» описывается опыт командной разработки программного продукта «Экология» для кафедры «Теплофизика и информатика в металлургии» Уральского федерального университета. Управление проектом осуществлялось с использованием методологии Agile и инструментов от компании Atlassian - хостинг проекта на платформе Atlassian Bitbucket, управление проектом на платформе Atlassian Jira. С указанными публикациями можно согласиться в том, что гибкая методология Agile и фреймворк Scrum являются современными, эффективными для управления проектами по разработке программных продуктов. Таким образом, можно сформулировать ограничение к принципу управление проектом - возможность применения методологии Agile.

1.1.4. Анализ бизнес-процессов

Для моделирования процессов предметной области выбрана точка зрения Business Process Cooperation Viewpoint, отражающая взаимодействие бизнес-процессов и их окружения, а также использование бизнес-объектов. Выбранная точка зрения представляет интересы архитектора и руководства и иллюстрирует причинно-следственные связи между основными бизнес-процессами, а также использование общих данных. Представление Процессы, согласно спецификации, разрабатывается для целей проектирования и принятия решений.

Разработанное представление Процессы изображено на рис. 1.5. На представление вынесены два основных бизнес-процесса – Подготовка курса и Выполнение задания. На процесс Подготовка курса назначается роль Преподаватель, процесс проходит до начала учебного года. Процесс Выполнение задания многократно повторяется в течении учебного года и задействует роли Преподаватель и Студент. Преподаватель публикует задание, после чего команды приступают к его решению. После предоставления готового решения, выполняются автоматические тесты, если задание предполагает написание программного кода, и проверка решения преподавателем. В результате проверки преподаватель оценивает решение, либо возвращает его на доработку. Также преподаватель проводит консультации, если у студентов возникают вопросы по выполнению задания. На виде представлены бизнес-объекты Задание и Решение

доступные различным процессам. В качестве бизнес-сервисов выделены - сервис Задания, позволяющий студентам ознакомиться с опубликованными заданиями, сервис Обсуждение, предоставляющий возможности консультирования с преподавателем, и сервис Проверка программ, позволяющий автоматически проверить решение в виде программного кода.

Красным цветом на представлении выделены элементы, которые планируется внедрить в ходе реализации изменений. При этом представление иллюстрирует, что существующие процессы практически не изменяются, а лишь добавляются новые формы их реализации, вводятся новые роли для участников, в контексте управления проектом с использованием фреймворка Scrum. Так решение задания может быть реализовано в виде программного проекта, в котором команда студентов выступает в роли команды разработки, а преподаватель в роли скрам-мастера. Задание в этом случае может быть представлено в виде некоторой пользовательской истории, запрашивающей расширение функциональности продукта. Решение такого задания в итоге будет представлять инкремент продукта - версия разрабатываемого продукта, готовая к использованию и увеличивающая его ценность для конечных пользователей. Кроме того добавляется новый сервис - Управление проектом, позволяющий проводить учет и контроль реализации проекта.

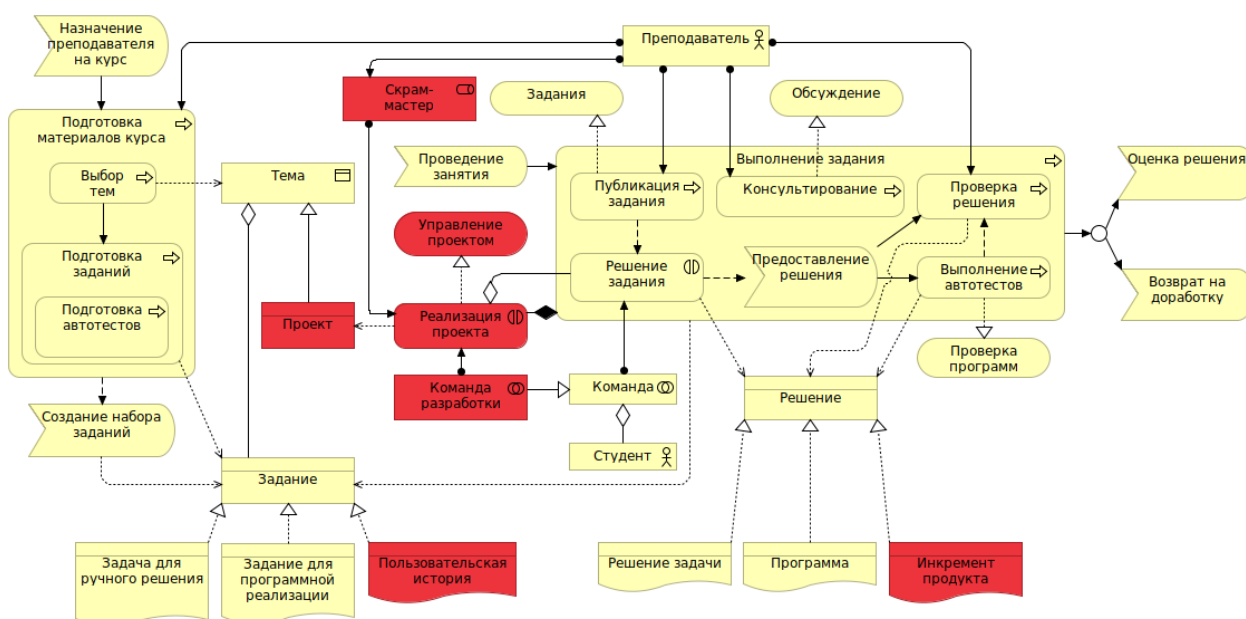


Рисунок 1.5. Представление Процессы

1.1.5. Поддержка бизнес-процессов

Для моделирования поддержки процессов выбрана точка зрения Application Usage Viewpoint, отражающая то, как приложения взаимодействуют между собой и используются для поддержки бизнес-процессов. Выбранная точка зрения представляет интересы архитектора и руководства и используется, согласно спецификации, для целей проектирования и принятия решений. В контексте текущего проекта данный вид можно использовать для целей информирования заинтересованных сторон, так как на нем иллюстрируются предлагаемые изменения, в том числе поддержка новых процессов.

Разработанное представление Приложения изображено на рис. 1.6. Представление иллюстрирует, что для поддержки процесса выполнения заданий используется сервис GitHub. Задания и решения публикуются в репозитории на хостинге GitHub, для выполнения автотестов используется сервис GitHub Actions. Тесты запускаются автоматически на виртуальной машине GitHub, при публикации студентами решения в репозиторий GitHub. Для подготовки решения, связанного с написанием программного кода, студенты используют интегрированную среду разработки.

Красным цветом на представлении выделены элементы, которые планируется внедрить в ходе реализации изменений. Для поддержки внедряемого процесса Реализация проекта планируется использовать сервис GitHub Projects, позволяющий организовать управление проектом по разработке программного продукта, исходные коды которого размещены в репозиториях GitHub. Проект предполагает разработку и поддержку некоторого программного продукта, в связи с чем на представление вынесены приложения составляющие программный продукт и сервис, который это продукт реализует.

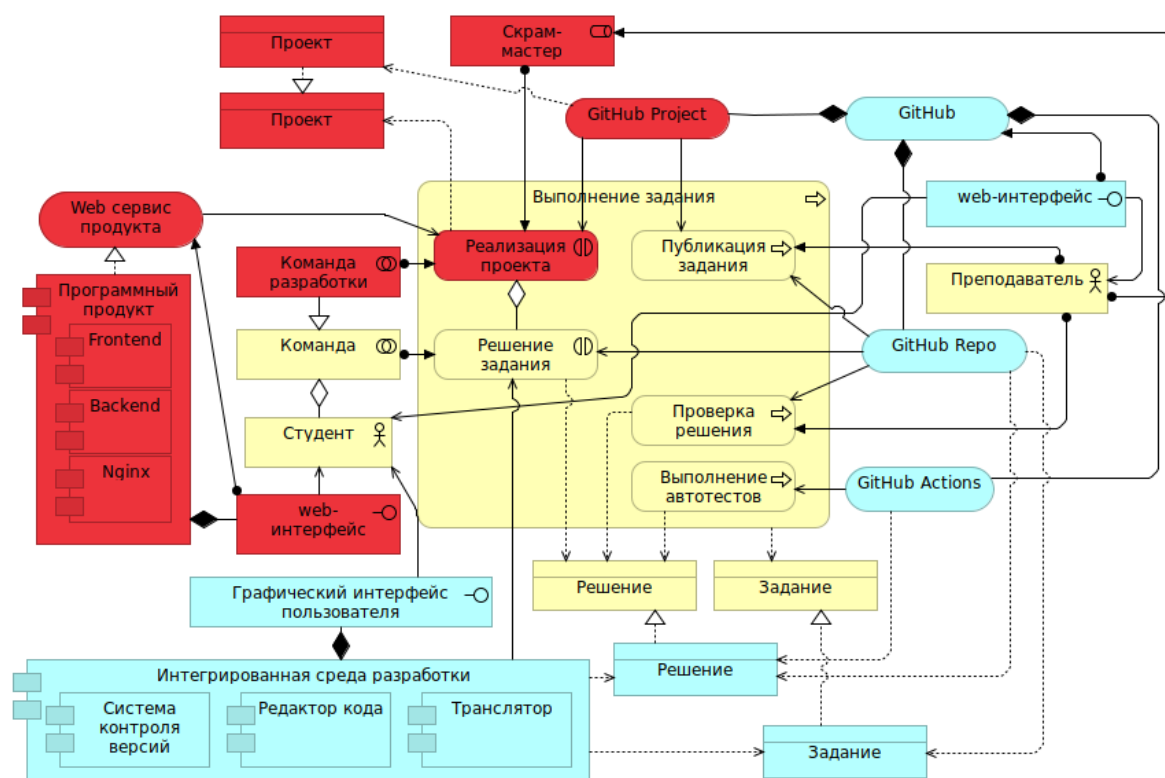


Рисунок 1.6. Представление Приложения

1.1.6. Используемая инфраструктура

Для моделирования использования инфраструктуры выбрана точка зрения Technology Usage Viewpoint, представляющая то, как приложения поддерживаются программными и аппаратными технологиями. Точка зрения используется для анализа зависимостей приложений и инфраструктуры, оценки производительности и масштабируемости. Точка зрения Technology Usage Viewpoint представляет интересы архитектора, разработчика и руководства. Представление на основе данной точки зрения разрабатывается для целей проектирования и планирования развертывания.

Разработанное представление Технологии изображено на рис. 1.7. Представление иллюстрирует, что для поддержки интегрированной среды разработки, которую студенты используют для выполнения задач, связанных с программированием, используются программные и аппаратные ресурсы персонального компьютера студента. Разрабатываемый программный продукт, представлен набором компонентов и для его поддержки необходимо арендовать виртуальную машину, установить на нее операционную систему Ubuntu Server, платформу для контейнеризации приложений Docker. Виртуальная машина с сервисом Docker daemon, представляет собой узел Сервер

приложений, который позволит развернуть и поддерживать разрабатываемый программный продукт. Программный продукт будет реализовывать некоторый web-сервис, построенный на клиент-серверной архитектуре, соответственно основными компонентами продукта являются клиентская часть, обозначенная как Frontend и серверная часть, обозначенная как Backend. Программный продукт должен быть развернут на двух стендах - продуктивном, основной сервис для конечного пользователя, и тестовом, сервис, на котором студенты могут отлаживать и тестировать доработки продукта. Для поддержки двух стендов компоненты Frontend и Backend должны содержать продуктивную и тестовую версию, кроме того продукт должен включать в себя обратный прокси сервер, для маршрутизации клиентских запросов между стендами и компонентами. Все компоненты программного продукта будут использоваться в виде Docker-контейнеров, то есть на узле Сервер приложений достаточно установить платформу Docker.

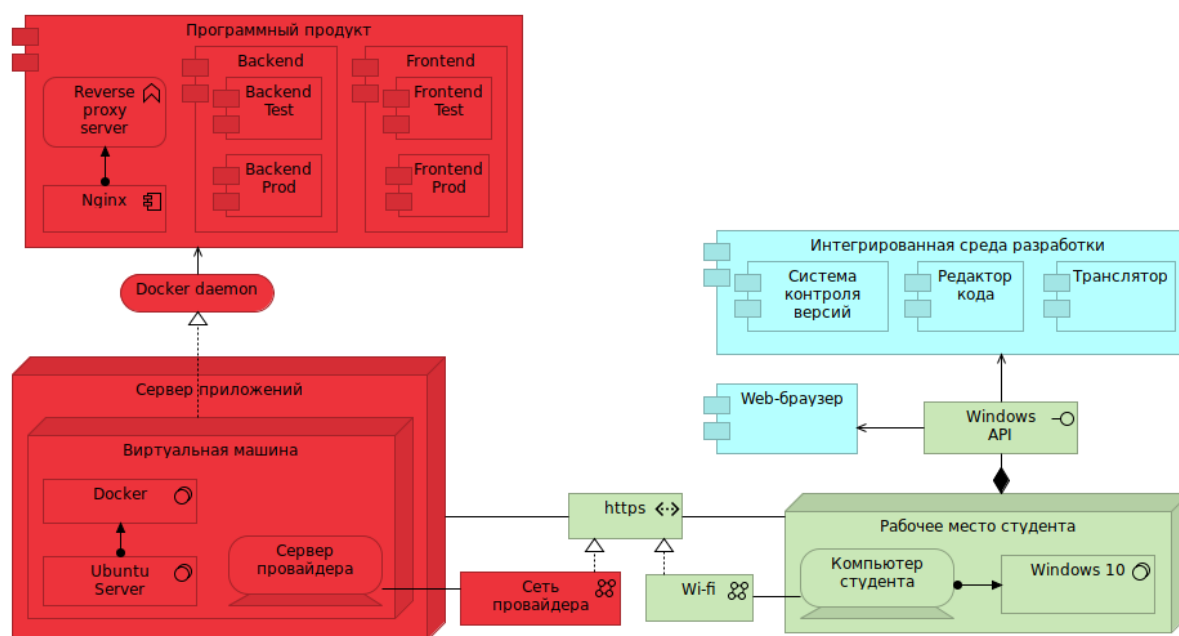


Рисунок 1.7. Представление Технологии

1.1.7. Требования к разрабатываемому продукту

Ожидаемым результатом изменений является разработка программного продукта, который можно использовать в учебном процессе. Студенты могут реализовывать проекты по разработке данного продукта, приобретая тем самым навыки проектной и командной работы и опыт разработки реального программного продукта. Необходимо

сформулировать требования к разрабатываемому продукту, с точки зрения его использования в учебных целях.

Продукт должен быть достаточно прост, чтобы не студентам не требовалось затрачивать усилия на изучение предметной области. В то же время продукт должен быть легко расширяем, чтобы студенты могли легко добавить новую функциональность. Продукт должен представлять собой web-приложение, доступное через браузер и не требующее регистрации или установки, чтобы любой студент мог непосредственно видеть результаты реализации проекта. Важно использовать методологию DevOps для организации системы автоматического тестирования, сборки и обновления продукта, чтобы студенты могли сосредоточиться на решении задач расширения функционала.

Учитывая изложенные требования, а также то, что продукт планируется использовать в курсе «Алгоритмы и структуры данных», предлагается в качестве программного продукта реализовать web-приложение онлайн калькулятор для различных алгоритмов. онлайн калькулятор будет содержать каталог алгоритмов и предоставлять возможность выполнить алгоритм с заданными параметрами и получить результат. Расширение функционала онлайн калькулятора будет заключаться в добавлении новых алгоритмов. В курсе «Алгоритмы и структуры данных» часть практических заданий предполагают программную реализацию определенного алгоритма. Такие задания можно модифицировать, чтобы требовалось реализовать не алгоритм сам по себе, а разработать программный модуль с данным алгоритмом для онлайн калькулятора. При этом от студента не будут требоваться навыки web-разработки, для добавления алгоритма будет достаточно декларативно его описать и разработать функцию, реализующую заданный алгоритм. Задания курса, предполагающие программную реализацию выполняются на языке Python, соответственно серверная часть онлайн калькулятора должна разрабатываться на одном из web-фреймворков для языка Python.

Для развертывания онлайн калькулятора необходимо два стенда - продуктовый и тестовый. Обновление стендов должно автоматически происходить при изменении исходного кода в репозиториях. Соответственно в репозиториях с исходным кодом для клиентской и серверной частей должны быть продуктовые и тестовые ветки. При изменении исходного кода в продуктовой ветке должен автоматически обновляться продуктовый стенд, при изменении в тестовой ветке должен обновляться тестовый.

Тестовый стенд будет использоваться для отладки и тестирования доработок студентов, изменения для продуктового стенда будут вноситься после успешной проверки выполненных заданий. Вносимые изменения необходимо проверять на выполнение автоматических тестов, наличие ошибок и проводить автоматический анализ качества кода.

Также для организации разработки онлайн калькулятора необходимо выбрать облачную систему управления проектом и хостинг для исходного кода и документации проекта. В ходе анализа материалов изучены статьи на тему использования различных хостингов для программных проектов в образовательном процессе. В работах [4], [20], [21] описывается использование GitHub при проведении занятий со студентами, в работе [22] описывается использование платформы GitLab. Также рассмотрены работы [23], [24], [25], в которых производится сравнение систем контроля версий и платформ для хостинга проектов и работа [26], в которой рассматриваются возможности сервиса GitHub Actions по организации непрерывной интеграции программного кода в рамках методологии DevOps. В результате проведенного анализа можно сделать вывод о целесообразности использования платформы GitHub для разработки онлайн калькулятора, так как GitHub уже используется для проведения занятий и имеет возможности по управлению проектом, хостингу документации и исходного кода и настройкам автоматизации DevOps процессов. Исходный код и документацию можно хранить в репозиториях GitHub, автоматизацию тестирования, сборки и обновления можно настроить с помощью сервиса GitHub Actions, а управление проектом осуществлять с помощью сервиса GitHub Projects. Сервис GitHub Projects позволяет создавать проекты, связанные с GitHub репозиториями, создавать в рамках проекта задачи, планировать и контролировать их выполнение. Участников проекта можно объединять в команды, назначать командам задачи, настраивать групповые политики доступа к ресурсам проекта. Задачи проекта можно выводить на разные представления - канбан-доску, таблицу или дорожную карту, строить различные диаграммы и графики, отображающие ход выполнения проекта. Таким образом, все этапы проекта по разработке онлайн калькулятора можно реализовывать на единой облачной платформе GitHub.

С учетом изложенного, можно сформулировать требования к разрабатываемому программному продукту:

1. Продукт должен представлять собой web-приложение онлайн калькулятор алгоритмов.
2. В начальной версии онлайн калькулятор должен содержать реализацию не менее трех алгоритмов, демонстрирующих его работу.
3. Исходный код продукта должен быть расположен в публичных репозиториях GitHub с лицензией MIT.
4. Управление проектом разработки продукта должно осуществляться с помощью сервиса GitHub Projects.
5. Серверная часть приложения должна разрабатываться на языке программирования Python с использованием одного из web-фреймворков.
6. Студенты должны иметь возможность добавлять в онлайн калькулятор новые алгоритмы, посредством внесения изменений в исходный код серверной части приложения.
7. онлайн калькулятор должен быть развернут на двух стендах - продуктивном и тестовом.
8. В репозиториях с исходным кодом с помощью сервиса GitHub Actions, должна быть настроена система автоматического тестирования, анализа кода, сборки и обновления стендов.

1.2. Анализ существующих онлайн калькуляторов

Для выявления основных характеристик онлайн калькуляторов проводится анализ существующих аналогов.

Первый рассматриваемый аналог - онлайн калькулятор Math [27], включающий в себя 18 направлений калькуляторов, среди которых: «Высшая математика»; «Методы решения СЛАУ»; «Аналитическая геометрия»; «Матричный калькулятор»; «Теория вероятностей»; «Эконометрика онлайн»; «Статистика онлайн»; «Сетевая модель»; «Информатика онлайн»; «Математические методы в психологии»; «Вычислительная математика»; «Методы оптимизации»; «Динамическое программирование»; «Линейное программирование»; «Исследование операций»; «Системы массового обслуживания»; «Теория игр онлайн»; «Теория автоматического управления». Каждое из направлений включает в себя ряд подразделов, содержащих

встроенные онлайн функции для решения отдельных типов задач. Так, например, направление «Высшая математика» включает в себя подраздел «Линейная алгебра», в котором доступны калькуляторы для решения задач данного раздела, например, вычисление определителя матрицы. Каждый отдельный калькулятор Math.semestr содержит инструкцию пользователя и теоретические материалы по теме решаемой задачи. Также на сайте имеются разделы для предоставления обратной связи в виде комментариев к тому или иному калькулятору. Данный онлайн калькулятор можно использовать для проверки своего решения по многим математическим и экономическим дисциплинам. Чтобы не вводить много данных, можно использовать вставку из Excel и Word. Результат решения может быть экспортирован в формате Word и Excel (при необходимости). Экспортированный отчет будет содержать ход решения с комментариями, а также исходные формулы и выводы.

Следующий рассматриваемый аналог - онлайн калькулятор Calculat.org [28]. Данный калькулятор содержит множество разделов для выполнения различного рода расчетов: «Площадь и периметр»; «Объем и площадь»; «Проценты»; «Тройное правило»; «Уравнения»; «Среднее арифметическое»; «Степени и корни»; «Тригонометрические функции»; «Логарифмы»; «Перевод единиц»; «Энергия и топливо»; «ИМТ калькулятор». Перечисленные разделы содержат подразделы, которые направлены на решения конкретных задач. Так, например, калькуляторы раздела «Площадь и периметр» выполняют расчеты площади и периметра плоских геометрических фигур, а также некоторые другие вычисления, такие как определение длины диагоналей, внутренних углов, высоты, и т.д. Каждый подраздел калькулятора содержит теоретическую информацию о решаемой задаче. Помимо этого, приводятся используемые в ходе решения формулы и графики с подробными пояснениями.

Третий анализируемый аналог - онлайн калькулятор Geleot [29], позволяющий производить расчеты по следующим направлениям: «Учеба и наука» (математика); «Техника» (ИТ; автомобили; навигация); «Конвертеры» (основные; общие; инженерные); «Строительство» (кубатура досок; кубатура бревен; вес древесины). Каждый раздел калькулятора содержит ряд подразделов для решения разных типов задач. Разделы данного онлайн калькулятора снабжены не только теоретической информацией, но и интерактивными калькуляторами, которые позволяют производить расчеты. Важно

отменить, что на данный момент сайт проходит открытое бета-тестирование, доступны следующие разделы: Математика, Конвертеры.

Четвертый аналог - онлайн калькулятор OnlineMSchool [30], содержащий такие разделы, как: упражнения; калькуляторы; справочник; таблицы и формулы; заказать решение. В разделе Калькуляторы представлены онлайн калькуляторы для решения таких задач по математике, как: «Конвертеры величин»; «Теория чисел»; «Дроби»; «Проценты»; «Решение уравнений»; «Графики»; «Прогрессии»; «Пределы и производные функций»; «Интегралы»; «Комбинаторика. Теория вероятности»; «Комплексные числа»; «Векторы»; «Матрицы»; «Аналитическая геометрия. Декартовы координаты»; «Площадь геометрических фигур»; «Периметр геометрических фигур»; «Объем геометрических фигур»; «Площадь поверхности геометрических фигур»; «Простой калькулятор». Для получения решения интересующей задачи необходимо выбрать подходящий калькулятор и ввести данные, далее программа найдет ответ и выдаст детально расписанное пошаговое решение задачи. Это дает возможность не только получить результат, но и научиться решать математические задачи, найти и исправить ошибки в своем решении или проверить правильность своего решения. Каждый калькулятор содержит не только интерактивный калькулятор с полями ввода входных данных, но и инструкцию пользователя, а также блок теории. Существующие калькуляторы постоянно совершенствуются автором сайта и, по мере возможностей, добавляются новые. В случае отсутствия необходимого математического калькулятора или наличия идей об усовершенствовании существующих калькуляторов, есть возможность написать об этом в комментариях или отзывах.

Следующий аналог - онлайн калькулятор Zaoschnik [31], включающий такие разделы, как: «Решение уравнений»; «Калькулятор площади фигур»; «Объем фигур»; «Операции над векторами»; «Решение матриц»; «Точка, прямая, плоскость»; «Конвертеры»; «Периметр фигур». Каждый раздел онлайн калькулятора содержит ряд подразделов для решения разных типов задач. Все разделы снабжены не только теоретической информацией, но и калькуляторами, которые позволяют производить расчеты, а также включают в себя блок инструкции пользователя. Помимо наличия блоков, содержащих интерактивные калькуляторы для решения разных типов

математических задач, сервис оснащен разделом, в котором каждый желающий имеет возможность оставить комментарий.

Последний рассматриваемый аналог - онлайн калькулятор Calculatorium [32]. Данный онлайн калькулятор содержит такие разделы, как: «Математические калькуляторы»; «Конвертеры»; «Калькуляторы даты и времени»; «Работа с текстом»; «Финансовые калькуляторы»; «Интернет-маркетинг»; «Шифрование данных»; «Информатика, IT»; «Здоровье и спорт»; «Физика»; «Химия»; «Ставки на спорт, трейдинг»; «Простой калькулятор». Каждый раздел онлайн калькулятора содержит ряд подразделов для решения разных типов задач. Все разделы снабжены как теоретической информацией, так и интерактивными калькуляторами, которые позволяют производить расчеты, а также блоком, в котором можно оставить комментарий. Помимо онлайн калькуляторов, сайт также предоставляет актуальную информацию по курсам валют и криптовалют, затратах на дорогах, праздниках и значимых событиях, случившихся в этот день. Кроме того, сервис оснащен разделом для предоставления обратной связи, что обеспечивает возможность написать об ошибках или же отправить авторам сайта свои пожелания и предложения.

Далее, на основе статей [33], [34], [35], освещающих вопрос проверки и оценки юзабилити сайта, разрабатывается чек-лист, релевантный для анализа существующих аналоговых платформ онлайн калькуляторов. Обращаясь к понятию юзабилити сайта (usability - удобство использования) - это показатель того, насколько легко и удобно пользователю взаимодействовать с интерфейсом сайта. Пользователю должно быть понятно, как найти нужную информацию, как не запутаться в страницах и как выполнить целевые действия. Перечень критериев, используемых для сравнительного анализа онлайн калькуляторов (табл. 1.1), составлен, основываясь на существующих чек-листах юзабилити сайтов и возможностях рассмотренных онлайн калькуляторов. Для проверки скорости сайтов использовалась платформа [36], содержащая инструмент для оценить того, как быстро веб-сайт загружается в браузере у пользователя. Оценка дизайна сайта выполнялась посредством выставления экспертных оценок участниками проектной команды с последующим выявлением средней оценки для каждого рассмотренного онлайн калькулятора.

Таблица 1.1. Сравнительный анализ онлайн калькуляторов

№	Критерий сравнения / Онлайн калькулятор	Math	Calculat.org	Geleot	OnlineMSchool	Zaochnik	Calculatorium
1	Возможность использовать вставку из word / excel для ввода данных	+	-	-	-	-	-
2	Возможность экспорта решения в word / excel	+	-	-	-	-	-
3	Наличие разбиения на разделы и подразделы (от широких тем к более узким)	+	+	+	+	+	+
4	Наличие теоретических материалов	+	+	+	+	+	+
5	Наличие инструкции пользователя	+	-	-	+	+	-
6	Наличие простого калькулятора	-	-	-	+	-	+
7	Наличие контактов авторов сайта или службы поддержки (телефон / e-mail)	+	+	-	+	+	-
8	Наличие раздела обратной связи / комментариев непосредственно на сайте	+	-	+	+	+	+
9	Наличие поиска по сайту	+	-	+	+	+	+
10	Возможность выбора языка	-	+	-	+	-	-
11	Адаптивность под мобильные устройства	+	+	+	+	+	+
12	Скорость загрузки сайта (для компьютера):						
12.1	First Contentful Paint (сек)	0,3	0,3	0,3	0,7	0,6	0,5
12.2	Время загрузки достаточной части контента (сек)	0,3	0,3	0,3	0,7	0,6	0,6
12.3	Speed Index (сек)	0,4	0,6	0,7	0,7	0,7	1,3
12.4	Time to Interactive (сек)	0,3	1,0	1,4	1,1	1,7	2,1

№	Критерий сравнения / Онлайн калькулятор	Math	Calculat.org	Geleot	OnlineMSchool	Zaochnik	Calculatorium
12.5	Total Blocking Time (мс)	0	10	90	0	70	10
12.6	Cumulative Layout Shift	0,006	0,576	0	0,215	0,022	0
13	Скорость загрузки сайта (для телефона):						
13.1	First Contentful Paint (сек)	0,9	1,0	1,9	1,1	2,0	2,3
13.2	Время загрузки достаточной части контента (сек)	0,9	1,0	1,9	1,1	2,0	2,3
13.3	Speed Index (сек)	1,2	1,9	2,3	3,2	3,2	4,7
13.4	Time to Interactive (сек)	0,9	5,2	5,7	6,7	7,6	8,9
13.5	Total Blocking Time (мс)	0	490	370	480	810	720
13.6	Cumulative Layout Shift	0,001	0,581	0	0,022	0	0
14	Дизайн сайта (цвета и шрифты). Используется шкала от 1 до 5	3	4	4	3	4	4

Таким образом, для создания информативного и удобного сайта, содержащего онлайн калькуляторы, важен не только объем информации (количество разнообразных типов решаемых задач), но и дизайн и структура сайта, а именно - грамотно подобранные цвета и шрифты, удобная навигация по сайту, в том числе наличие поиска, удобные форматы ввода и вывода данных, а также другие критерии, входящие в чек-листы юзабилити сайта. Основываясь на проведенном анализе существующих онлайн калькуляторов, в проекте предлагается реализовать:

1. Высокую скорость загрузки сайта для использования с компьютера;
2. Удобный и красивый интерфейс, включающий:
 - a. Разбиение на разделы;
 - b. Навигационные элементы для переключения между разделами;
 - c. Возможность сбора обратной связи;
 - d. Грамотное сочетание цветового оформления сайта и шрифтов;
3. Наличие теоретической информации для каждой отдельной задачи;
4. Наличие руководства пользователя для студентов и преподавателей.

В перспективе может быть выполнено следующее: адаптация сайта под мобильные устройства и обеспечение высокой скорости загрузки с телефона / планшета; при наличии большого числа алгоритмов вычисления - добавление поиска по сайту и группировка решаемых задач в более широкие темы путем разбиения сайта на разделы и подразделы (от широких тем к более узким) с добавлением удобных навигационных элементов для переключения между разделами и подразделами; добавление возможности выбора языка.

1.3. Предметная область реализации онлайн калькуляторов

Решение любой вычислительной задачи в процессах обработки информации всегда сводится к применению алгоритмов. Со времен появления математических наук алгоритмы использовались в простейших операциях: сложение, вычитание, умножение, деление, построение фигур и других расчетов. Научное понятие алгоритма в трудах Алферовой З.В. приводится как «конечная совокупность точно сформулированных правил решения некоторого класса задач» [37]. К примеру, для задачи поиска наибольшего общего делителя для двух натуральных чисел можно применить алгоритм Евклида. Суть его в том, что из большего числа вычитается меньшее число до тех пор,

пока наименьшее число не станет равным нулю. Наибольшее число после всех итераций вычитаний - это и будет наибольшим делителем.

Говоря об онлайн калькуляторах, в основе их расчетов также лежат алгоритмы - правила и последовательность выполнения действий вычислительных операций. При использовании онлайн калькуляторов пользователь может выполнять небольшой набор действий: посмотреть перечень существующих расчетов (они же алгоритмы), запускать выбранные расчеты для решения конкретной задачи, задавать входные параметры и получать результаты вычислений. На примере онлайн калькулятора math.semestr.ru, пользователь может выбрать задачу по теории вероятности, например, нахождение вероятности, что из корзины с n-шарами будет извлечено k-шаров определенного цвета. Для решения задачи ему достаточно указать количество шаров в корзине, количество цветов определенного цвета, общее количество извлеченных шаров из корзины и количество шаров конкретного цвета для расчета вероятности, что именно они будут входить в число извлеченных шаров. В результате выполнения расчета вероятности, пользователь может выгрузить ответ в файл формата Word и посмотреть по какой формуле (алгоритму) был получен ответ. На рис. 1. приведена диаграмма понятий предметной области реализации онлайн калькуляторов.

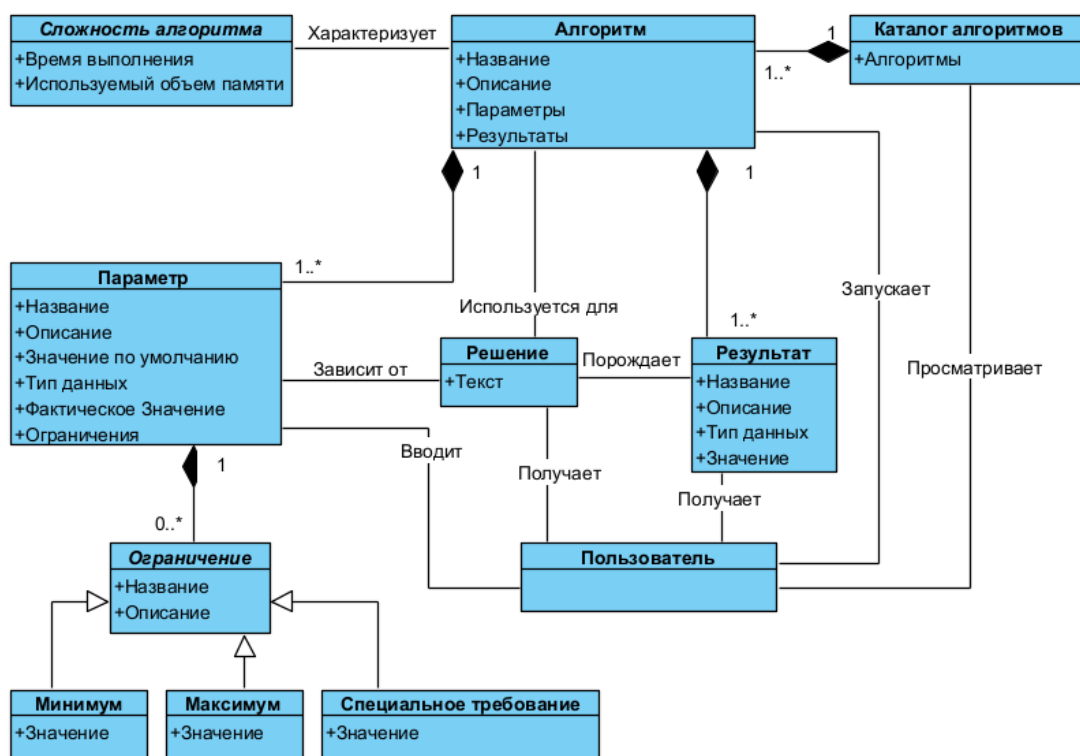


Рисунок 1.8. Диаграмма понятий предметной области

Для понятия алгоритм можно выделить атрибуты название и описания (что выполняет и какую задачу решает). Для выполнения алгоритма необходимо задавать его входные параметры и после выполнения получить результаты (конечный ответ). Каждый алгоритм характеризуется сложностью, описывающей сколько времени и сколько вычислительного объема памяти требуется на выполнение. Например, рекурсивный расчет чисел Фибоначчи имеет экспоненциальную сложность $O(2^n)$ и с каждым увеличением параметров требует большего объема памяти и времени на выполнение. Соответственно, обработка подобных алгоритмов может приводить к ошибкам при недостаточной вычислительной мощности компьютера или сервера.

Параметры являются частью алгоритма и имеют описание, тип данных (число, строка, массив и другие типы данных), фактическое значение, кроме того, значение параметра может быть задано по умолчанию. На значения параметров для конкретного алгоритма могут накладываться ограничения - требования или условия, которые ожидает алгоритм для успешной обработки. Ограничением могут быть условия к минимальным, максимальным значениям для чисел, а также специальные требования. К примеру, алгоритм Дейкстры для поиска кратчайшего пути требует, чтобы в графе не было ребер с отрицательным весом, поэтому данное условие считается специальным требованием для конкретного алгоритма.

В зависимости от набора используемых параметров и задачи, которую решает алгоритм, может быть сформировано решение, т.е. текстовое описание последовательности действий, которые привели к конечному результату. Пользователь в данной модели рассматривается как отдельный объект для демонстрации связей между его сущностью и другими объектами понятия алгоритма. В рамках диаграммы, пользователь может просматривать каталог с перечнем доступных алгоритмов в онлайн калькуляторе, вводить параметры для выбранного алгоритма и запускать алгоритм на выполнение. После выполнения алгоритма он получает результат и решение - вывод описания последовательности работы алгоритма. На этапе формирования требований к онлайн калькулятору необходимо учесть особенности рассмотренных сущностей и их атрибутов в разрезе понятия алгоритма в онлайн калькуляторах.

Для всех рассмотренных в предыдущих разделах онлайн калькуляторов, перечень возможных действий пользователей одинаковый: посмотреть перечень доступных

расчетов, выбрать и запустить на выполнение онлайн калькулятор по выбранному расчету, указать входные параметры для обработки и выгрузить при необходимости результаты. При необходимости пользователь может ознакомиться с теоретическими материалами по приведенным расчетам. Перечисленных возможностей будет достаточно для определения требований к разрабатываемому онлайн калькулятору. На рис. 1.9 приведена USE-CASE диаграмма возможностей пользователей в онлайн калькуляторе.

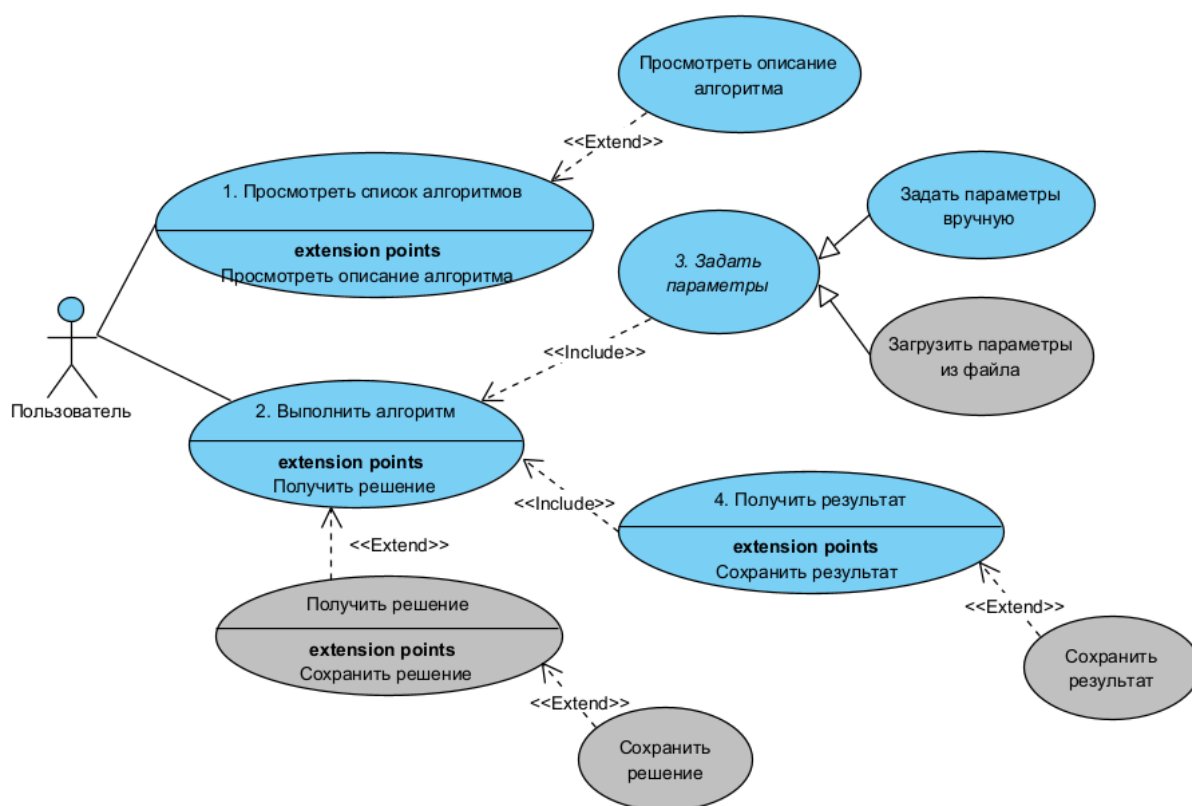


Рисунок 1.9. Диаграмма вариантов использования онлайн калькуляторов

Голубым цветом обозначены обязательные функции, которые будут учтены при формировании требований к системе, а серым цветом - необязательные функции и модули, не влияющие на общий процесс работы онлайн калькулятора. Пользователь - это основной актер, пользующийся системой онлайн калькулятора. Он может просматривать список доступных алгоритмов или запускать на выполнение выбранный алгоритм. При просмотре списка алгоритмов Пользователь может ознакомиться с описанием работы и назначения алгоритма, поэтому на диаграмме этот процесс расширяет функцию «Просмотреть список алгоритмов». Данная возможность будет учтена при формировании требований к системе.

Выполнение алгоритма обязательно включает в себя функцию заполнения входных параметров. Под указанием входных параметров подразумевается, что пользователь заполняет их вручную или загружает из файла с готовым набором данных. Последнее является необязательной функцией, поскольку эта возможность может быть полезна только если алгоритм требует большого объема входных данных, а заведение их вручную будет трудозатратной операцией.

Функции «Получить решение» и «Получить результат» включают в себя функцию выполнения выбранного алгоритма, поскольку они не могут существовать как отдельные и независимые модули. При этом возможность получения решения является необязательной функцией, т.к. она может быть только в алгоритмах, в которых используются расчеты с применением разных формул (от 2 и более). Если алгоритм не предусматривает больших расчетов по нескольким формулам, то системе достаточно вывести результат без вывода всех последовательностей действий.

Возможности «Сохранить решение» и «Сохранить результат» будут необязательными функциями в рамках разрабатываемого онлайн калькулятора. Сохранение решения или результата может быть полезно только в том случае, если они являются достаточно содержательными для включения, например, в отчетность или иной документ пользователем. В текущей работе алгоритмы онлайн калькулятора будут только выводить результат выполнения. Возможно в перспективе, данные функции будут добавлены как дополнительные модули работы системы. В приложении А. приведены спецификации вариантов использования.

1.4. Требования к онлайн калькулятору

В рамках работы по созданию open-source проекта для решения учебных практикоориентированных задач программной инженерии разрабатываются требования, которым должен соответствовать проект. Требования приведены в подразделах текущего параграфа работы.

1.4.1. Функциональные требования

1. онлайн калькулятор позволяет пользователям возможность просматривать список алгоритмов.

2. онлайн калькулятор позволяет пользователям просматривать описание алгоритма, которое может содержать теоретические или исторические сведения об алгоритме.
3. онлайн калькулятор позволяет пользователям вводить данные параметров для выполнения алгоритма.
4. Для каждого параметра выбранного алгоритма пользователь может ознакомиться с информацией о типе данных и ограничениях для вводимых значений.
5. онлайн калькулятор позволяет пользователям запускать выполнение алгоритма с заданными параметрами.
6. онлайн калькулятор позволяет пользователям просматривать результаты выполнения алгоритма с заданными параметрами.

1.4.2. Требования к программной документации

Программная документация должна включать в себя:

1. Техническое задание;
2. Исходный код;
3. Руководство пользователя;
4. Руководство администратора.

1.4.3. Требования к эргономике

Отображение онлайн калькулятора для пользователей должно быть адаптивным для разрешения экрана 1920*900, для просмотра на персональном компьютере. При загрузке системы с мобильных устройств должна выводиться статическая страница с информацией о недоступности системы с мобильных устройств.

1.4.3. Требования к хостингу

Хостинг для размещения онлайн калькулятора должен иметь следующее программное обеспечение: - операционная система Ubuntu 20.04 LTS, - docker v20.

1.4.4. Требования к информационной и программной совместимости

Исходный код разрабатывается, руководствуясь стандартами W3C. Для реализации статических страниц должны использоваться языки HTML 5 и CSS 3. Для реализации

интерактивных элементов клиентской части должны использоваться языки HTML и JavaScript, с использованием одного из актуальных Фреймворков, таких как React, Angular, Vue. Для реализации серверной части должен использоваться язык один из web фреймворков для языка программирования Python (FastApi, Flask, Django).

1.4.5. Требования к клиентскому программному обеспечению

онлайн калькулятор должен быть доступен для полнофункционального просмотра с помощью следующих браузеров:

1. Mozilla Firefox, начиная с версии 78;
2. Google Chrome, начиная с версии 98;
3. Apple Safari, начиная с версии 15;
4. Yandex Browser, начиная с версии 21.11.

1.4.6. Требования к маркировке и упаковке

Разработанный продукт передается заказчику в виде ссылок на публичный репозиторий GitHub на сайте <https://github.com> с исходными кодами всех программных модулей. Учитывая изложенное, требования по маркировке и упаковке не предъявляются.

1.4.7. Прочие требования

онлайн калькулятор должен быть развернут в двух контурах Продуктовом и Тестовом. В репозиториях с исходным кодом должна быть настроена система автоматического тестирования, сборки и развертывания продукта на продуктовый и тестовый контуры.

1.5. Прототип системы

Текущий раздел работы содержит описание прототипа сайта онлайн калькулятора, разработанного при использовании онлайн-сервиса Figma.

1.5.1. Главная страница

Главная страница сайта состоит из нескольких частей: шапка сайта, основная часть, боковое меню, ссылки на социальные сети, подвал сайта (рис. 1.10).

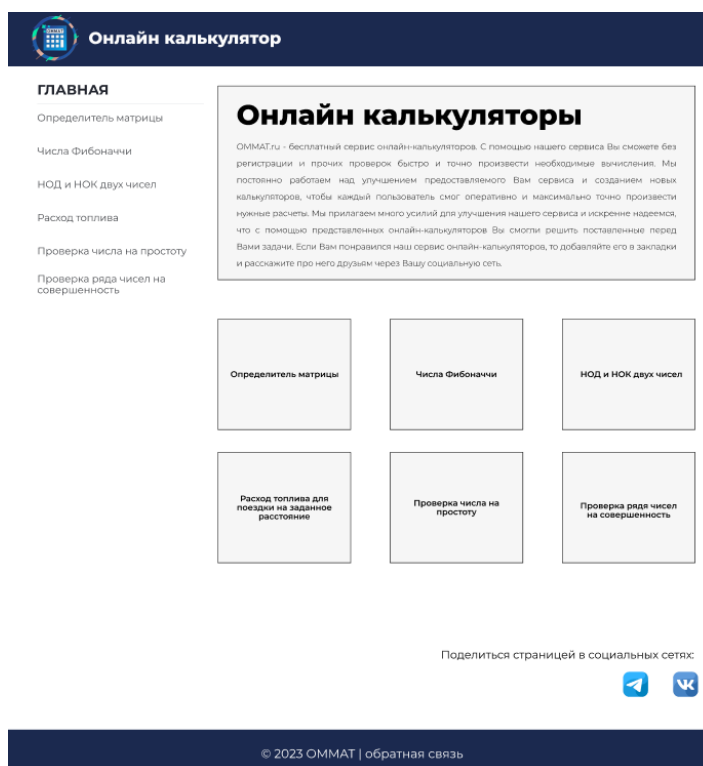


Рисунок 1.10. Главная страница

Шапка сайта - элемент, расположенный в верхней части страницы каждого раздела сайта, - содержит логотип онлайн калькулятора ОММАТ (см. рис. 1.11), разработанный участниками проектной команды, и кликабельный заголовок «Онлайн калькулятор», при нажатии на который будет осуществлен переход на главную страницу сайта.



Рисунок 1.11. Логотип онлайн калькулятора ОММАТ

В левой части страницы представлен блок меню, в котором представлены все разделы сайта. На данный момент блок меню состоит из 7 разделов, в том числе раздел

«Главная». При нажатии на тот или иной элемент меню осуществляется переход на соответствующую страницу сайта.

В основной части главной страницы приведено описание продукта, как им пользоваться и его преимущества. Под описанием продукта расположены отдельные ячейки с разделами сайта, при нажатии на любую ячейку с названием раздела осуществляется переход на соответствующую страницу сайта.

Внизу главной страницы с правой стороны расположен раздел с ссылками на социальные сети. Заголовок раздела: «Поделиться страницей в социальных сетях:». Ссылки оформлены в виде кнопок со значками популярных социальных сетей, в данном случае это «Telegram» и «VK». При нажатии на одну из кнопок пользователь перенаправляется на соответствующую страницу, где он может опубликовать ссылку на калькулятор в своей ленте новостей или поделиться ею с друзьями.

Подвал сайта содержит информацию о годе создания сайта и об авторских правах группы разработки сайта - «© 2023 ОММАТ». Кроме того, в подвале сайта размещен элемент «обратная связь», позволяющий пользователю связаться с командой поддержки или отправить сообщение на электронный адрес support@ommat.ru о проблеме, возникшей в процессе использования калькулятора.

1.5.2. Раздел «Определитель матрицы»

В основной части раздела «Определитель матрицы» (см. рис. 1.12) приводится информационный блок, в котором описано, что именно вычисляет данный раздел калькулятора. Далее раздел «Определитель матрицы» предоставляет пользователям возможность вычислить детерминант матрицы. Далее представлен блок для ввода входных данных - размер матрицы и сама матрица. В данном разделе пользователь вводит размер матрицы из диапазона чисел от 1 до 9. После выбора размера матрицы, пользователь может самостоятельно ввести числа матрицы. Однако, существует ограничение на ввод чисел: каждое число в матрице должно быть не больше 100. После ввода матрицы, пользователь может нажать на кнопку «Получить результат», расположенную под матрицей. В результате на экране появится блок ответа, который будет содержать значение определителя матрицы.

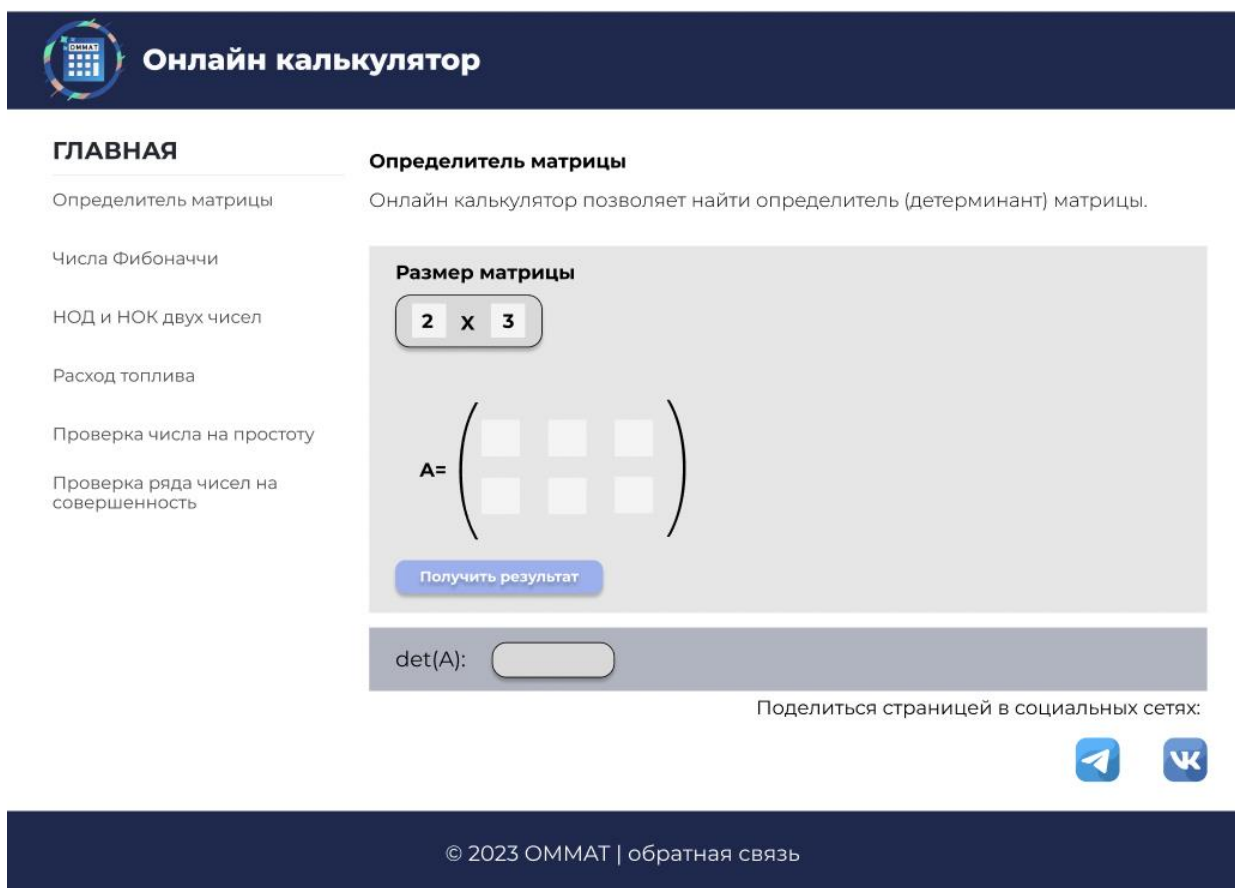



Рисунок 1.12. Раздел «Определитель матрицы»

1.5.3. Раздел «Числа Фибоначчи»

В основной части раздела «Числа Фибоначчи» (см. рис. 1.13) приводится блок теоретической информации, а именно приводится определение «Числа Фибоначчи» и инструкция, что именно нужно вводить пользователю. Далее представлен блок ввода числа. Пользователь может ввести значение n -го члена, для которого необходимо сформировать последовательность. Однако, существует ограничение на ввод числа: n -ый член должен быть не больше 1000. После ввода значения n -го члена пользователь может нажать на кнопку «Сформировать последовательность», расположенную под полем ввода. В результате на экране появится блок ответа, который будет содержать последовательность чисел Фибоначчи до n -го члена. Последовательность Фибоначчи выводится на экран по порядку, начиная с первого числа и заканчивая n -ым числом. Таким образом, пользователь может увидеть последовательность чисел Фибоначчи, которая формируется до заданного n -го члена.




Онлайн калькулятор

ГЛАВНАЯ
Определитель матрицы
Числа Фибоначчи
НОД и НОК двух чисел
Расход топлива
Проверка числа на простоту
Проверка ряда чисел на совершенность

Числа Фибоначчи
Числа Фибоначчи - последовательность чисел, каждый член которой равен сумме двух предыдущих.
Введите n-ый член, для которого надо сформировать ряд Фибоначчи, и калькулятор выдаст вам последовательность до n-го члена.

Длина последовательности

Последовательность чисел Фибоначчи:

Поделиться страницей в социальных сетях:



© 2023 ОММАТ | обратная связь

Рисунок 1.13. Раздел «Числа Фибоначчи»

1.5.4. Раздел «НОД и НОК двух чисел»

В основной части раздела «НОД и НОК двух чисел» (см. рис. 1.14) приводится блок теоретической информации о рассматриваемой математической задаче - приводятся определения НОД и НОК. Далее представлен блок для ввода входных данных - двух чисел, для которых необходимо рассчитать НОД и НОК. Ввод чисел должен быть осуществлен с клавиатуры в соответствующие элементы сайта: первое число вводится в поле под заголовком «Число 1», второе число - в поле под заголовком «Число 2». В данном разделе предполагается ввод целых чисел в качестве входных данных, ограничение - каждое число должно содержать до 20 символов. По завершении ввода входных данных, при нажатии кнопки «Получить результат» решение будет выведено в блок ответа, который располагается непосредственно под блоком ввода входных данных. Формат вывода результата (выходных данных) соответствует формату ввода входных данных - имеются заголовки и соответствующие им элементы, в которые выводятся результаты вычислений. Расчетное значение наибольшего общего делителя выводится в

элемент, расположенный справа от заголовка «НОД:», а расчетное значение наименьшего общего кратного - в элемент, который расположен справа от заголовка «НОК:».

The screenshot shows the 'Онлайн калькулятор' (Online Calculator) website. The main navigation menu on the left includes: Главная (Home), Определитель матрицы (Matrix Determinant), Числа Фибоначчи (Fibonacci Numbers), НОД и НОК двух чисел (GCD and LCM of two numbers), Расход топлива (Fuel Consumption), Проверка числа на простоту (Prime Number Check), and Проверка ряда чисел на совершенность (Perfect Number Sequence Check). The selected section is 'НОД и НОК двух чисел'. It contains a description of GCD and LCM, input fields for 'Число 1' and 'Число 2', a 'Получить результат' (Get Result) button, and output fields for 'НОД:' and 'НОК:'. At the bottom, there are social media sharing icons for Telegram and VK, and a footer with the copyright notice '© 2023 ОММАТ | обратная связь'.

Рисунок 1.14. Раздел «НОД и НОК двух чисел»

1.5.5. Раздел «Расход топлива для поездки на заданное расстояние»

В основной части раздела «Расход топлива для поездки на заданное расстояние» (см. рис. 1.15) приводится блок информации, повествующей о возможностях данного раздела. Далее представлен блок для ввода входных данных - трех чисел, на основе которых необходимо рассчитать расход и стоимость топлива. Ввод чисел должен быть осуществлен с клавиатуры в соответствующие элементы сайта: первое число вводится в поле, расположенное справа от заголовка «Сколько хотите проехать (км)», второе число - в поле справа от заголовка «Средний расход топлива (л/100км)», третье - в поле, которое находится справа от заголовка «Стоимость 1 л. топлива (руб)». В данном разделе возможен ввод чисел с плавающей точкой. Важно отметить, что здесь предусмотрена возможность настройки формата вывода результирующих значений - имеется возможность установки параметра округления результирующих значений до целого. Для этого необходимо поставить галочку «Округлять результат». По завершении ввода

входных данных и принятия решения о необходимости округления результата, при нажатии кнопки «Получить результат» решение будет выведено в блок ответа, который располагается непосредственно под блоком ввода входных данных. Формат вывода результата (выходных данных) соответствует формату ввода входных данных - имеются заголовки и соответствующие им элементы, в которые выводятся результаты вычислений. Расчетное значение количества топлива в литрах выводится в элемент, расположенный справа от заголовка «Потребуется топлива (л):», а расчетное значение стоимости топлива - в элемент, который расположен справа от заголовка «Стоимость топлива (руб):».

The screenshot shows a web application titled "Онлайн калькулятор" (Online Calculator) with a logo on the left. A sidebar on the left lists various calculator functions: "Определитель матрицы", "Числа Фибоначчи", "НОД и НОК двух чисел", "Расход топлива", "Проверка числа на простоту", and "Проверка ряда чисел на совершенность". The main content area is titled "Расход топлива для поездки на заданное расстояние" (Fuel consumption for a trip to a given distance). Below the title, a description states: "Калькулятор расхода топлива поможет рассчитать количество и стоимость топлива для поездки на заданное расстояние." The calculator interface includes three input fields: "Сколько хотите проехать (км)", "Средний расход топлива (л/100км)", and "Стоимость 1 л. топлива (руб)". There is a checked checkbox for "Округлять результат" (Round the result) and a "Получить результат" button. Below the input fields, the results are displayed: "Потребуется топлива (л):" and "Стоимость топлива (руб):", each followed by an empty input field. At the bottom right, there are social media sharing icons for Telegram and VK, with the text "Поделиться страницей в социальных сетях:". The footer contains the copyright notice "© 2023 ОММАТ | обратная связь".

Рисунок 1.15. Раздел «Расход топлива для поездки на заданное расстояние»

1.5.6. Раздел «Проверка числа на простоту»

В основной части раздела «Проверка числа на простоту» (рис. 1.16) приводится блок теоретической информации о рассматриваемой математической задаче - приводятся определение простого числа. Далее представлен блок для ввода входных данных - одного числа, которое требуется проверить на простоту. Ввод числа должен быть осуществлен с клавиатуры в соответствующий элемент сайта - в поле, расположенное под

заголовком «Число». В данном случае предполагается ввод целого числа. По завершении ввода входных данных, при нажатии кнопки «Получить результат» решение будет выведено в блок ответа, который располагается непосредственно под блоком ввода входных данных. Формат вывода результата (выходных данных) соответствует формату ввода входных данных - имеются заголовки и соответствующие им элементы, в которые выводятся результаты вычислений. Результат проверки числа на простоту выводится в поле, расположенное справа от заголовка «Проверка числа на простоту:». Вывод результата предполагается в формате True / False: значение «да» - в случае, если число является простым, иначе - значение «нет».

Онлайн калькулятор

ГЛАВНАЯ

- Определитель матрицы
- Числа Фибоначчи
- НОД и НОК двух чисел
- Расход топлива
- Проверка числа на простоту
- Проверка ряда чисел на совершенность

Проверка числа на простоту

Простое число - натуральное (целое положительное) число, имеющее ровно два различных натуральных делителя — единицу и самого себя. Другими словами, число x является простым, если оно больше 1 и при этом делится без остатка только на 1 и на x .

Число

Получить результат

Проверка числа на простоту:

Поделиться страницей в социальных сетях:


© 2023 ОММАТ | обратная связь

Рисунок 1.16. Раздел «Проверка числа на простоту»

1.5.7. Раздел «Проверка ряда чисел на совершенность»

В основной части раздела «Проверка ряда чисел на совершенность» (рис. 1.17) приводится блок теоретической информации о рассматриваемой математической задаче - приводятся определение совершенного числа. Также в информационном блоке приведен ожидаемый формат и пример ввода входных данных. Далее представлен блок для ввода входных данных - ряда чисел для проверки наличия совершенных чисел. Ввод входных

данных должен быть осуществлен с клавиатуры в формате ряда чисел через запятую в соответствующий элемент сайта - в поле, расположенное под заголовком «Ряд чисел для проверки на совершенность». По завершении ввода входных данных, при нажатии кнопки «Получить результат» решение будет выведено в блок ответа, который располагается непосредственно под блоком ввода входных данных. Формат вывода результата (выходных данных) соответствует формату ввода входных данных - имеются заголовки и соответствующие им элементы, в которые выводятся результаты вычислений. Результат проверки наличия в ряде совершенных чисел выводится в поле, расположенное справа от заголовка «Проверка наличия совершенных чисел:». Вывод результата в данном случае предполагается в формате True / False: значение «да» - в случае, если в ряде присутствуют совершенные числа, иначе - значение «нет». Кроме того, список найденных в ряде совершенных чисел выводится в поле, которое расположено справа от заголовка «Совершенные числа:». В случае отсутствия в ряде совершенных чисел - поле, расположенное справа от заголовка «Совершенные числа:», остается пустым.

 **Онлайн калькулятор**



ГЛАВНАЯ
Определитель матрицы
Числа Фибоначчи
НОД и НОК двух чисел
Расход топлива
Проверка числа на простоту
Проверка ряда чисел на совершенность

Проверка ряда чисел на совершенность
Совершенное число - число, равное сумме всех своих собственных делителей (то есть всех положительных делителей, отличных от самого числа).
Введите ряд чисел через запятую для проверки наличия совершенных чисел.
Пример: 4, 5, 28, 496, 6789, 5235906

Ряд чисел для проверки на совершенность

Проверка наличия совершенных чисел:
Совершенные числа:

Поделиться страницей в социальных сетях:

© 2023 ОММАТ | обратная связь

Рисунок 1.17. Раздел «Проверка ряда чисел на совершенность»

1.5.8. Контроль ввода входных данных и соответствие прототипа заявленным требованиям

Каждый раздел сайта, за исключением главной страницы, оснащен автоматической проверкой формата входных данных, которые вводятся пользователем. В случае, если формат входных данных не соответствует требованиям раздела, пользователю должно высвечиваться уведомление об ошибке (рис. 1.18).

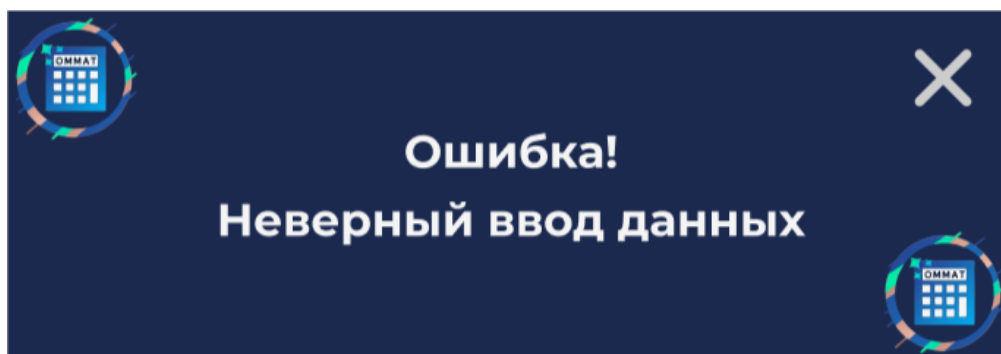


Рисунок 1.18. Уведомление об ошибке при неверном формате ввода данных

Прототип сайта, разработанный в рамках данной работы, соответствует всем обозначенным функциональным требованиям. Создание прототипа является одним из этапов разработки open-source проекта для решение учебных практикоориентированных задач программной инженерии. В процессе разработки прототипа учитывались поставленные перед проектом требования, например такие как:

1. Легкость в использовании;
2. Красивый интерфейс с грамотным сочетанием цветового оформления сайта и шрифтов;
3. Удобство навигации между разделами;
4. Возможность сбора обратной связи;
5. Наличие информационной части в каждом разделе сайта, в которую может быть включена теоретическая информация о решаемых математических задачах и примеры ввода входных данных;
6. Визуальное разделение блоков, предназначенных для ввода входных данных и вывода результатов.

Глава 2. Проектирование онлайн калькулятора

онлайн калькулятор планируется разрабатывать на клиент-серверной архитектуре. Перед проектированием клиентской и серверной частей необходимо определить способ, которым они будут взаимодействовать, то есть необходимо выполнить проектирование API (Application Programming Interface). Проектировать API предполагается с использованием технологии REST, протокола HTTP с передачей данных в формате JSON. После проектирования API необходимо выполнить проектирование серверной и клиентской частей онлайн калькулятора.

2.1. Проектирование API

На основе анализа функциональных требований к онлайн калькулятору и концептуальной диаграммы предметной области можно определить данные, которыми должны обмениваться клиентская и серверная части приложения. По запросу клиентской части серверная часть должна предоставлять список имеющихся алгоритмов. Для каждого из имеющихся алгоритмов по запросу клиентской части необходимо предоставлять описание этого алгоритма, в том числе описание наборов входных и выходных данных. По запросу, содержащему фактические значения для набора входных данных, необходимо предоставлять фактические значения для набора выходных данных, полученные в результате выполнения выбранного алгоритма.

Ресурсом, к которому API предоставляет доступ являются алгоритмы и конечная точка доступа к списку алгоритмов должна быть доступна по URI `/api/algorithms`. Получить список имеющихся алгоритмов можно посредством выполнения GET запроса к конечной точке `/api/algorithms`. Ответ должен содержать список, каждый из экземпляров которого имеет поля `name` - уникальное имя алгоритма и `title` - название алгоритма.

Доступ к конкретному алгоритму осуществляется по URI `/api/algorithms/{algorithm_name}`, с указанием уникального имени алгоритма вместо `algorithm_name`. Получить описание выбранного алгоритма можно посредством выполнения GET запроса к конечной точке `/api/algorithms/{algorithm_name}`. Ответ должен содержать описание алгоритма в поле `result` или информацию об ошибке в поле `errors`. Описание алгоритма должно содержать поля `name` - уникальное имя алгоритма и

title - название алгоритма, description - описание алгоритма, parameters - список описаний входных данных, outputs - список описаний выходных данных. Экземпляр входных или выходных данных можно описать структурой одного типа, содержащей поля name - уникальное имя параметра и title - название параметра, description - описание параметра, data_type - тип данных параметра, data_shape - размерность параметра и default_value - значение по умолчанию.

Получить результат выполнения алгоритма можно посредством выполнения POST запроса к конечной точке /api/algorithms/{algorithm_name}, с передачей фактических значений для набора входных параметров, то есть списка объектов с полями name - уникальное имя параметра и value - фактическое значение параметра. Ответ должен содержать список фактических значений выходных данных в поле result или информацию об ошибке в поле errors. Для описания фактических значений как входных, так и выходных данных можно использовать одну структуру с полями name и value.

Примеры запросов с указанием описанных конечных точек представлены в таблице 2.1. Для проектирования API использован инструмент моделирования Visual Paradigm. В UML-модель, разработанную на этапе анализа, добавлена диаграмма классов, представляющая доступные по API ресурсы, а также конечные точки API для получения ресурсов. Разработанная диаграмма классов представлена на рисунке 2.1.

Таблица 2.1. Сравнительный анализ онлайн калькуляторов

№	HTTP-запрос	Конечная точка	Действие
1	GET	/api/algorithms	Получить список имеющихся алгоритмов
2	GET	/api/algorithms/fibonacci	Получить описание алгоритма fibonacci
3	POST	/api/algorithms/fibonacci	Выполнить алгоритм fibonacci

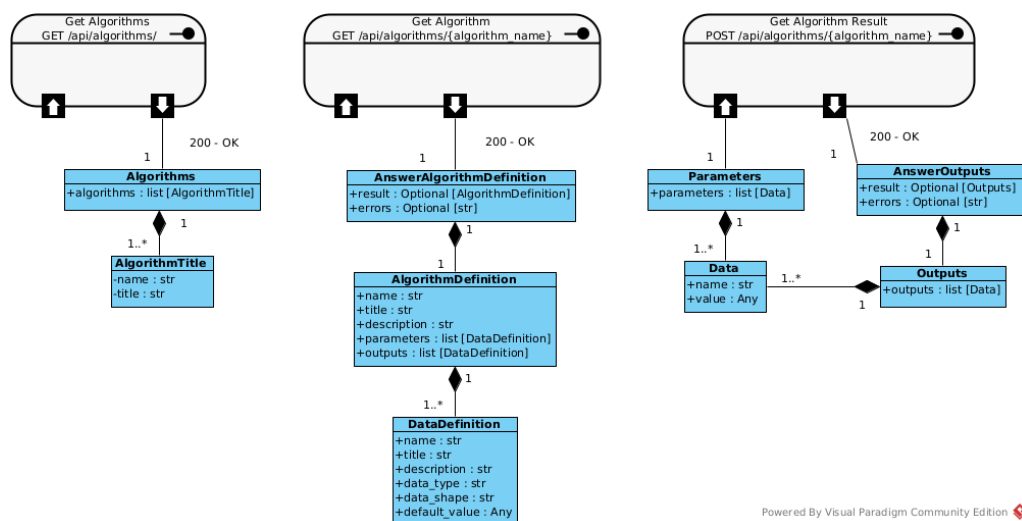


Рисунок 2.1. Диаграмма классов для API

2.2. Проектирование серверной части онлайн калькулятора

2.2.1 Проектирование структуры серверной части онлайн калькулятора

Согласно техническому заданию, разрабатывать серверную часть необходимо на языке программирования Python. API для онлайн калькулятора предельно прост, содержит всего три конечных точки, взаимодействие с базой данных или другими приложениями не предусмотрено. С учетом изложенного для разработки серверной части онлайн калькулятора решено использовать web-фреймворк FastAPI. FastAPI является проектом с открытым исходным кодом, позволяет разрабатывать API на основе открытых стандартов OpenAPI, автоматически генерировать документацию для API. FastAPI основан на python библиотеке Pydantic, что позволяет использовать иерархические модели данных Pydantic, валидаторы данных и документирование схемы данных в виде JSON Schema, для объектов любой глубины вложенности. Кроме того, в FastAPI имеется тестовый клиент для организации автоматического тестирования API. Таким образом, непосредственно API планируется реализовать с использованием web-фреймворка FastAPI, с реализацией классов, описанных на диаграмме классов для API, с помощью библиотеки Pydantic.

онлайн калькулятор разрабатывается как проект для проведения практических занятий со студентами. Основная цель - предоставить студентам возможность участия в разработке web-приложения с минимальными затратами и навыками разработки. Предполагается, что студент может добавить новый алгоритм в онлайн калькулятор,

описав декларативно его структуру - название, описание, конфигурацию входных и выходных данных, и разработав функцию на языке программирования python, которая этот алгоритм реализует. В результате основная задача серверной части онлайн калькулятора - организация проверки и сборки алгоритмов из файлов с описанием и исходным кодом алгоритма, подготовленных студентами.

Предполагается, что в проекте с исходным кодом для серверной части будет создан отдельный каталог с именем `algorithms`. В это каталоге каждый алгоритм будет располагаться в отдельном подкаталоге. Имя подкаталога можно использовать в качестве уникального имени для алгоритма. Описание алгоритма будет располагаться в JSON файле с названием `definition.json`, реализация функции для алгоритма в файле `main.py`, кроме того, в файле `tests.py` будет расположен набор модульных тестов для автоматической проверки реализации алгоритма. В результате каждый алгоритм будет представлять собой набор однотипных файлов в подкаталогах каталога `algorithms` и сборку онлайн калькулятора можно будет осуществить последовательно обрабатывая указанные подкаталоги. При сборке алгоритма в онлайн калькулятор необходимо убедиться, что он корректно описан и реализован. Для проверки описания алгоритма можно использовать валидацию данных в формате JSON с помощью JSON Schema - языка описания структуры JSON. При сборке алгоритма нужно запускать на выполнение набор автотестов, проверяющих работу функцию и прерывать сборку при провале теста. Также необходимо проверять, что функция для алгоритма принимает набор параметров, описанный в файле `definition.json`, и возвращает набор результатов, описанный в файле. Для организации такой проверки, планируется добавить в описание входных и выходных данных значение по умолчанию, тогда значения по умолчанию для входных данных можно передать в функцию, выполнить ее и проверить, что выходные данные соответствуют заданным значениям по умолчанию.

В результате серверная часть должна содержать класс для описания структуры входных/выходных данных, класс, описывающий алгоритм и предоставляющий возможность для его выполнения, класс, конструирующий алгоритмы из исходного кода, написанного студентами, и класс, представляющий набор алгоритмов и реализующий функциональность для выполнения запросов к API. Указанные классы реализуют внутреннюю логику и составляют ядро системы, будут разработаны на чистом python.

Для проектирования структуры серверной части онлайн калькулятора использован инструмент моделирования Visual Paradigm. В UML-модель, разработанную на этапе анализа, добавлена диаграмма классов, представляющая ядро системы. Разработанная диаграмма классов представлена на рисунке 2.2.

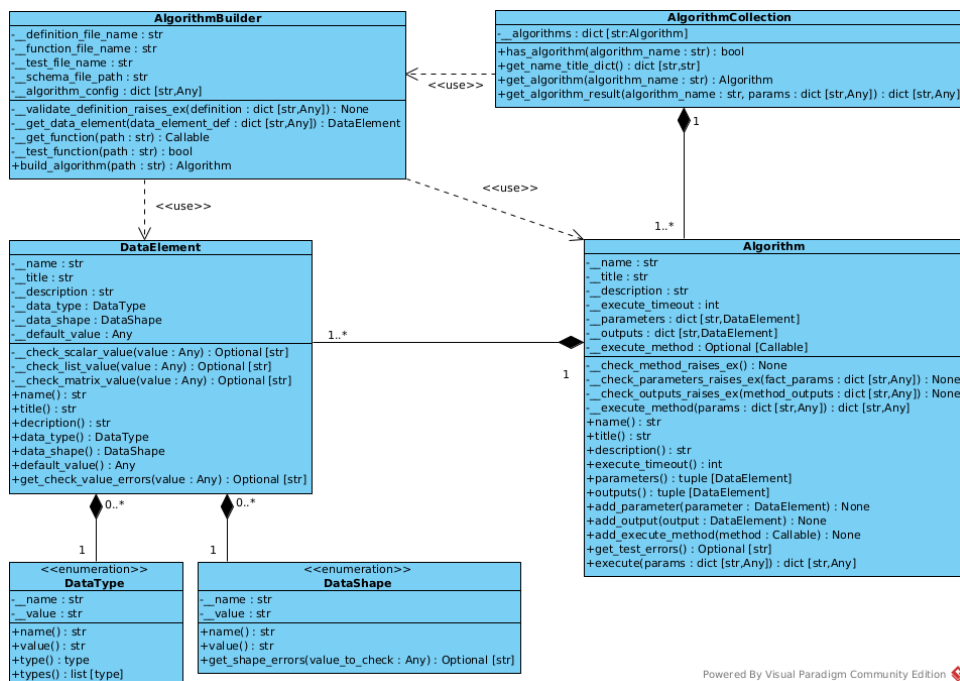


Рисунок 2.2. Диаграмма классов

Класс `DataType` является перечислением, представляет допустимые типы данных для входных и выходных данных. Возможными значениями класса являются `INT`, `FLOAT`, `STRING`, `BOOL`, соответствующие типам данных языка python `int`, `float`, `str`, `bool`. Класс имеет свойство `type`, возвращающее соответствующий тип данных языка python и свойство `types`, возвращающее список допустимых типов данных.

Класс `DataShape` является перечислением, представляет допустимые размерности для входных и выходных данных. Возможными значениями класса являются `SCALAR`, `LIST`, `MATRIX`, соответствующие скалярному значению, списку скалярных значений и двумерную матрицу соответственно. Класс имеет метод `get_shape_errors`, проверяющий соответствие проверяемого параметра размерности и возвращающее текст ошибки или значение `None`, если проверка не выявила ошибок.

Класс `DataElement` представляет элемент входных или выходных данных для алгоритма. `DataElement` содержит следующие приватные атрибуты:

- `__name` - уникальное имя элемента входных/выходных данных,

- `__title` - название имя элемента входных/выходных данных,
- `__description` - описание,
- `__data_shape` - тип данных (`DataShape`),
- `__data_type` - размерность данных (`DataElement`)
- `__default_value` - значение по умолчанию.

Для каждого приватного поля в классе `DataElement` имеется соответствующее публичное свойство для получения значения приватного поля. В классе имеется публичный метод `get_check_value_errors`, проверяющий соответствие проверяемого параметра типу данных и размерности и возвращающее текст ошибки или значение `None`, если проверка не выявила ошибок. Также имеются приватные методы `__check_scalar_value`, `__check_list_value`, `__check_matrix_value`, проверяющие корректность значения для скалярного, списочного или матричного значения соответственно.

Класс `Algorithm` представляет непосредственно алгоритм с описанием и методом для его выполнения. `Algorithm` содержит следующие приватные атрибуты:

- `__name` - уникальное имя алгоритма,
- `__title` - название алгоритма,
- `__description` - описание алгоритма,
- `__execute_timeout` - время отведенное для выполнения алгоритма,
- `__parameters` - словарь входных данных, где уникальное имя используется в качестве ключа, а объект `DataElement` в качестве значения,
- `__outputs` - список выходных данных, где уникальное имя используется в качестве ключа, а объект `DataElement` в качестве значения,
- `__execute_method` - метод, реализующий алгоритм, импортируемый из исходного кода студентов.

Для доступа на просмотр приватных полей реализованы следующие свойства:

- `name` - уникальное имя алгоритма,
- `title` - название алгоритма,
- `description` - описание алгоритма,
- `execute_timeout` - время отведенное для выполнения алгоритма,
- `parameters` - кортеж входных данных (`DataElement`),

- `outputs` - кортеж выходных данных (`DataElement`).

Публичными методами класса `Algorithm` являются:

- `add_parameter` - добавляет в словарь `__parameters` элемент входных данных, переданный в параметре `parameter`.
- `add_output` - добавляет в словарь `__outputs` элемент выходных данных, переданный в параметре `output`.
- `add_execute_method` - сохраняет в поле `__execute_method` переданный в параметре `method` метод, реализующий алгоритм.
- `get_test_errors` - запускает тестовое выполнение алгоритма со входными данными, заданными по умолчанию. Метод возвращает текст ошибки или значение `None`, если выполнение алгоритма прошло без ошибок.
- `execute` - запускает алгоритм на выполнение со входными параметрами, переданными в виде словаря в переменной `params`. Возвращает словарь с результатами выполнения алгоритма.

Приватными методами класса `Algorithm` являются:

- `__check_method_raises_ex` - проверяет возможность добавления метода в алгоритм. Вызывает исключение, если проверка выявила ошибки.
- `__check_parameters_raises_ex` - проверяет соответствие переданных параметров описанным входным данным. Вызывает исключение, если проверка выявила ошибки.
- `__check_outputs_raises_ex` - проверяет соответствие полученных результатов описанным выходным данным. Вызывает исключение, если проверка выявила ошибки.

Класс `AlgorithmBuilder` создает объекты класса `Algorithm` из файлов с исходным кодом. Класс содержит следующие приватные атрибуты:

- `__definition_file_name` - название файла с описанием алгоритма.
- `__function_file_name` - название файла с методом для алгоритма.
- `__test_file_name` - название файла с автотестами для метода алгоритма.
- `__schema_file_path` - путь к файлу JSON Schema для валидации описания алгоритма.

- `__algorithm_config` - словарь с параметрами для создания алгоритма, на данный момент имеется один параметр - `execute_timeout` - время отведенное для выполнения алгоритма.

В классе `AlgorithmBuilder` имеется один публичный метод - `build_algorithm`, для создания объекта `Algorithm` на основе файлов в указанном каталоге.

Приватными методами класса `Algorithm` являются:

- `__validate_definition_raises_ex` - осуществляет валидацию описания алгоритма на основе JSON Schema. Вызывает исключение, если проверка выявила ошибки.
- `__get_data_element` - создает элемент данных (`DataElement`), для включения во входные/выходные данные алгоритма.
- `__get_function` - импортирует функцию для алгоритма из файла с исходным кодом.
- `__test_function` - запускает выполнение автотестов для импортируемой функции. Возвращает результат выполнения автотестов.

Класс `AlgorithmCollection` представляет собой набор объектов класса `Algorithm`, созданных объектом класса `AlgorithmBuilder`. Именно объект класса `AlgorithmCollection` предоставляет данные, необходимые для API серверной части онлайн калькулятора. Класс имеет одно приватное поле `__algorithms`, содержащее словарь с уникальными именами алгоритмов в качестве ключей и объекты класса `Algorithm` в качестве значений.

Публичными методами класса `AlgorithmCollection` являются:

- `has_algorithm` - проверяет есть ли в словаре ключ с именем алгоритма, переданным в параметре `algorithm_name`.
- `get_name_title_dict` - возвращает словарь с уникальными именами алгоритмов в качестве ключей и названиями алгоритмов в качестве значений.
- `get_algorithm` - возвращает объект класса `Algorithm` с именем алгоритма, переданным в параметре `algorithm_name`.
- `get_algorithm_result` - возвращает результат выполнения алгоритма по имени, переданным в параметре `algorithm_name`, со входными параметрами, переданными в параметре `params`.

2.2.2 Проектирование взаимодействия элементов серверной части онлайн калькулятора

На основе диаграммы прецедентов, функциональных требований, разработанных на этапе анализа, а также диаграммы классов и описания API выполнено проектирование взаимодействия элементов структуры онлайн калькулятора. Серверная часть онлайн калькулятора реализует прецеденты «Просмотреть список алгоритмов» (GET запрос к конечной точке /api/algorithms), «Просмотреть описание алгоритма» (GET запрос к конечной точке /api/algorithms/{algorithm_name}) и «Выполнить алгоритм» (POST запрос к конечной точке /api/algorithms/{algorithm_name}). Кроме того, серверная часть осуществляет сборку алгоритмов из исходного кода, написанного студентами. Сборка алгоритмов не описывалась как прецедент, так как этот процесс скрыт от пользователя, можно считать, что сборка алгоритмов входит в прецедент «Просмотреть список алгоритмов». В ходе проектирования, с помощью инструмента моделирования Visual Paradigm, для прецедентов разработаны диаграммы последовательностей, отражающие взаимодействие элементов серверной части онлайн калькулятора.

На рисунке 2.3 представлена диаграмма последовательностей для основного потока прецедента «Просмотреть список алгоритмов». В основном потоке объект класса AlgorithmCollection обращается в цикле к входящим в него объектам класса Algorithm и запрашивает значения свойств name и title. В результате объект класса AlgorithmCollection возвращает словарь с ключами name и значениями title для каждого алгоритма.

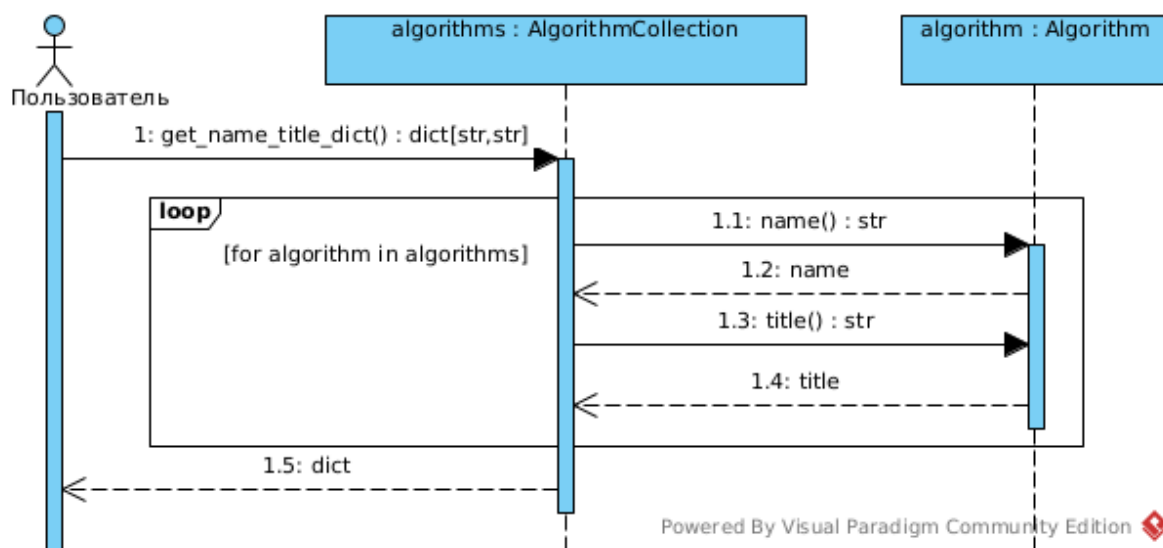


Рисунок 2.3. Диаграмма последовательностей для основного потока прецедента «Просмотреть список алгоритмов»

Описание системных операций, представленных на диаграмме.

Имя: `get_name_title_dict()`.

Обязанности: предоставить словарь, в котором для каждого из имеющихся алгоритмов имеется ключ с уникальным именем алгоритма и значение с названием алгоритма.

Ссылки: прецедент «Просмотреть список алгоритмов».

Примечания: нет.

Исключения: нет.

Предусловия: создан объект `algorithms` класса `AlgorithmCollection`, содержащих набор объектов класса `Algorithm`.

Постусловия: создан объект `dict` - словарь, в котором для каждого из имеющихся алгоритмов имеется ключ с уникальным именем алгоритма и значение с названием алгоритма.

На рисунке 2.4 представлена диаграмма последовательностей для наполнения объекта класса `AlgorithmCollection`. Объект класса `AlgorithmCollection` создает объект класса `AlgorithmBuilder`, который в цикле создает из файлов с исходным кодом объекты класса `Algorithm`, обрабатывая каталоги. После создания объекта класса `Algorithm` производится добавление входных и выходных данных и добавление метода, реализующего алгоритм. Созданные объекты класса `Algorithm` возвращаются объекту класса `AlgorithmCollection` и включаются в словарь `__algorithms`.

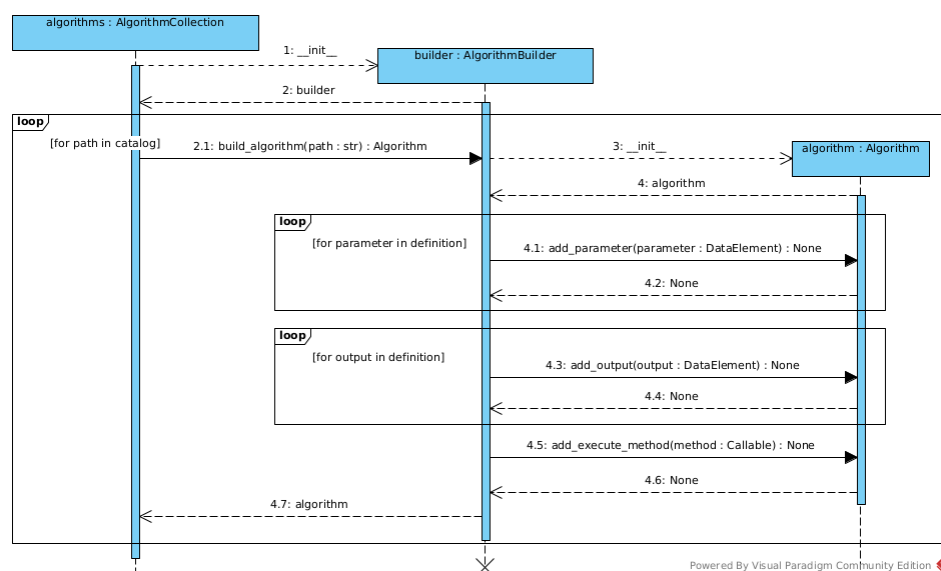


Рисунок 2.4. Диаграмма последовательностей для наполнения объекта класса `AlgorithmCollection`

Описание системных операций, представленных на диаграмме.

Имя: `build_algorithm(path)`.

Обязанности: создать объект класса `Algorithm` из файлов с исходным кодом в указанном каталоге.

Ссылки: прецедент «Просмотреть список алгоритмов».

Примечания: в параметре `path` указывается путь к каталогу с файлами исходного кода.

Исключения: не найден файл с исходным кодом, описание алгоритма не проходит валидацию, метод для алгоритма не прошел проверку.

Предусловия: нет.

Постусловия: создан объект класса `Algorithm` из файлов с исходным кодом в указанном каталоге.

Имя: `add_parameter(parameter)`.

Обязанности: добавить параметр `parameter` в атрибут `__parameters` объекта класса `Algorithm`.

Ссылки: прецедент «Просмотреть список алгоритмов».

Примечания: в параметре `parameter` передается объект класса `DataElement`.

Исключения: атрибут `__parameters` уже содержит параметр с указанным именем.

Предусловия: создание объекта класса `Algorithm`.

Постусловия: объект класса `DataElement`, переданный в параметре `parameter`, добавлен в словарь `__parameters` объекта класса `Algorithm`.

Имя: `add_output(output)`.

Обязанности: добавить элемент выходных данных `output` в атрибут `__outputs` объекта класса `Algorithm`.

Ссылки: прецедент «Просмотреть список алгоритмов».

Примечания: в параметре `output` передается объект класса `DataElement`.

Исключения: атрибут `__outputs` уже содержит элемент выходных данных с указанным именем.

Предусловия: создание объекта класса `Algorithm`.

Постусловия: объект класса `DataElement`, переданный в параметре `output`, добавлен в словарь `__outputs` объекта класса `Algorithm`.

Имя: `add_execute_method(method)`.

Обязанности: добавить метод в атрибут `__execute_method` объекта класса `Algorithm`.

Ссылки: прецедент «Просмотреть список алгоритмов».

Примечания: в параметре `method` передается функция, импортированная из файла с исходным кодом.

Исключения: метод для алгоритма не прошел проверку.

Предусловия: создание объекта класса `Algorithm`.

Постусловия: функция, переданная в параметре `method`, связана с атрибутом `__execute_method` объекта класса `Algorithm`.

На рисунке 2.5 представлена диаграмма последовательностей для основного потока прецедента «Просмотреть описание алгоритма». В основном потоке объект класса `AlgorithmCollection` возвращает объект класса `Algorithm` с уникальным именем согласно указанному в параметре.

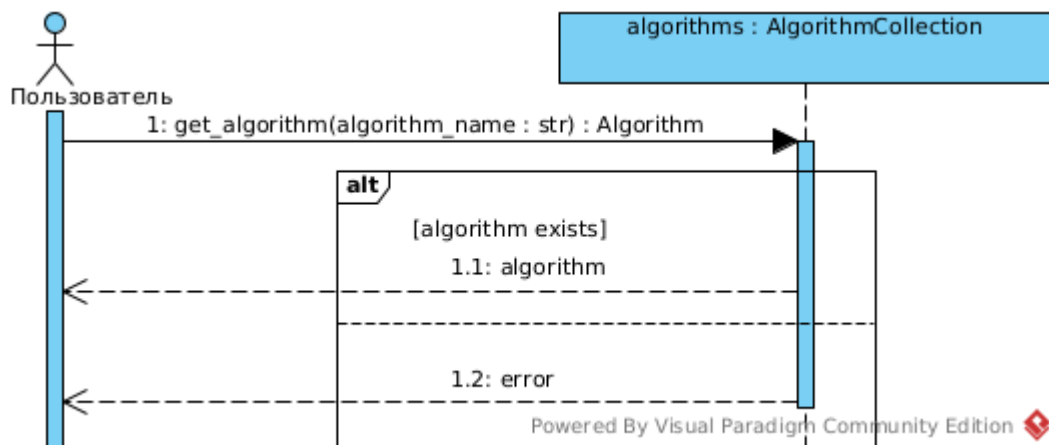


Рисунок 2.5. Диаграмма последовательностей для основного потока прецедента «Просмотреть описание алгоритма»

Описание системных операций, представленных на диаграмме.

Имя: `get_algorithm(algorithm_name)`.

Обязанности: передать вызывающей стороне ссылку объект класса `Algorithm`.

Ссылки: прецедент «Просмотреть описание алгоритма».

Примечания: в параметре `algorithm_name` указывается уникальное имя алгоритма.

Исключения: не найден алгоритм с указанным именем.

Предусловия: создан объект `algorithms` класса `AlgorithmCollection`, содержащих набор объектов класса `Algorithm`.

Постусловия: вызывающей стороне передана ссылка объект класса `Algorithm`.

На рисунке 2.6 представлена диаграмма последовательностей для основного потока прецедента «Выполнить алгоритм». В основном потоке объект класса `AlgorithmCollection` обращается к объекту класса `Algorithm`, с уникальным именем согласно указанному в параметре, передает фактические значения для входных данных алгоритма и получает значения выходных данных. Проверки входных и выходных данных вынесены на отдельные диаграммы последовательностей.

Описание системных операций, представленных на диаграмме.

Имя: `get_algorithm_result(algorithm_name, params)`.

Обязанности: создать объект словарь с выходными данными, полученными в результате выполнения алгоритма.

Ссылки: прецедент «Выполнить алгоритм».

Примечания: в параметре `algorithm_name` указывается уникальное имя алгоритма, в параметре `params` передаются фактические значения входных данных для алгоритма.

Исключения: не найден алгоритм с указанным именем, метод для алгоритма не прошел проверку, входные данные не соответствуют описанию алгоритма, время для выполнения алгоритма истекло, ошибка во время выполнения алгоритма, выходные данные не соответствуют описанию алгоритма.

Предусловия: создан объект `algorithms` класса `AlgorithmCollection`, содержащих набор объектов класса `Algorithm`.

Постусловия: создан объект словарь с выходными данными, полученными в результате выполнения алгоритма.

Имя: `execute(params)`.

Обязанности: создать объект словарь с выходными данными, полученными в результате выполнения алгоритма.

Ссылки: прецедент «Выполнить алгоритм».

Примечания: в параметре `params` передаются фактические значения входных данных для алгоритма.

Исключения: метод для алгоритма не прошел проверку, входные данные не соответствуют описанию алгоритма, время для выполнения алгоритма истекло, ошибка во время выполнения алгоритма, выходные данные не соответствуют описанию алгоритма.

Предусловия: создан объект класса Algorithm.

Постусловия: создан объект словарь с выходными данными, полученными в результате выполнения алгоритма.

Имя: `__check_method_raises_ex()`.

Обязанности: проверить возможность выполнения алгоритма и вызвать исключение в случае наличия ошибок.

Ссылки: прецедент «Выполнить алгоритм».

Примечания:

Исключения: метод для алгоритма не прошел проверку.

Предусловия: создан объект класса Algorithm.

Постусловия: вызов исключения в случае наличия ошибок.

Имя: `__execute_method(params)`.

Обязанности: создать объект словарь с выходными данными, полученными в результате выполнения алгоритма.

Ссылки: прецедент «Выполнить алгоритм».

Примечания: в параметре `params` передаются фактические значения входных данных для алгоритма.

Исключения: время для выполнения алгоритма истекло, ошибка во время выполнения алгоритма.

Предусловия: создан объект класса Algorithm.

Постусловия: создан объект словарь с выходными данными, полученными в результате выполнения алгоритма.

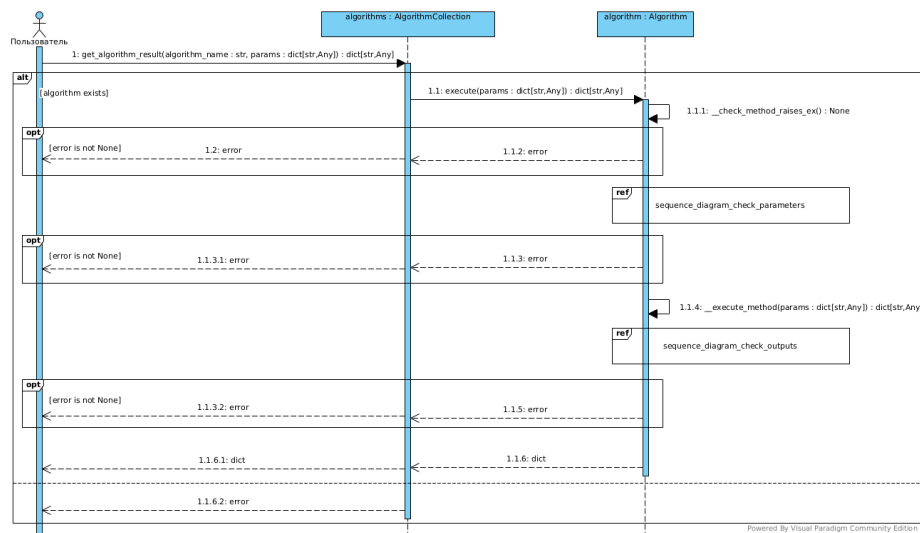


Рисунок 2.6. Диаграмма последовательностей для основного потока прецедента «Выполнить алгоритм»

На рисунке 2.7 представлена диаграмма последовательностей для проверки входных данных, где для каждого параметра проверяется его значение на предмет соответствия размерности и типу данных, указанным в описании алгоритма.

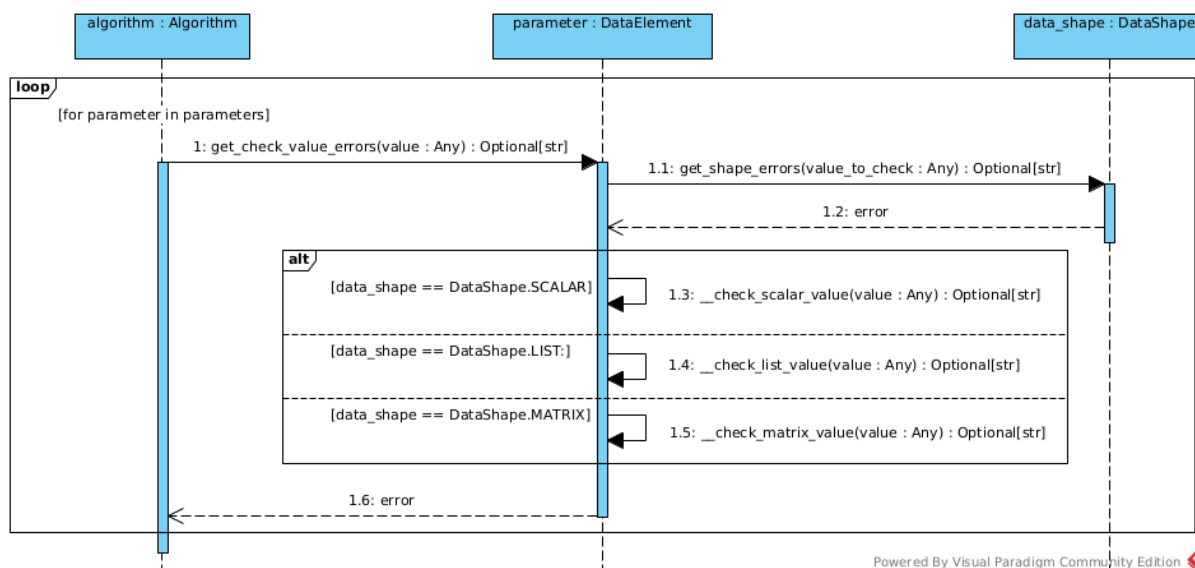


Рисунок 2.7. Диаграмма последовательностей для проверки входных данных

Описание системных операций, представленных на диаграмме.

Имя: get_check_value_error(value).

Обязанности: проверить значение элемента входных данных на предмет соответствия размерности и типу данных.

Ссылки: прецедент «Выполнить алгоритм».

Примечания: в параметре value передаются фактическое значение элемента входных данных для алгоритма.

Исключения: нет.

Предусловия: создан объект класса Algorithm.

Постусловия: создан объект строка содержащая ошибки проверки значения.

Имя: get_shape_error(value).

Обязанности: проверить значение элемента входных данных на предмет соответствия размерности.

Ссылки: прецедент «Выполнить алгоритм».

Примечания: в параметре value передаются фактическое значение элемента входных данных.

Исключения: нет.

Предусловия: создан объект класса DataElement.

Постусловия: создан объект строка содержащая ошибки проверки значения.

Имя: __check_scalar_value(value).

Обязанности: проверить значение скалярного элемента входных данных на предмет соответствия типу данных.

Ссылки: прецедент «Выполнить алгоритм».

Примечания: в параметре value передаются фактическое значение элемента входных данных для алгоритма.

Исключения: нет.

Предусловия: создан объект класса DataElement.

Постусловия: создан объект строка содержащая ошибки проверки значения.

Имя: __check_list_value(value).

Обязанности: проверить значения в списке элемента входных данных на предмет соответствия типу данных.

Ссылки: прецедент «Выполнить алгоритм».

Примечания: в параметре value передаются фактическое значение элемента входных данных для алгоритма.

Исключения: нет.

Предусловия: создан объект класса DataElement.

Постусловия: создан объект строка содержащая ошибки проверки значения.

Имя: `__check_matrix_value(value)`.

Обязанности: проверить значения в матрице элемента входных данных на предмет соответствия типу данных.

Ссылки: прецедент «Выполнить алгоритм».

Примечания: в параметре `value` передаются фактическое значение элемента входных данных для алгоритма.

Исключения: нет.

Предусловия: создан объект класса `DataElement`.

Постусловия: создан объект строка содержащая ошибки проверки значения.

На рисунке 2.8 представлена диаграмма последовательностей для проверки выходных данных, где для каждого элемента выходных данных проверяется его значение на предмет соответствия размерности и типу данных, указанным в описании алгоритма. В связи с тем, что входные и выходные данные представлены экземплярами класса `DataElement`, для проверки выходных данных используются те же системные операции, что и для проверки входных данных.

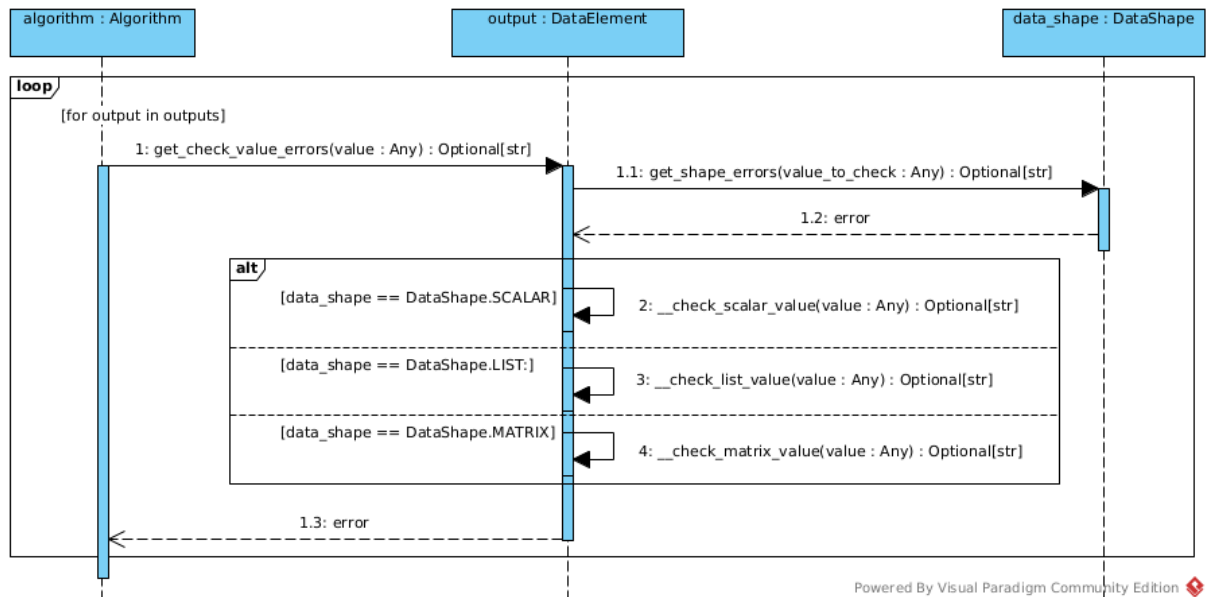


Рисунок 2.8. Диаграмма последовательностей для проверки выходных данных

Глава 3. Разработка онлайн калькулятора

Для разработки онлайн калькулятора на платформе GitHub создана организация ОММАТ, в которую объединены участники проекта. Для организации создан проект для управления и координации работы участников, участники организации распределены по командам в соответствии с ролями. Также для организации создан набор репозиторий:

1. Для документации проекта.
2. Для модели предметной области на языке Archimate.
3. Для исходного кода серверной части онлайн калькулятора.
4. Для исходного кода клиентской части онлайн калькулятора.

Разработке программного продукта предшествовал этап разработки и настройки автоматических процессов непрерывной интеграции и непрерывной поставки изменений согласно методологии DevOps. Внедрение данных процессов позволяет снизить затраты и упростить все процессы жизненного цикла продукта, в том числе сопровождение и дальнейшее развитие продукта.

В результате внедрены следующие автоматические процессы непрерывной интеграции и непрерывной поставки:

1. Сборка и публикация Archimate-модели для продукта.
2. Выполнение модульных тестов.
3. Анализ кода на платформе SonarCloud.
4. Проверка стиля кода.
5. Обновление тестового стенда.
6. Обновление продуктового стенда.
7. Сборка и публикация документации к исходному коду.

Для репозиторий с исходным кодом процессы непрерывной интеграции и непрерывной поставки объединены в конвейер, в котором отдельные процессы запускаются в зависимости от различных условий, в том числе результата выполнения предшествующих процессов. Пример конвейерного выполнения процессов при добавлении изменений в тестовую ветку репозитория представлен на рисунке 3.1. Согласно рисунку в первую очередь выполняются модульные для исходного кода с изменениями и если имеются ошибки выполнения тестов, то конвейер останавливается.

Если тесты выполнены без ошибок, то исходный код с изменениями передается на сервер анализатора кода SonarCloud и параллельно обновляется на сервере с развернутым тестовым стендом продукта. Если изменения добавляются в продуктивную ветку репозитория, то процесс обновления тестового стенда пропускается, вместо него запускается процесс обновления продуктового стенда и процесс обновления документации исходного кода.

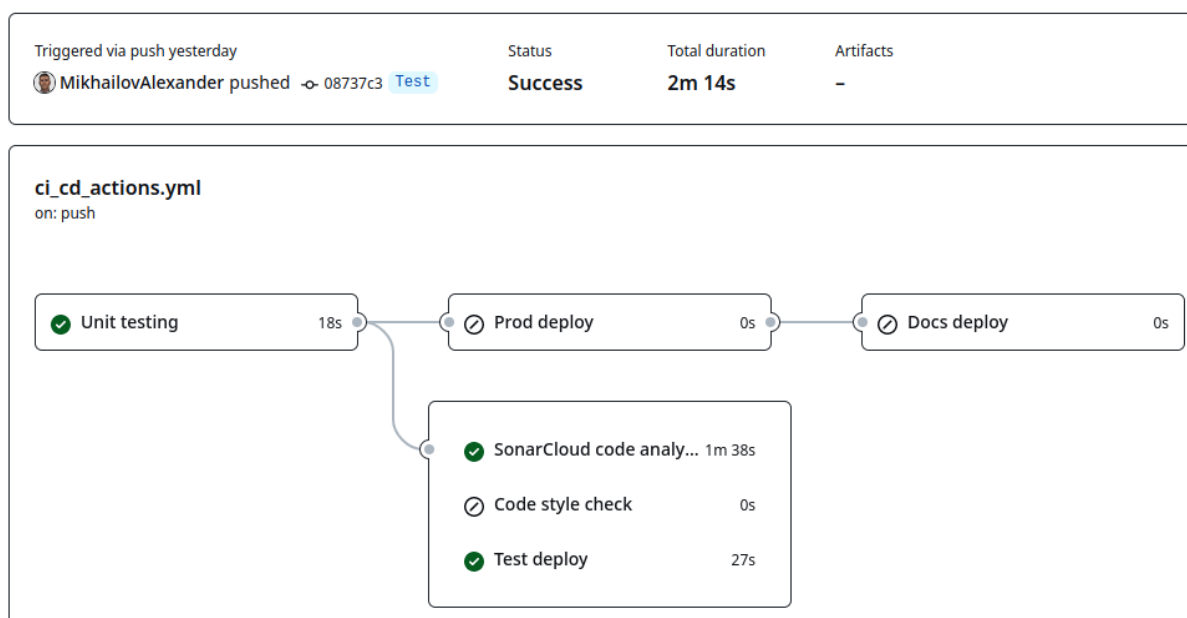


Рисунок 3.1. Выполнение процессов непрерывной интеграции и непрерывной поставки.

Процесс автоматической проверки стиля кода, позволяет поддерживать единообразие исходного кода и его соответствие стандартам языка Python, в частности PEP 8. При создании запроса на добавление изменений в исходный код, процесс проверяет стиль кода и добавляет комментарии непосредственно к строкам, содержащим ошибки, как это иллюстрирует рисунок 3.2., где первый комментарий сообщает о лишнем пробеле, а второй об отсутствующих пробелах.

Анализ кода на платформе SonarCloud позволяет автоматически выявлять ошибки в исходном коде до их внесения в продукт. При открытии запроса для внесения изменений, предлагаемые изменения проверяются на наличие ошибок, угроз безопасности, низкокачественного кода и сложности восприятия кода. В результате оформляется отчет об ошибках, пример которого представлен на рисунке 3.3. Для каждой ошибки можно посмотреть справку о том, почему данный код является ошибочным, а также рекомендации по исправлению ошибки.



Рисунок 3.2. Комментарии к ошибкам оформления исходного кода.



Рисунок 3.3. Ошибки выявленные при автоматическом анализе исходного кода.

Для замечаний о сложности восприятия исходного кода предоставляется детализация с указанием конкретных операторов и уровня сложности, который они приносят в исходных код. Пример такой детализации представлен на рисунке 3.4. Данный пример иллюстрирует, что для снижения сложности до приемлемого уровня достаточно исключить оператор иначе в 9 строке.

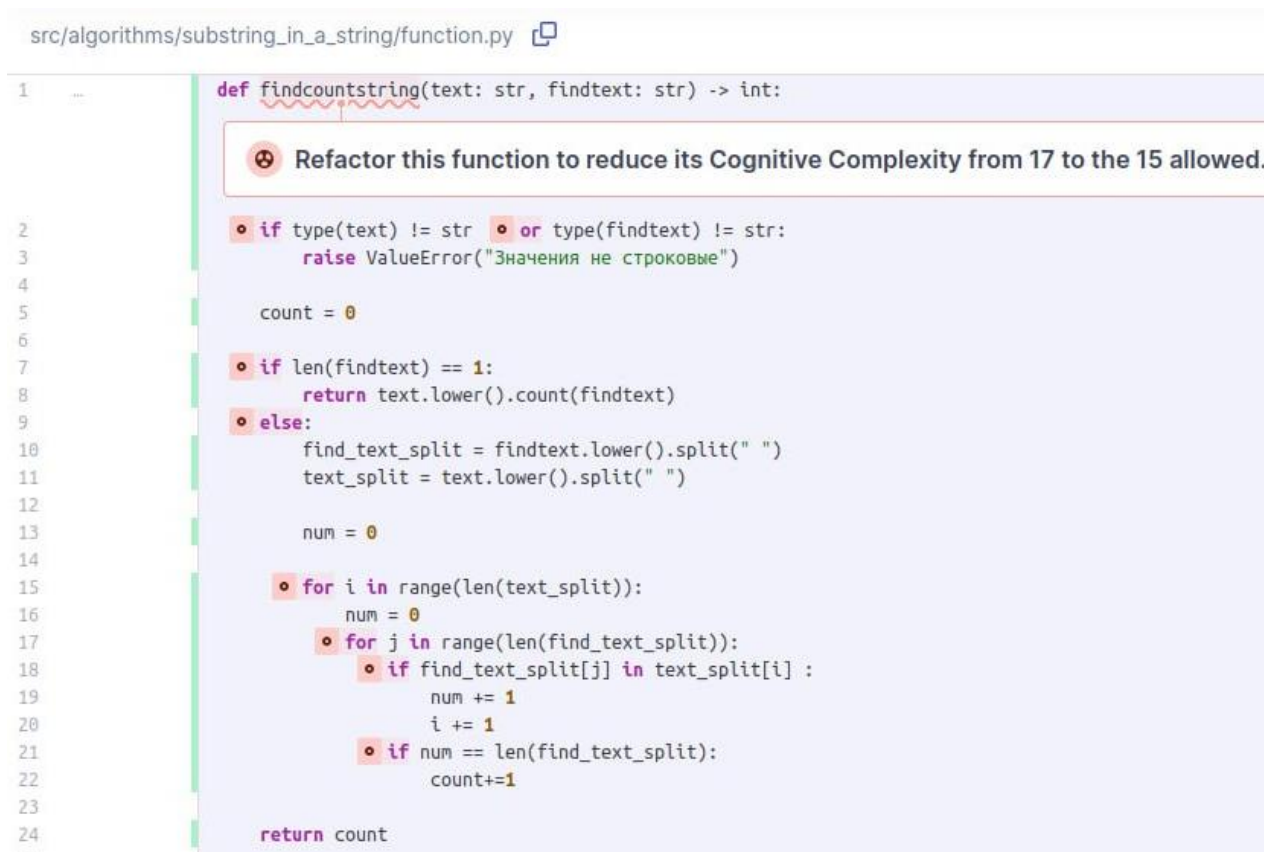


Рисунок 3.4. Детализация сложности восприятия исходного кода.

3.1. Разработка серверной части онлайн калькулятора

Согласно сформулированным требованиям серверная часть Онлайн-калькулятора должна быть реализована на языке программирования Python. В качестве среды для разработки выбрана IDE PyCharm Professional Edition, предоставляемая компанией JetBrains для некоммерческого использования по студенческой лицензии. IDE PyCharm предназначена для разработки на языке Python и имеет встроенную интеграцию с сервисом GitHub, инструменты отладки, анализа кода и модульного тестирования.

Разрабатываемый исходный код снабжался специальными строками документации кода согласно стандарту PEP 257. Для каждого реализуемого класса в строке документации описывается назначение класса, публичные атрибуты, свойства и методы. Документация для методов содержит краткое описание метода, типы и описание параметров, вызываемые исключения и возвращаемое значение. Указанные строки документации используются для автоматической генерации и публикации документации.

3.1.1. Структура проекта и конфигурационные файлы программного модуля

В репозитории для проекта создана следующая структура каталогов и файлов:

```
|— .github
|   |— workflows
|       |— ci_cd_actions.yml
|— config
|   |— app_config.json
|   |— log_config.json
|— docs
|— src
|   |— algorithms
|   |— app_tests
|   |— core
|   |— core_tests
|   |— __init__.py
|   |— api_models.py
|   |— main.py
|   |— requirements.txt
|   |— test_runner.py
|— Dockerfile
|— LICENSE
|— README.md
|— docker-compose.prod.yml
|— docker-compose.yml
|— sonar-project.properties
|— tox.ini
```

В каталоге `.github/ workflows` расположен файл `ci_cd_actions.yml`, содержащий конфигурацию автоматических процессов непрерывной интеграции и непрерывной поставки.

В каталоге `config` расположены конфигурационные файлы продукта `log_config.json` и `app_config.json`. В файле `log_config.json` в формате словаря размещены настройки

логирования, в том числе уровень логирования, название файла, предельный размер и количество файлов. Логирование в программном модуле реализовано с использованием стандартной библиотеки logging языка программирования Python. В файле app_config.json размещены настройки самого модуля, сгруппированные в разделы - пути к файлам, параметры для алгоритмов и параметры для настройки web-сервера. Пример заполнения конфигурационного файла представлен на рисунке 3.5.

В каталоге docs расположены файлы для настройки автоматической генерации документации к исходному коду.

В каталоге src расположен исходный код серверной части Онлайн-калькулятора, в том числе классы ядра в каталоге core, модульные тесты для классов ядра в каталоге core_tests и модульные тесты серверной части в целом в каталоге app_tests. Исходный код для API серверной части реализован в файлах api_models.py и main.py. Таким образом, ядро продукта и тесты для классов ядра отделены от реализации API с использованием конкретного web-фреймворка.

```
{
  "path_config": {
    "definition_file_name": "definition.json",
    "function_file_name": "function.py",
    "test_file_name": "tests.py",
    "json_schema_file_path": "src/core/schemas/algorithm_schema.json",
    "algorithms_catalog_path": "src/algorithms"
  },
  "algorithm_config": {
    "execute_timeout": 5
  },
  "web_config": {
    "cors": {
      "origins": [
        "http://test.ommat.ru",
        "http://prod.ommat.ru",
        "https://test.ommat.ru",
        "https://prod.ommat.ru",
        "http://localhost",
        "http://localhost:4444",
        "http://localhost:5555",
        "http://localhost:44486"
      ],
      "credentials": true,
      "methods": ["*"],
      "headers": ["*"]
    }
  }
}
```

Рисунок 3.5. Пример заполнения конфигурационного файла.

Файлы Dockerfile, docker-compose.prod.yml и docker-compose.yml в корне репозитория необходимы для развертывания серверной части Онлайн-калькулятора в контейнере Docker на тестовом и продуктивном стендах.

Файлы sonar-project.properties и tox.ini в корне репозитория необходимы для процесса автоматического анализа кода на сервисе SonarCloud.

3.1.2. Разработка ядра Онлайн-калькулятора

В каталоге src/core, согласно описанию классов, диаграммам последовательностей и описанию системных операций, разработанных на этапе проектирования, реализованы классы составляющие ядро Онлайн-калькулятора - DataType, DataShape, DataElement, Algorithm, AlgorithmBuilder, AlgorithmCollection. Исходный код классов ядра представлен в приложениях В, Г, Д, Е. На рисунке 3.6. В качестве примера представлен исходный код класса DataShape, описывающего размерность входных или выходных данных для алгоритма.

Класс DataShape содержит строки документации, как для самого класса, так и для метода get_shape_errors. На рисунке 3.7 представлен фрагмент автоматически сгенерированной документации для этого класса, согласно строкам документации, размещенным в исходном коде.

```
class DataShape(UppercaseStrEnum):
    """Класс DataShape является перечислением, представляет допустимые
    размерности для входных и выходных данных. Возможными значениями
    класса являются SCALAR, LIST, MATRIX, соответствующие скалярному значению,
    списку скалярных значений и двумерную матрицу соответственно.

    """
    SCALAR = auto()
    LIST = auto()
    MATRIX = auto()

    def get_shape_errors(self, value_to_check: Any) -> Optional[str]:
        """Проверяет соответствие проверяемого значения размерности.

        :param value_to_check: значение для проверки;
        :type value_to_check: Any
        :return: текст сообщения об ошибке проверки размерности.
        :rtype: str or None
        """
        if value_to_check is None:
            return NONE_VALUE_MSG
        if self.value == self.SCALAR and type(value_to_check) \
            not in DataType.types():
            return NOT_SCALAR_VALUE_MSG
        if self.value == self.LIST and type(value_to_check) != list:
            return NOT_LIST_VALUE_MSG
        if self.value == self.MATRIX:
            if type(value_to_check) != list:
                return NOT_MATRIX_VALUE_MSG
            if len(value_to_check) == 0:
                return NOT_MATRIX_VALUE_MSG
            for row_idx, row in enumerate(value_to_check):
                if type(row) != list:
                    return NOT_LIST_ROW_TEMPL.format(row_idx)
            return None
```

Рисунок 3.6. Реализация класса DataShape.

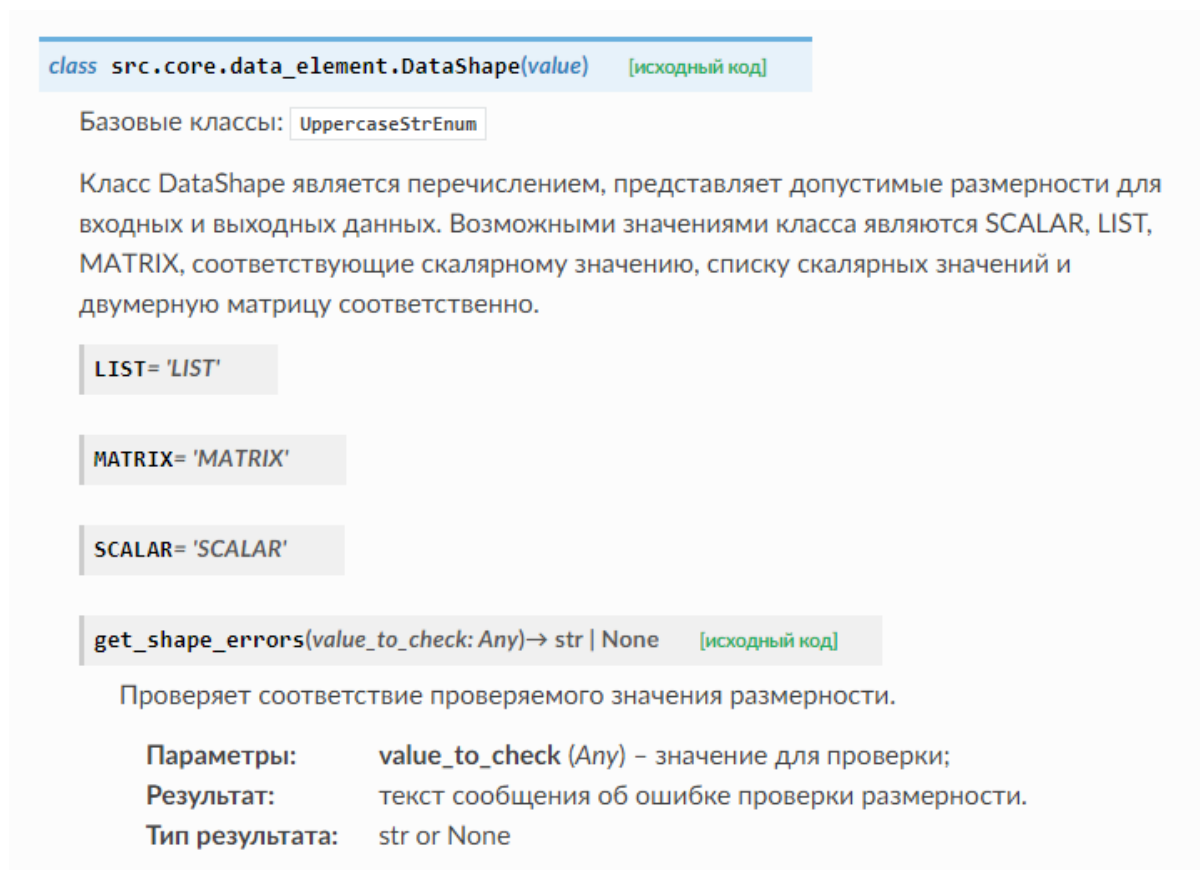


Рисунок 3.7. Документация класса *DataShape*.

На рисунке 3.8 представлена реализация метода `build_algorithm` класса `AlgorithmBuilder`, который на основе файлов с исходным кодом создает объекты класса `Algorithm` для включения в состав объекта `AlgorithmCollection`. В методе реализовано логирование с использованием стандартной библиотеки `logging`, как обычного хода выполнения метода, так и возникающих ошибок, в зависимости от установленного в конфигурации уровня логирования. Также в методе производится валидация файла в формате JSON, содержащего описание алгоритма. Валидация проводится с помощью шаблона `JSON Schema` - языка описания структуры JSON. Шаблон позволяет описать допустимую структуру JSON файла, в том числе обязательные и необязательные разделы, типы данных, кратность и допустимые значения. Для валидации описания алгоритма разработаны шаблоны для входных/выходных данных и для алгоритма в целом. Пример шаблона представлен на рисунке 3.9.

```

self.__logger.info(path)
with open(path + '/' + self.__definition_file_name, 'r',|
        encoding='utf-8') as def_file:
    definition = json.load(def_file)
self.__validate_definition_raises_ex(definition)
name = os.path.split(path)[-1]
alg = Algorithm(name, definition[self.TITLE],
                definition[self.DESCRPTION], self.__log_config,
                self.__algorithm_config[self.EXECUTE_TIMEOUT])
for param_def in definition[self.PARAMETERS]:
    alg.add_parameter(self.__get_data_element(param_def))
for output_def in definition[self.OUTPUTS]:
    alg.add_output(self.__get_data_element(output_def))
if not self.__test_function(path):
    self.__logger.error(UNIT_TEST_FAILED_MSG)
    raise RuntimeError(UNIT_TEST_FAILED_MSG)
alg.add_execute_method(self.__get_function(path))
return alg

```

Рисунок 3.8. Реализация метода `build_algorithm` класса `AlgorithmBuilder`.

```

{
  "type": "object",
  "properties": {
    "title": { "type": "string" },
    "description": { "type": "string" },
    "parameters": {
      "type": "array",
      "items": {
        "$ref": "file:src/core/schemas/data_element_schema.json"
      }
    },
    "outputs": {
      "type": "array",
      "items": {
        "$ref": "file:src/core/schemas/data_element_schema.json"
      }
    }
  },
  "required": ["title", "description", "parameters", "outputs"]
}

```

Рисунок 3.9. Шаблон `JSON Schema` для валидации описания алгоритма.

3.1.3. Разработка API Онлайн-калькулятора

Основная логика API Онлайн-калькулятора реализована в файле `main.py` с использованием библиотеки `FastAPI`. `FastAPI` — это web-фреймворк языка Python для разработки HTTP API серверов со встроенными валидацией, сериализацией данных и асинхронной обработкой. `FastAPI` имеет встроенный клиент для организации модульного тестирования. Также имеется возможность автоматической генерации документации API согласно `OpenAPI` — стандарту в формате JSON для описания конечных точек API. В результате клиент может прочитать определение `OpenAPI` для любой конечной точки и автоматически определить схемы для данных, отправляемых и получаемых API продукта.

На рисунке 3.10 представлен фрагмент исходного кода из файла `main.py`, где создается объект класса `AlgorithmCollection`, представляющий набор алгоритмов. Затем создается объект приложения, которое импортируется на сервер и обрабатывает поступающие от клиентов запросы. Далее в исходном коде описана конечная точка API `/api/algorithms`, предоставляющая данные об имеющихся алгоритмах.

В файле `main.py` реализовано логирование с использованием стандартной библиотеки `logging`, в зависимости от установленного в конфигурации уровня логирования может проводиться логирование всех поступающих запросов либо только возникающих ошибок.

Типы входных и выходных данных для конечных точек API описаны как классы с использованием библиотеки `Pydantic` в файле `api_models.py`. На рисунке 3.11 представлен класс `AlgorithmDefinition`, описывающий алгоритм в том формате, в котором его возвращает API Онлайн-калькулятора. Класс `AlgorithmDefinition` наследует от класса `BaseModel` библиотеки `Pydantic` и реализует автоматическую валидацию данных, имеет встроенные методы JSON сериализации и десериализации.


```

algorithms = AlgorithmCollection(path_config, algorithm_config, log_config)

app = FastAPI()
web_config = config['web_config']
app.add_middleware(
    CORSMiddleware,
    allow_origins=web_config['cors']['origins'],
    allow_credentials=web_config['cors']['credentials'],
    allow_methods=web_config['cors']['methods'],
    allow_headers=web_config['cors']['headers']
)

@app.get(ALGORITHMS_ENDPOINT)
✓ async def get_algorithms() -> Algorithms:
    """Возвращает список имеющихся алгоритмов.

    :return: список имеющихся алгоритмов.
    :rtype: Algorithms
    """
    logger.info('Request received')
    res = Algorithms(algorithms=[])
    for name, title in algorithms.get_name_title_dict().items():
        res.algorithms.append(AlgorithmTitle(name=name, title=title))
    return res

```

Рисунок 3.10. Исходный код реализации API.

```

class AlgorithmDefinition(BaseModel):
    """Класс представляет описание алгоритма.

    :param name: уникальное имя алгоритма;
    :type name: str
    :param title: название алгоритма;
    :type title: str
    :param description: описание алгоритма;
    :type description: str
    :param parameters: список описаний входных данных алгоритма;
    :type parameters: list[DataDefinition]
    :param outputs: список описаний выходных данных алгоритма.
    :type outputs: list[DataDefinition]
    """
    name: str
    title: str
    description: str
    parameters: list[DataDefinition]
    outputs: list[DataDefinition]

```

Рисунок 3.11. Класс, описывающий алгоритм для API.

На рисунке 3.12 представлен фрагмент автоматически генерируемой документации по стандарту OpenAPI, в том числе описание разработанных конечных точек и структуры данных, принимаемых и возвращаемых API.

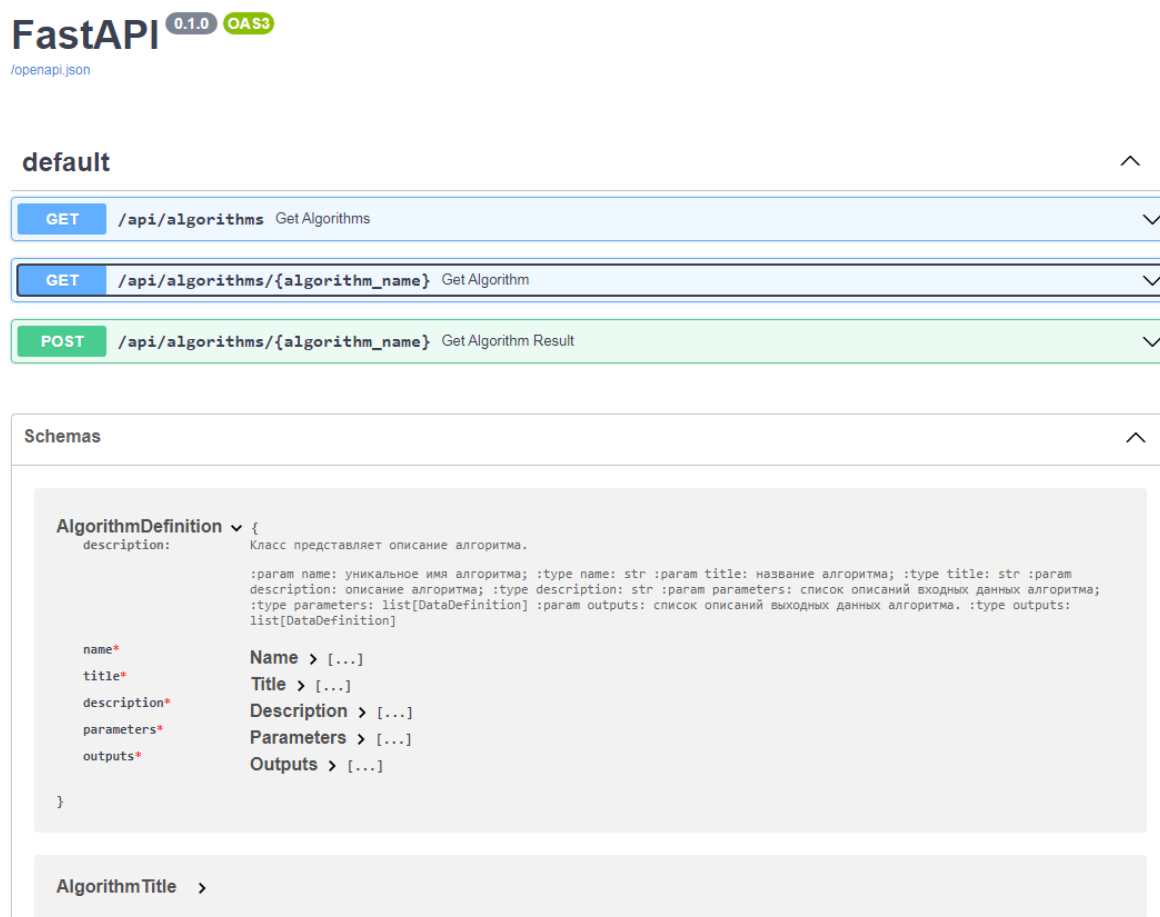


Рисунок 3.12. Автоматическая документация API.

3.1.4. Создание docker-образа для серверной части Онлайн-калькулятора

«Docker — это проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых автономных контейнеров, выполняемых в облаке или локальной среде».

Для развертывания на продуктивном и тестовом стендах серверная часть Онлайн-калькулятора должна быть упакована в docker-образ со всеми своими зависимостями и конфигурациями. Для запуска сервера на основе собранного образа, создается docker-контейнер.

Для создания docker-образа используется специальный текстовый файл Dockerfile, содержащий инструкции по сборке образа. В корень проекта добавлен файл Dockerfile представленный на рисунке 3.13.

```

FROM python:3.9
WORKDIR /code
COPY src/requirements.txt /code/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt
COPY . /code
CMD ["uvicorn", "src.main:app", "--host", "0.0.0.0", "--port", "8080"]

```

Рисунок 3.13. Dockerfile для сборки образа программного модуля.

Образ серверной части Онлайн-калькулятора собирается на основе образа python:3.9. В образ устанавливаются зависимости проекта, перечисленные в файле requirements.txt. Каталоги и файлы проекта копируются в каталог образа /code. Точкой входа является команда запуска uvicorn-сервера с разработанным приложением.

Развертывание контейнера с серверной частью Онлайн-калькулятора на продуктивном и тестовом стендах описывается в файлах docker-compose.prod.yml и docker-compose.yml соответственно. Содержание файла docker-compose.prod.yml представлено на рисунке 3.14. Контейнер запускается в режиме автоматического перезапуска в случае возникновения ошибки. В контейнер монтируется каталог для записи логов, расположенный на сервере.

```

version: '3.7'

services:
  backend_prod:
    container_name: BackendProd
    restart: always
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - '5556:8080'
    volumes:
      - backend_prod_logs:/code/logs

volumes:
  backend_prod_logs:

```

Рисунок 3.14. Docker-compose файл для развертывания контейнера.

3.1.5. Разработка алгоритмов для Онлайн-калькулятора с участием студентов

После разработки ядра и API серверной части Онлайн-калькулятора необходимо разработать начальный набор алгоритмов для его наполнения. Основной целью разработки Онлайн-калькулятора является его использование в образовательных целях. Предполагается, что студенты могут принять участие в поддержке и развитии Онлайн-калькулятора посредством разработки алгоритмов. С учетом изложенного разработку начального набора алгоритмов целесообразно организовать с участие студентов, проверив тем самым возможности использования Онлайн-калькулятора для решения учебных задач.

Для участия в разработке были приглашены четверо студентов бакалавриата - один с направления «Программная инженерия» и трое с направления «Бизнес-информатика». Учетные записи студентов на GitHub были добавлены в состав организации ОММАТ, студенты включены в команду algo-developers. Для команды настроены права доступа на просмотр репозитория с документацией проекта и права на внесение изменений в репозиторий с исходным кодом серверной части Онлайн-калькулятора.

Совместная разработка алгоритмов была организована в виде двухнедельного спринта по проекту. В начале спринта было проведено общее собрание участников в видеоконференции Zoom, ознакомление с проектом и целями спринта. На первую неделю поставлена общая задача – подобрать такой набор алгоритмов, реализация которого позволит провести интеграционное тестирование Онлайн-калькулятора. То есть в наборе должны присутствовать алгоритмы с различными вариантами входных и выходных данных. Студентам было предложено выбрать алгоритмы, представленные в прототипе, либо предложить собственные варианты. Кроме того, индивидуальным заданием на первую неделю было клонирование репозитория с исходным кодом на собственный компьютер, установка необходимых зависимостей и запуск сервера с Онлайн-калькулятором на своем компьютере.

Алгоритмы для реализации выбирались максимально простые, так как они предназначены, прежде всего, для тестирования Онлайн-калькулятора и важным является проверить принципиальную возможность участия студентов в разработке алгоритмов для Онлайн-калькулятора.

По итогам первой недели, для реализации выбраны следующие алгоритмы с различными вариантами входных и выходных данных:

- Числа Фибоначчи - скалярный целочисленный параметр - номер числа, скалярный целочисленный результат - значение заданного числа Фибоначчи.
- Список чисел Фибоначчи - скалярный целочисленный параметр - номер числа, результат список целых чисел - список чисел Фибоначчи от 1 до заданного.
- Расход топлива для поездки на заданное расстояние, входные параметры с плавающей точкой и логический, выходные параметры с плавающей точкой.
- Вычитание двух матриц - два входных параметра, матрицы с целыми числами, один результат, матрица с целыми числами.
- Подсчет количества подстрок входной строки - входной параметр строка, выходной параметр - целое число.
- Поиск корней квадратного уравнения - входные параметры три числа с плавающей точкой, на выходе строка со значениями корней или с сообщением, о том, что действительных корней нет.
- Проверка ряда чисел на совершенность - входные параметры список целых чисел, выходные параметры - список целых чисел и логическое значение.

Для настройки и запуска сервера с Онлайн-калькулятором на своем компьютере студентам направления «Бизнес-информатика» потребовалась консультация.

На вторую неделю для каждого студента была создана индивидуальная задача на разработку алгоритма. Каждый алгоритм разрабатывается в отдельном каталоге внутри каталога `src/algorithms` и представляет собой набор из трех файлов. Для реализации алгоритма необходимо разработать файл с его описанием в формате JSON, где указывается название, описание, входные и выходные данные алгоритма. Формат JSON достаточно прост, и проблем с созданием файла описания алгоритма у студентов не возникло. Пример описания алгоритма Фибоначчи в формате JSON представлен на рисунке 3.15.

```

{
  "title": "N-е число Фибоначчи",
  "description": "Числа Фибоначчи - последовательность чисел, каждый член которой равен сумме двух предыдущих.\nВведите порядковый номер числа Фибоначчи и калькулятор выдаст вам соответствующее значение.",
  "parameters": [
    {
      "name": "n",
      "title": "Номер числа Фибоначчи",
      "description": "Введите целое положительное число",
      "data_type": "INT",
      "data_shape": "SCALAR",
      "default_value": 1
    }
  ],
  "outputs": [
    {
      "name": "result",
      "title": "Число Фибоначчи",
      "description": "Число Фибоначчи с номером n",
      "data_type": "INT",
      "data_shape": "SCALAR",
      "default_value": 1
    }
  ]
}

```

Рисунок 3.15. Описание алгоритма.

Помимо описания алгоритма необходимо разработать функцию, которая позволит запускать алгоритм на выполнение, принимающую такие параметры и возвращающую такие результаты, которые указаны в файле описания. Необходимость соответствия функции описанию алгоритма пришлось подробно разъяснять студентам. Пример реализации функции для расчета числа Фибоначчи представлен на рисунке 3.16.

В третьем файле для алгоритма размещаются модульные тесты, проверяющие корректность его работы. При сборке в состав Онлайн-калькулятора проверяется успешное выполнение модульных тестов для каждого алгоритма. Кроме того, модульные тесты алгоритмов важны для использования Онлайн-калькулятора в учебных целях. Задание на разработку алгоритма может подразумевать наличие заготовки для алгоритма в виде набора файлов с описанием, функцией и тестами, где не реализована только сама функция. Таким образом, студенты могут разрабатывать алгоритмы согласно подготовленному описанию и модульным тестам без учета особенностей устройства Онлайн-калькулятора.

```
def fibonacci(n: int) -> int:
    if n == 1 or n == 2:
        return 1
    return fibonacci(n - 1) + fibonacci(n - 2)


def main(n: int):
    return {'result': fibonacci(n)}

if __name__ == '__main__':
    num = 10
    print(f'n = {num}, n-е число Фибоначчи = {fibonacci(num)}')
```

Рисунок 3.16. Реализация функции для алгоритма.

Каждый студент создал в репозитории отдельную ветку для работы по задаче. Работая параллельно, студенты разработали необходимые алгоритмы и создали заявки на внесение изменений в общую ветку. Предлагаемые изменения были проверены с помощью автоматических DevOps процессов и после исправления выявленных недочетов переданы на проверку. Пример работы по замечаниям проверки представлен на рисунке 3.17. После устранения замечаний проверок и одобрения изменений они были внесены в общую ветку, что представлено на рисунке 3.18 и обновлены на соответствующем стенде с помощью автоматических DevOps процессов.

В результате в отведенные две недели все поставленные задачи были выполнены – выбран набор алгоритмов, которые необходимо реализовать, указанные алгоритмы разработаны с участием студентов, все доработки автоматически проверены и собраны в общую ветку репозитория с исходным кодом и обновлены на тестовом стенде Онлайн-калькулятора. Все этапы процесса совместной разработки были организованы на сервисе GitHub, в том числе управление проектом.


 MikhailovAlexander reviewed on May 4 View reviewed changes

src/algorithms/quadratic_equation/function.py Hide resolved


```


10 +     if discriminant < 0:
11 +         return 'действительных корней нет, т. к. D < 0'
12 +     elif discriminant == 0:
13 +         x = -b / (2 * a)

```


 MikhailovAlexander on May 4 • edited Member ...


если результат 0, а коэффициенты а и b положительные, то в результате получается минус ноль




 Reply...

Unresolve conversation MikhailovAlexander marked this conversation as resolved.

 Исправлено имя ключа, добавлены проверки на Type и Value errors. В те... ✓ 54acd2e

 YourBestSolution commented on May 5 Member Author ...

Исправила параметры в функции assertRaisesRegex, экранировала возведение в степень. Добавила условие, чтобы выдавался 0.0, а не -0.0, когда уравнения вида $a * x^2 = 0$






 code style + 2 строки после подключения модуля ✓ 18a782f


Рисунок 3.17. Устранение замечаний к исходному коду.


Merged Task 11 #12
MikhailovAlexander merged 6 commits into `Test` from `task-11` on May 5


  MikhailovAlexander approved these changes on May 5 View reviewed changes


MikhailovAlexander left a comment • edited Member ...

Все отлично! Замечания устранены, тесты пройдены. Можно вливать в Test.










 YourBestSolution closed this on May 5


 YourBestSolution reopened this on May 5

 sonarcloud (bot) commented on May 5 ...


Kudos, SonarCloud Quality Gate passed! Passed

-  0 Bugs
-  0 Vulnerabilities
-  0 Security Hotspots
-  0 Code Smells
-  87.7% Coverage
-  0.0% Duplication



 MikhailovAlexander merged commit 61fcb4 into `Test` on May 5 View details Revert

12 checks passed

 Pull request successfully merged and closed Delete branch

You're all set—the `task-11` branch can be safely deleted.

Рисунок 3.18. Одобрение изменений исходного кода.

3.2. Разработка клиентской части онлайн калькулятора

Разработка клиентской части онлайн калькулятора осуществлялась при помощи фреймворка React с использованием библиотеки MUI. Первым этапом были разработаны статические страницы согласно прототипу и техническому заданию. После одобрения качества верстки и внесения небольших правок было осуществлено подключение методов API серверной части онлайн калькулятора.

Вторым этапом были спроектированы три основных элемента управления: галочка для работы с булевым типом; поле для ввода строк, чисел как целых, так и дробных, и перечислений чисел или слов; матричный элемент управления для отображения полей матрицы $n*m$ и их поля для ввода. Пример реализации элемента управления для работы с булевым типом представлен на рисунке 3.19.

Третьим этапом была настроена динамическая отрисовка страницы алгоритма согласно JSON структуры полученной от серверной части. При этом когда выполняется отрисовка, на странице появляются как входные элементы управления алгоритма, так и выходные, в которых будут отображаться полученные результаты выполнения алгоритма.

Четвертым этапом была произведена настройка сбора всех введенных значений из элементов управления в JSON после нажатия на кнопку «Получить результат», а затем отправка полученных JSON-данных на серверную часть для получения результата выполнения алгоритма. Пятым этапом было реализовано получение ответа от серверной части в формате JSON. Далее был осуществлен разбор полученного ответа и вывод всех результатов в соответствующие элементы управления для отображения ответов. Также для удобства были добавлены кнопки поделиться страницей нашего алгоритма или главной страницы в Телеграмм и ВКонтакте, а также возможность переходить на главную страницу при нажатии на кнопку Главная в меню или на логотип, или на заголовок онлайн калькулятора. Также на этом этапе были исправлены незначительные ошибки, найденные тестировщиками.

Пример реализованной страницы с алгоритмом вычитания матриц представлен на рисунке 3.20.

```

// Libraries
import * as React from 'react';

// Components
import FormControlLabel from '@mui/material/FormControlLabel';
import Checkbox from '@mui/material/Checkbox';
import Tooltip from '@mui/material/Tooltip';

// Types
type BooleanInputTypes = {
  description?: string;
  title?: string;
  callback?: () => void;
  isDisabeld?: boolean;
  isDefaultChecked?: boolean;
  isChecked?: boolean;
  size?: string;
  id?: string;
  data_shape?: string;
  data_type?: string;
};

const BooleanInput = ({ description, title, isDisabeld, isChecked,

  return (
    <>
      <Tooltip title={description} arrow placement="left">
        <FormControlLabel
          control={
            <Checkbox
              checked={isChecked}
              defaultChecked={isDefaultChecked}
              disabled={isDisabeld}
              size={size}
              id={id}
            />
          }
          label={title}
        />
      </Tooltip>
    </>
  )
};

```

Рисунок 3.19. Пример реализации контролла.



Главная

Расход топлива для поездки на заданное расстояние

Количество подстрок в строке

Корни квадратного уравнения

Вычитание матриц

Проверка ряда чисел на совершенность

Числа Фибоначчи

N-е число Фибоначчи

Вычитание матриц

Вычитание матриц

Размер матрицы

Строки
3

X

Столбцы
3

Размер матрицы

Строки
3

X

Столбцы
3

ПОЛУЧИТЬ РЕЗУЛЬТАТ

Рисунок 3.20. Разработанная страница с алгоритмом вычитания матриц.

3.3. DevOps

Первым этапом был арендован домен, на котором впоследствии будут работать все сервисы. Был выбран домен OMMAT.RU так как это название нашей команды. Данное доменное имя было арендовано у провайдера Timeweb, поскольку у него самые лучшие и дешевые цены на покупку и продление доменов (179 рублей первый год, 399 руб каждый последующий, при этом чем больше доменов будет на аккаунте, тем дешевле будет стоить продление).

Вторым этапом было определение конфигурации сервера и его аренда. Итоговые характеристики арендованного сервера: 1 CPU, 4 ОЗУ, 30 NVME. Расчет производился больше по ОЗУ - 2 гигабайта примерно использует GitHub Runner + по 1 гигабайту на каждый из стендов продукта (тестовый и продуктовый). После определения конфигурации выполнен поиск разных провайдеров, которые могут арендовать данную машину. Были выбраны 3 основных конкурента: Aeza, Beget и Timeweb. Цены на виртуальные машины у Aeza и Beget были примерно одинаковые, а вот у Timeweb цена оказалась ниже практически на 150 руб. После чего была запрошена скидка на конфигурацию для студенческого проекта «в качестве меры поддержки образования». В итоге Beget отказался, а Timeweb и Aeza согласились сделать скидку. У Aeza сервер без скидки был бы 622 рубля в месяц, но была предоставлена скидка в 15% и цена стала 528 руб в месяц. А Timeweb после запроса документов, подтверждающих статус студента, выдали навсегда скидку 20%. Без скидки цена за сервер была бы 485 руб в месяц, но с применением скидки цена стала 388 руб в месяц. Так как оборудование абсолютно одинаковое, то мы уточнили в чем причина разницы в стоимости и оказалось, что у Aeza пропускной канал 1Гбит в секунду, а у Timeweb 200 МБит в секунду. Но так как наша система не нуждается в больших скоростях интернет канала + мы как студенты хотели сэкономить, то было принято решение выбрать аренду сервера у Timeweb.

Итоговый сервер получил IP адрес 188.225.38.67. Сам сервер расположен в дата центре Санкт-Петербурга.

Третьим этапом было выполнено делегирование домена на NS сервера службы Cloudflare. Данная служба позволяет более удобно выполнять любые настройки, связанные с присвоением IP адресов для доменов и поддоменов. Также данная служба позволяет выполнять оптимизацию и выдачу страниц за счет дополнительного кэширование + защищает наши сайты от любых DDoS атак. После делегирования домена были определены следующие поддомены, которые будут использоваться в нашем проекте:

1. ommat.ru – сайт визитка нашей команды;
2. prod.ommat.ru – продуктивная версия нашего продукта;
3. test.ommat.ru – тестовая версия нашего продукта;
4. archi.ommat.ru – Archimate модели созданные по нашему проекту;

5. swagger.ommat.ru – документация API;
6. backend-docs.ommat.ru – документация исходного кода бэкенда;
7. mail.ommat.ru – наша доменная почта.

После определения доменов им были назначены необходимые IP адреса и CNAME-записи, по которым будут осуществляться основные маршрутизации запросов. Также был включен режим принудительной переадресации запросов с http на https, поскольку мы используем SSL сертификаты для шифрования наших запросов и безопасной передачи.

Четвертым этапом была выполнена настройка нашего сервера для автоматического развертывания всех наших приложений и осуществления их работы. В начале были обновлены все существующие пакеты в системе. Затем был изменен SSH порт подключения и созданы новые пользователи для работы с сервером. Далее был установлен запрет авторизации под root пользователем. После чего были установлены различные необходимые библиотеки для удобства работы с сервером и для улучшения его безопасности. Ну и в конце был установлен Git, Docker и Docker Compose.

Пятым этапом была выполнена настройка у всех репозиторий docker, docker compose и yaml файлов для того чтобы они могли выполнять автоматическую сборку образов наших сервисов, а также автоматическое развертывание их на сервере. Также в yaml файле были подключены автотесты для проверки работоспособности кода, а также подключен сервис SonarCloud для проверки качества и стиля кода.

Шестым этапом была произведена настройка GitHub для автоматического запуска непрерывной интеграции и непрерывной поставки (Ci\Cd) после одобрения мердж реквестов. Пример выполнения Ci\Cd процессов представлен на рисунке 3.21. Также был осуществлен первичный запуск всех контейнеров для корректной работы Ci\Cd. Так как все наши сервисы работали на разных портах, то было необходимо настроить маршрутизацию запросов внутри сервера. Для этого был развернут сервис Nginx Proxy Manager в котором были настроены все маршрутизации запросов, а также подключены SSL сертификаты к поддоменам для работы с HTTPS протоколом.

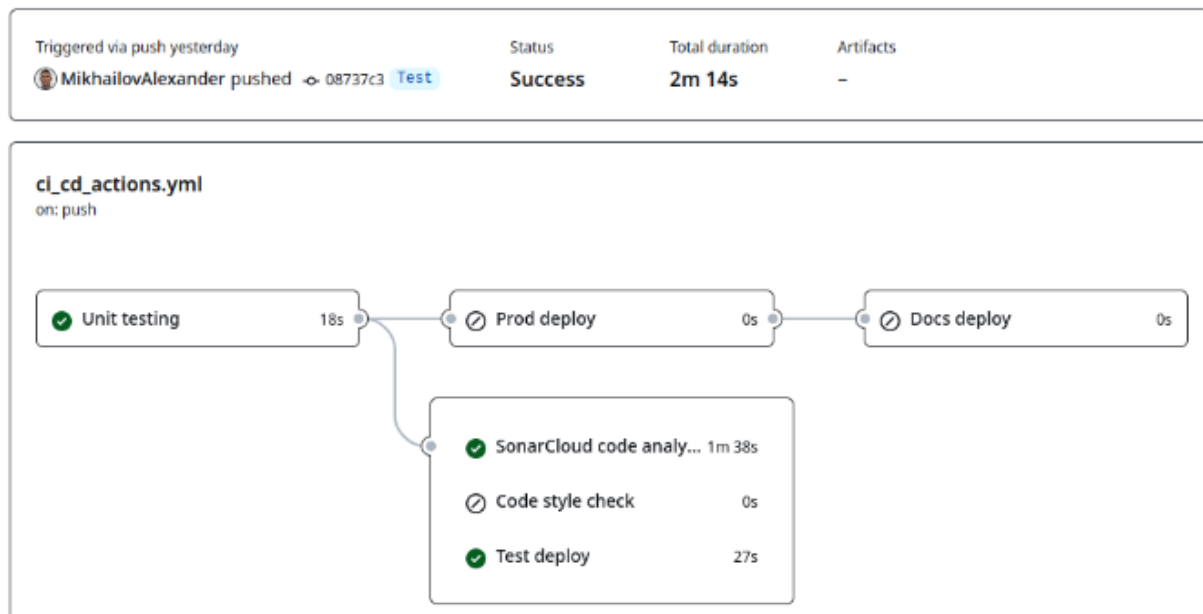


Рисунок 3.21. Выполнение процессов непрерывной интеграции и непрерывной поставки.

Глава 4. Тестирование

В текущей главе приведен разработанный план тестирования продукта онлайн калькулятора. Помимо этого, глава содержит информацию о реализации разработанного плана и о ходе тестирования.

4.1. План тестирования

Тестирование является неотъемлемой частью процесса разработки онлайн калькулятора. Оно обеспечивает правильное функционирование калькулятора, предоставляя точные и надежные результаты пользователям. Перед началом тестирования был разработан план тестирования, который включал в себя ключевые аспекты тестирования. Ниже представлен план тестирования.

1. Что надо тестировать?

Необходимо провести тестирование онлайн калькулятора, который предоставляет функционал расчета расхода топлива для поездки на заданное расстояние, подсчета количества подстрок в строке, нахождения корней квадратного уравнения, вычитания матриц, проверки ряда чисел на совершенность и решения задачи, связанных с рядом чисел Фибоначчи, а именно - генерация ряда чисел по заданной длине последовательности и определение n-ого числа ряда Фибоначчи.

2. Что будете тестировать?

В процессе тестирования необходимо проверить соответствие функциональным требованиям каждого алгоритма калькулятора. Это включает:

- расчет расхода топлива: проверка точности расчета при заданных значениях расстояния и других параметров;
- подсчет количества подстрок в строке: проверка правильности подсчета подстрок в различных строках;
- нахождение корней квадратного уравнения: проверка правильности вычисления корней для разных входных значений;
- вычитание матриц: проверка правильности операции вычитания для различных матриц;

- проверка ряда чисел на совершенность: проверка правильности определения совершенных чисел в заданном ряде;
- генерация ряда чисел Фибоначчи: проверка правильности генерации заданного количества чисел Фибоначчи;
- определение n-ого числа ряда Фибоначчи: проверка правильности вывода n-ого числа ряда Фибоначчи.

Также необходимо проверить соответствие нефункциональным требованиям, включая отзывчивость системы, ее надежность и соответствие пользовательского интерфейса требованиям.

3. Как будете тестировать?

В плане тестирования предусмотрены следующие виды тестирования:

- модульное тестирование: проведение отдельных тестов для каждого алгоритма калькулятора
- интеграционное тестирование: проверка взаимодействия различных алгоритмов и их работоспособности вместе
- функциональное тестирование: проверка соответствия каждого алгоритма функциональным требованиям
- нагрузочное тестирование: проверка производительности калькулятора при большом количестве запросов.
- тестирование пользовательского интерфейса: проверка удобства использования и соответствия интерфейса требованиям. Также будет использовано автоматизированное тестирование для повторяемых сценариев

4. Когда будете тестировать?

Тестирование будет проводиться в следующей последовательности:

- подготовка: подготовка тестового стенда, получение документации и требований, настройка тестовых сценариев (до 31.05.2023)
- тестирование: выполнение запланированных видов тестирования в соответствии с разработкой функциональности (01.06.2023 - 07.06.2023)
- анализ результатов: анализ полученных результатов, выявление и отчетность о найденных проблемах (до 09.06.2023)

5. Критерии начала тестирования:

- готовность тестовой платформы (тестового стенда)
- завершенность разработки требуемого функционала
- наличие всей необходимой документации

6. Критерии окончания тестирования:

Результаты тестирования удовлетворяют критериям качества продукта:

- требования к количеству открытых багов выполнены
- выдержка определенного периода без изменения исходного кода приложения
- выдержка определенного периода без открытия новых багов

7. Окружение тестируемой системы:

- Веб-сервер
- Программное обеспечение операционной системы Браузеры (Mozilla Firefox, Google Chrome, Apple Safari, Yandex Browser)

8. Необходимое для тестирования оборудование и программные средства:

- Тестовый стенд с доступом к веб-серверу и базе данных
- Компьютеры для запуска автоматизированных тестов
- Браузеры разных версий для проверки совместимости

9. Риски и пути их разрешения:

- Риск: Неправильные результаты расчетов.
 - Путь разрешения: Проверка алгоритмов расчетов на различных тестовых данных и сравнение результатов с ожидаемыми значениями.
- Риск: Низкая производительность при большой нагрузке.
 - Путь разрешения: Проведение нагрузочного тестирования, оптимизация алгоритмов и инфраструктуры сервера при необходимости.
- Риск: Несоответствие интерфейса требованиям.
 - Путь разрешения: Проведение тестирования пользовательского интерфейса, внесение корректировок и улучшений.

4.2. Реализация плана тестирования

В текущей части работы рассматривается реализация плана тестирования, включающего модульное тестирование, функциональное тестирование и исследовательское тестирование в соответствии с заданными критериями и сценариями.

1. Модульное тестирование:

- На этапе разработки производится модульное тестирование, при котором каждый модуль онлайн калькулятора проверяется отдельно.
- Код каждого модуля тщательно тестируется для проверки его функциональности и выявления возможных проблем или ошибок.
- Модульное тестирование помогает выявить и устранить проблемы в отдельных модулях перед интеграцией их в полную систему калькулятора.

2. Функциональное тестирование:

- Функциональное тестирование направлено на проверку того, что калькулятор соответствует заданным критериям и выполняет свои функции корректно (см. прил. Б).
- Определяются критерии и сценарии для оценки функциональности калькулятора, включая проверку валидации входных данных, выполнение расчетов и точность вывода результатов.
- Создается комплексный набор тестовых случаев, охватывающих различные сценарии и граничные случаи, чтобы убедиться, что калькулятор работает правильно в разных условиях.
- Функциональное тестирование направлено на проверку соответствия калькулятора заданным требованиям и гарантирует получение ожидаемых результатов.

3. Исследовательское тестирование:

- Помимо функционального тестирования, проводится исследовательское тестирование для оценки нефункциональных требований онлайн калькулятора.
- В ходе этого типа тестирования изучаются пользовательский интерфейс, удобство использования, производительность и совместимость калькулятора.

- Тестировщики выполняют различные действия на калькуляторе, учитывая разные пользовательские сценарии, чтобы выявить потенциальные проблемы или улучшения.
- Исследовательское тестирование помогает выявить проблемы с удобством использования, производительностью или совместимостью, которые могут повлиять на общее впечатление пользователей.

Для реализации плана тестирования разработаны сценарии тестирования, включающие различные кейсы (действия и ожидаемые результаты выполнения заданных действий), которые необходимо проверить в рамках тестирования продукта. Сценарии тестирования (см. прил. В) позволяют протестировать онлайн калькулятор на предмет соответствия функциональным требованиям. Таким образом, путем выполнения комплексной стратегии тестирования онлайн калькулятор может быть проверен на функциональность, точность и соответствие указанным требованиям. Выявленные проблемы и недостатки могут быть устранены, что позволит создать оптимизированный, надежный и удобный для пользователей онлайн калькулятора.

4.3. Модульное тестирование

Модульное тестирование позволяет автоматически проверить работу отдельных модулей системы. «Модульные тесты — это сегменты кода, которые проверяют работу других частей кода в приложении, например изолированных функций, классов и т. д. Если приложение успешно проходит все модульные тесты, то вы, по меньшей мере, уверены, что все низкоуровневые функции работают правильно».

Для организации модульного тестирования на Python используется фреймворк unittest. Файлы с исходным кодом для модульного тестирования ядра и API Онлайн-калькулятора расположены в каталогах `src/core_tests` и `src/app_tests` соответственно.

Для каждого разработанного класса создан класс с аналогичным названием и префиксом `Test`, наследующий от класса `unittest.TestCase`. Тестовый класс содержит не менее одного метода для каждого публичного метода проверяемого класса. Название метода тестирования начинается с префикса `test_`, затем следует название тестируемого метода, затем краткое пояснение критерия тестирования. Для организации юнит-тестирования используется подход AAA (`arrange`, `act`, `assert` — подготовка, действие и

проверка). Для тестов используется одинаковая структура, состоящая из трех этапов. На этапе подготовки - подготавливаются объекты и данные, необходимые для теста. На этапе действия выполняется проверяемое поведение. На третьем этапе проверяются утверждения, подтверждающие корректную работу модуля или выявляющие ошибки.

В файле `test_runner.py` все классы тест-кейсов для Онлайн-калькулятора объединяются в один тестовый набор и запускаются на выполнение, что представлено на рисунке 4.1.

```
import os
import unittest
from core_tests.algorithm_collection_tests import AlgorithmCollectionTests
from core_tests.algorithm_builder_tests import AlgorithmBuilderTest
from core_tests.algorithm_tests import AlgorithmTests
from core_tests.data_element_tests import DataElementTests
from core_tests.data_type_tests import DataTypeTests
from core_tests.data_shape_tests import DataShapeTests
from app_tests.app_tests import AppTest

if __name__ == '__main__':
    if os.path.exists(os.path.basename(__file__)):
        os.chdir('..')
        suite = unittest.TestSuite()
        suite.addTest(unittest.makeSuite(DataShapeTests))
        suite.addTest(unittest.makeSuite(DataTypeTests))
        suite.addTest(unittest.makeSuite(DataElementTests))
        suite.addTest(unittest.makeSuite(AlgorithmTests))
        suite.addTest(unittest.makeSuite(AlgorithmBuilderTest))
        suite.addTest(unittest.makeSuite(AlgorithmCollectionTests))
        suite.addTest(unittest.makeSuite(AppTest))

        runner = unittest.TextTestRunner(verbosity=2)
        runner.run(suite)
```

Рисунок 4.1. Содержимое файла `test_runner.py`.

Классы тест-кейсов для ядра Онлайн-калькулятора объединены в пакет и расположены в каталоге `src/core_tests` со следующей структурой файлов:

```
|— __init__.py
|— algorithm_builder_tests.py
|— algorithm_collection_tests.py
```

```

├── algorithm_tests.py
├── data_element_tests.py
├── data_shape_tests.py
└── data_type_tests.py

```

В файл пакета `__init__.py` вынесены все константы и параметры, используемые для тестирования. В пакете `core_tests` представлены 6 классов тест-кейсов для тестирования ядра Онлайн-калькулятора, содержащих 94 модульных теста. Тесты проверяют, как корректную работу методов тестируемого класса, так и вызываемые исключения и сообщения об ошибках. Например, на рисунке 4.2 представлен исходный код теста `test_execute`, в котором создается объект алгоритма и проверяется корректность его выполнения. На рисунке 4.3 представлен исходный код теста `test_empty_string_title`, в котором проверяется вызов исключения с соответствующим текстом ошибки, при указании пустой строки в качестве названия алгоритма.

```

def test_execute(self):
    alg = Algorithm('sum', 'sum', 'returns the sum of two numbers',
                    LOG_CONFIG_STUB)
    param_a = DataElement('a', 'a number', 'just an integer', DataType.INT,
                           DataShape.SCALAR, 1)
    param_b = DataElement('b', 'b number', 'just an integer', DataType.INT,
                           DataShape.SCALAR, 2)
    alg.add_parameter(param_a)
    alg.add_parameter(param_b)
    output = DataElement('sum', 'sum', 'the sum of two numbers',
                          DataType.INT, DataShape.SCALAR, 3)
    alg.add_output(output)
    alg.add_execute_method(lambda a, b: {'sum': a + b})
    self.assertEqual(alg.execute({'a': 10, 'b': 20}), {'sum': 30})

```

Рисунок 4.2. Тест, проверяющий выполнение алгоритма.

```

def test_empty_string_title(self):
    with self.assertRaises(ValueError) as error:
        Algorithm(NAME, '', 'description', LOG_CONFIG_STUB)
    self.assertEqual(EMPTY_STRING_PARAM_TEMPL.format('title'),
                     str(error.exception))

```

Рисунок 4.3. Тест, проверяющий вызов исключения.

Тесты для классов `AlgorithmBuilder` и `AlgorithmCollection` предусматривают создание объектов класса `Algorithm` из файлов с исходным кодом. Для соответствующих классов тест-кейсов реализовано взаимодействие с файловой системой, представленное на рисунке 4.4. Перед выполнением тестов создается каталог для вспомогательных файлов, после проведения каждого теста каталог очищается, а после завершения тестов каталог удаляется.

```
class AlgorithmBuilderTest(unittest.TestCase):
    builder = AlgorithmBuilder(DEFINITION_FILE_NAME, FUNCTION_FILE_NAME,
                               TEST_FILE_NAME, SCHEMA_FILE_PATH,
                               ALGORITHM_CONFIG, LOG_CONFIG_STUB)

    @classmethod
    def setUpClass(cls) -> None:
        if os.path.exists(os.path.basename(__file__)):
            os.chdir('../..')
        if not os.path.exists(FOLDER_PATH):
            os.mkdir(FOLDER_PATH)

    @classmethod
    def tearDownClass(cls):
        if os.path.exists(FOLDER_PATH):
            os.removedirs(FOLDER_PATH)

    def tearDown(self) -> None:
        if os.path.exists(FOLDER_PATH):
            for file in os.listdir(FOLDER_PATH):
                path = FOLDER_PATH + '/' + file
                if os.path.isfile(path):
                    os.remove(path)
                else:
                    rmtree(path)
```

Рисунок 4.4. Создание и удаление вспомогательных файлов и каталогов.

Класс `AppTest` для тестирования API Онлайн-калькулятора реализован в файле `src/app_tests/app_tests.py`. В классе `AppTest` с помощью встроенного тестового клиента фреймворка `FastAPI` проверяются все конечные точки API Онлайн-калькулятора. Данные валидируются согласно схемам JSON для классов API. Пример теста для проверки конечной точки API Онлайн-калькулятора представлен на рисунке 4.5.

Разработанный набор модульных тестов используется для тестирования и является основой процесса непрерывной интеграции по методологии DevOps. При запросе на внесение изменений в исходный код автоматически выполняются все тесты, и если хотя бы один из тестов завершился с ошибкой, изменения не могут быть перенесены в общую ветку репозитория.

```
class AppTest(unittest.TestCase):
    client = None

    @classmethod
    def setUpClass(cls) -> None:
        AppTest.client = TestClient(app)

    @classmethod
    def tearDownClass(cls):
        os.environ[IS_TEST_APP] = str(False)

    def test_get_algorithms(self):
        response = AppTest.client.get(ALGORITHMS_ENDPOINT)
        self.assertEqual(response.status_code, 200)
        self.assertIsNone(jsonschema.validate(response.json(),
                                                Algorithms().schema()))
```

Рисунок 4.5. Тестирование API Онлайн-калькулятора.

Для процесса автоматического анализа кода на сервисе SonarCloud настроен расчет метрики покрытия исходного кода модульными тестами. Согласно расчету модульными тестами покрыто 93.7% исходного кода. На сервисе SonarCloud есть возможность просмотреть детализацию покрытия кода тестами в разрезе файлов, что представлено на рисунке 4.6, а также просмотреть детализацию конкретного файла с указанием конкретных строк кода, не покрытых модульными тестами. При запросе внесения изменений в исходный код на сервисе SonarCloud автоматически рассчитывается доля покрытия модульными тестами нового кода. Отчет с проверкой SonarCloud и рассчитанными метриками автоматически публикуется в запрос изменений на GitHub.

Coverage on New Code 93.7%		New code: since previous version		
		Coverage on New Code	Uncovered Lines on New Code	Uncovered Conditions on New Code
src/test_runner.py		42.3%	12	3
src/main.py		75.7%	18	8
src/algorithms/fibonacci/function.py		76.9%	2	1
src/algorithms/fibonacci_list/function.py		83.3%	2	1
src/algorithms/fibonacci_list/tests.py		84.6%	1	1
src/algorithms/fibonacci/tests.py		84.6%	1	1
src/algorithms/matrix_sub/function.py		85.4%	4	2
src/algorithms/quadratic_equation/function.py		86.1%	4	1
src/algorithms/substring_in_a_string/tests.py		88.2%	1	1
src/core_tests/algorithm_collection_tests.py		90.0%	3	7
src/algorithms/quadratic_equation/tests.py		91.3%	1	1
src/algorithms/substring_in_a_string/function.py		91.9%	2	1

Рисунок 4.6. Детализация покрытия кода тестами в разрезе файлов.

Coverage 75.7%		New code: since previous version		
1	—	"""Реализация API для онлайн-калькулятора с использованием фреймворка FastAPI.		New code
2	—	"""		
3	—			
4	—	import os		
5	—	import json		
6	—	import logging.config		
7	—	from fastapi import FastAPI		
8	—	from fastapi.middleware.cors import CORSMiddleware		
9	—			
10	—	from src.core.algorithm_collection import AlgorithmCollection, \		
11	—	ALGORITHM_NOT_EXISTS_TEMPL		
12	—	from src.api_models import AlgorithmTitle, Algorithms, DataDefinition, \		
13	—	AlgorithmDefinition, Data, Parameters, Outputs, AnswerOutputs, \		
14	—	AnswerAlgorithmDefinition		
15	—	from src import APP_CONFIG_FILE_PATH, LOG_CONFIG_FILE_PATH, \		
16	—	PATH_CONFIG, ALGORITHM_CONFIG, IS_TEST_APP, EXECUTE_TIMEOUT, \		
17	—	ALGORITHMS_ENDPOINT, TIME_OVER_MSG		
18	—			
19	—	if os.path.exists('...' + LOG_CONFIG_FILE_PATH):	Partially covered code	
20	—	os.chdir('...')		
21	—	with open(LOG_CONFIG_FILE_PATH, 'r') as log_conf_file:		
22	—	log_conf = json.load(log_conf_file)		
23	—	file_path = None		
24	—	try:		
25	—	file_path = log_conf["handlers"][0]["file_handler"]["filename"]		
26	—	except KeyError:		
27	—	pass		
28	—	if file_path:		
29	—	folder = os.path.split(file_path)[0]		
30	—	if not os.path.isdir(folder):		
31	—	os.mkdir(folder)		

Рисунок 4.7. Детализация покрытия кода тестами для отдельного файла.

4.4. Проведение тестирования

Проведение тестирования включает выполнение плана тестирования по разработанным сценариям. В соответствии с планом, модульное тестирование проводилось на этапе разработки, чтобы проверить каждый модуль онлайн калькулятора отдельно. Это позволило выявить и исправить возможные проблемы в коде до интеграции модулей.

Затем выполнялось функциональное тестирование, где онлайн калькулятор проверялся на соответствие заданным критериям, составленным на основе

функциональных требований. В процессе тестирования были обнаружены ошибки, которые включали неправильные результаты вычислений и некорректное отображение результатов.

Найденные ошибки были задокументированы и занесены в систему отслеживания ошибок на платформе GitHub в соответствии с установленным шаблоном отчета об ошибке (рис. 4.8). Каждая задача, включающая отчет об ошибке, была назначена соответствующему разработчику для доработки. Это позволило установить ответственность за исправление ошибок и отслеживать их статус до полного разрешения. Отчеты об ошибках, содержащие подробное описание некорректной работы продукта, шаги воспроизведения ошибки и верный ожидаемый результат, были оформлены на платформе GitHub, обеспечивая прозрачность и централизованное управление процессом исправления ошибок.

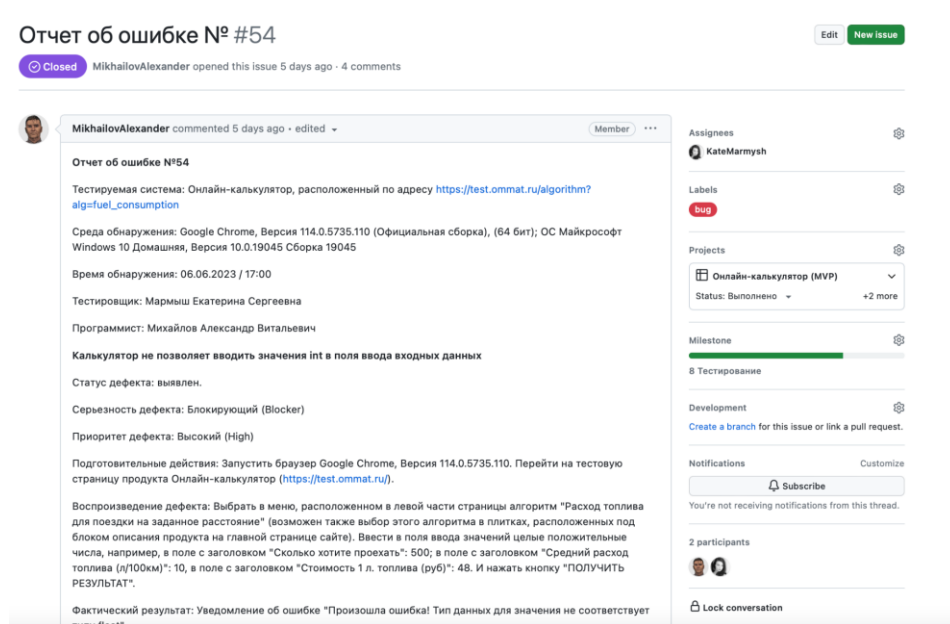


Рисунок 4.8. Оформление отчета об ошибке

Помимо функционального тестирования также было проведено исследовательское тестирование, позволяющее оценить нефункциональные требования калькулятора. Был изучен пользовательский интерфейс, а также онлайн калькулятор был проверен с точки зрения удобства использования, производительности и совместимости с различными платформами. Это позволило выявить потенциальные проблемы и предложить соответствующие улучшения для повышения качества разрабатываемого продукта.

Заключение

В текущей части работы представлено описание ролей каждого участника проектной команды и оценка индивидуальных результатов выполнения проекта, сформированных / развитых компетенций.

Овчинникова Аделина

Роль: аналитик, технический писатель

В рамках проекта выполнила следующие задачи:

- Анализ предметной области в использовании онлайн калькуляторов: возможности пользователей, функции и основное поведение систем онлайн калькуляторов.
- Анализ понятия алгоритма в разрезе применения в онлайн калькуляторов.
- Построение концептуальной модели алгоритма.
- Проектирование диаграммы USE-CASE, описание спецификаций использования.
- Составление технического задания, руководство пользователя.

Участие в проекте позволило улучшить навык в области постановки требований к программному продукту, проектирования USE-CASE и описания спецификаций. Практический опыт в составлении руководства пользователя.

Мармыш Екатерина

Роль: аналитик, тестировщик

В ходе работы над проектом я выполняла роли аналитика и тестировщика. В качестве аналитика я провела поиск и анализ существующих онлайн калькуляторов. Результаты выполнения этой задачи представлены в пункте 1.2 текущего отчета. В качестве тестировщика были выполнены следующие работы: разработка плана тестирования, разработка сценариев функционального тестирования, проведение тестирования. Помимо исследования продукта, в соответствии с разработанной комплексной стратегией тестирования, в случае нахождения ошибок в работе онлайн калькулятора осуществлялось взаимодействие с разработчиками с целью устранения выявленных неполадок. Взаимодействие выполнялось на платформе GitHub путем составления отчета об ошибке и постановки задачи на ответственного программиста. По

завершении исправления выявленных багов / ошибок выполнялось повторное тестирование продукта для подтверждения корректности работы продукта. Описание тестирования разработанного программного продукта представлено в 4 главе текущего отчета.

В рамках работы над проектом я успешно справилась с возложенными на меня обязанностями, включающими выполнение аналитических задач и проведение тестирования продукта.

В ходе выполнения проекта были развиты такие компетенции, как:

- Способность проводить исследования, организовывать самостоятельную и коллективную научно-исследовательскую работу для поиска, выработки и применения новых решений в области информационно-коммуникационных технологий;
- Способность анализировать и интерпретировать разработанные требования к программному продукту, являющегося результатом выполнения работы над проектом, для составления стратегии проведения комплексного тестирования;
- Способность проводить тестирование продукта в соответствии с разработанной стратегией, включающей план и сценарии тестирования, которые обеспечивают полное покрытие критериев соответствия продукта разработанным функциональным требованиям;
- Способность управлять взаимодействием с участниками проектной команды в процессе решения задач учебной деятельности;
- Способность определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки.

Таким образом, мое участие в проекте существенно расширило спектр моих знаний в области разработки программных продуктов, что важно для моего профессионального роста и может способствовать успешной реализации подобных проектов в будущем.

Михайлов Александр

Роль: менеджер проекта, Backend-разработчик

В ходе работы над проектом выполнил следующие задачи:

- Проведение анализа предметной области проведения семинарских занятий и разработка ArchiMate-модели архитектуры предметной области.

- Создание на сервисе GitHub организации ОММАТ, проекта для разработки Онлайн-калькулятора и набора репозиторий для документации и исходного кода проекта.
- Организация проектной работы команды студентов магистратуры.
- Проектирование серверной части Онлайн-калькулятора.
- Разработка серверной части Онлайн-калькулятора.
- Разработка набора модульных тестов серверной части Онлайн-калькулятора.
- Организация работы студентов бакалавриата по разработке алгоритмов для Онлайн-калькулятора.

Участие в проекте позволило развить компетенции организации коллективной работы, планирования процессов управления жизненным циклом ИТ-проекта.

Анненкова Валерия

Роль: аналитик, дизайнер

В ходе работы над проектом выполнила следующие задачи:

- Создание прототипа в инструменте Figma для визуализации концепции и функциональности продукта.
- Разработка логотипа, отражающего бренд и ценности проекта.
- Проведение анализа компетенций для ИТ-профессий в профессиональных и образовательных стандартах.
- Изучение ожиданий работодателей в отношении выпускников ИТ-специальностей.
- Анализ востребованных технологий и навыков для ИТ-профессий на основе данных НН.
- Анализ объема и динамики рынка труда в ИТ в период с 2020 по 2023 годы.
- Изучение тенденций командной работы и перехода на удаленную работу в ИТ-сфере.
- Создание презентации с результатами проекта

Участие в проекте в роли аналитика-дизайнера позволило мне значительно развить мои компетенции в организации самостоятельной работы и планировании процессов управления жизненным циклом ИТ-проекта. Взаимодействие с командой разработчиков, аналитиков и других специалистов помогло мне научиться эффективно координировать

и распределять свои задачи, а также планировать и контролировать процессы разработки. Это включало определение требований, создание прототипов и дизайна, анализ тенденций в ИТ-сфере и адаптацию продукта под ожидания пользователей. Полученный опыт позволит мне успешно применять свои навыки в будущих проектах и стремиться к постоянному развитию своих компетенций.

Трефилов Дмитрий

Роль: DevOps, Frontend-разработчик

В ходе работы над проектом выполнил следующие задачи:

- Проведение анализа провайдера для аренды домена OMMAT.RU и VPS сервера.
- Настройка доменов и поддоменов с автоматическим редиректом на HTTPS протокол.
- Подключение DDoS защиты.
- Настройка VPS сервера.
- Настройка Ci\Cd процессов.
- Подключение и настройка Nginx Proxy Manager для корректной маршрутизации запросов внутри сервера.
- Разработка клиентской части системы.

Участие в проекте позволило развить компетенции DevOps инженера, а также отточить навыки работы с React.

Отдельно стоит отметить, что в рамках работы над проектом каждый член команды был задействован на всех этапах на пути к достижению поставленной цели. По завершении выполнения каждого индивидуального задания, участники проектной группы вносили результаты своей работы в проект на платформе GitHub. Остальные члены команды, в свою очередь, имели возможность ознакомиться с полученными результатами и при необходимости предложить корректировки. В случае, если замечаний к выполненной работе нет, внесенные изменения с результатами работы согласовывались и задача закрывалась, то есть считалась выполненной.

Результат проекта

В результате выполнения проекта поставленная цель, заключающаяся в разработке программного продукта «Онлайн-калькулятор» в MVP версии, была достигнута посредством последовательного выполнения всех задач, стоящих перед участниками проектной команды.

Для организации групповой работы над проектом на платформе GitHub была создана Организация, в которую был заведен проект по разработке продукта, отражающий ход выполнения всех этапов работы и ответственных за выполнение каждой задачи лиц.

В результате работы над проектом созданы публичные репозитории для документации и исходного кода программного продукта с настроенными механизмами сборки и развертывания. Важно отметить, что архитектура продукта позволяет расширять его функциональность с минимальными усилиями.

Ресурсы проекта включают следующие элементы:

1. Визитка команды¹
2. Репозиторий² с документацией проекта, в том числе:
 - UML-модель³
3. Репозиторий⁴ с ArchiMate-моделью, которая автоматически собирается в HTML-отчет и публикуется:
 - ArchiMate-модель⁵ онлайн
4. Репозиторий⁶ с исходным кодом для серверной части онлайн-калькулятора:
 - Тестовый контур⁷ серверной части онлайн-калькулятора
 - Документация API⁸ серверной части онлайн-калькулятора

¹ <https://ommat.ru/>

² <https://github.com/OMMAT-HSE/algoscalc-docs>

³ <https://github.com/OMMAT-HSE/algoscalc-docs/tree/Prod/uml-model>

⁴ <https://github.com/OMMAT-HSE/algoscalc-archi>

⁵ <https://archi.ommat.ru/>

⁶ <https://github.com/OMMAT-HSE/algoscalc-back>

⁷ <https://test.ommat.ru/api/algorithms>

⁸ <https://swagger.ommat.ru/docs>

- Документация исходного кода⁹ серверной части онлайн-калькулятора
 - Продуктовый контур¹⁰ серверной части онлайн-калькулятора
5. Репозиторий¹¹ с исходным кодом для клиентской части онлайн-калькулятора:
- Тестовый контур¹² клиентской части онлайн-калькулятора
 - Продуктовый контур¹³ клиентской части онлайн-калькулятора

Подводя итог, можно заключить, что команда ОММАТ, участники которой являются студентами магистратуры НИУ ВШЭ в Перми, успешно реализовала проект по разработке Онлайн калькулятора. В ходе выполнения работы были задействованы все участники проектной команды. Разработанный программный продукт в дальнейшем может быть использован в рамках курсов по программной инженерии при выполнении практикоориентированных учебных задач.

⁹ <https://backend-docs.ommat.ru/index.html>

¹⁰ <https://prod.ommat.ru/api/algorithms>

¹¹ <https://github.com/OMMAT-HSE/algoscalc-front>

¹² <https://test.ommat.ru/>

¹³ <https://prod.ommat.ru/>

Список использованных источников

1. Cammeraat, E. and M. Squicciarini (2021), «Burning Glass Technologies' data use in policy-relevant analysis: An occupation-level assessment", OECD Science, Technology and Industry Working Papers, No. 2021/05. OECD Publishing, Paris. DOI: 10.1787/cd75c3e7-en
2. Бакалаврская программа «Программная инженерия». Алгоритмы и структуры данных - URL - <https://perm.hse.ru/ba/se/courses/646542111.html> (дата обращения 12.04.2023)
3. В.И. Загвязинский, А.Ф. Закирова, Т.А. Строкова и др., Педагогический словарь : учеб. пособие для студ. высш. учеб. заведений, М. : Издательский центр «Академия», 2008 - 352 с.
4. Шалтунович, А. В. Организация совместной разработки веб-приложений в рамках социальной сети GITHUB / А. В. Шалтунович // Вестник Нижневартковского государственного гуманитарного университета. – 2011. – № 3. – С. 86-89.
5. ArchiMate® 3.1 Specification - URL - <https://pubs.opengroup.org/architecture/archimate3-doc/> (дата обращения 12.04.2023)
6. Калевко, В. В. Управление образовательной программой вузов в контексте подготовки конкурентоспособных разработчиков программного обеспечения / В. В. Калевко, Д. Г. Лагерева, А. Г. Подвесовский // Современные информационные технологии и ИТ-образование. – 2018. – Т. 14, № 4. – С. 803-814.
7. Никонова, Е. З. Как и чему учить будущих звезд ИТ? // БИТ. БИЗНЕС & ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. – 2020. – № 7(100). – С. 38-52.
8. Kevin Gray, Andrea Koncz, The Key Attributes Employers Seek on Students' Resumes // National Association of Colleges and Employers - URL: <https://www.nacweb.org/about-us/press/2017/the-key-attributes-employers-look-for-on-students-resumes/> (Дата обращения 14.04.2023)
9. Barath Roy Michel, Manivannan Kannan, Aarthi Rajam Subramanian, Piyusha Lokre, Gianfranco Alvarado A Study on Skills Gap: Beyond COVID // Journal of Innovation in Polytechnic Education. - 2022. - № 4 (1). – С. 57-61.
10. The MIT License // The Open Source Initiative (OSI) - URL - <https://opensource.org/licenses/mit/> (дата обращения 15.04.2023)

11. Трибунский, А. А. Применение devops технологий при обучении программированию / А. А. Трибунский, Д. С. Горелко // Интеграция науки, технологии и образования: ИНТО - 2022 : Материалы VII межрегиональной конференции молодых исследователей с международным участием, Москва, 20 апреля 2022 года / Под общей редакцией Е.А. Вахтоминой. – Москва: Московский педагогический государственный университет, 2022. – С. 266-270.
12. Лазарева, Н. Б. Оптимальный подход к разработке программного обеспечения с использованием современных методологий и технических средств / Н. Б. Лазарева // Инженерный вестник Дона. – 2020. – № 10(70). – С. 54-63.
13. Коляда, Н. В. DEVOPS как механизм быстрого погружения в ИТ-проект / Н. В. Коляда // Молодёжь третьего тысячелетия : Сборник научных статей, Омск, 10–28 апреля 2017 года. – Омск: Омский государственный университет им. Ф.М. Достоевского, 2017. – С. 820-822.
14. Гороховская, Н. А. Модель формирования социальной компетентности у будущих программистов в условиях коллаборативного электронного обучения / Н. А. Гороховская, Л. Н. Рулиене // Проблемы современного образования. – 2018. – № 5. – С. 209-217.
15. Devprom ALM - управление процессом разработки ПО // ООО «Девпром» - URL - <https://devprom.ru/company/products/> (Дата обращения 15.04.2023)
16. Никонова, Е. З. Методологии управления программными проектами в подготовке ИТ-специалистов / Е. З. Никонова // . – 2018. – Т. 9, № 2-2. – С. 167-173.
17. Руководство по eduScrum // Команда eduScrum Россия - URL - https://eduscrum.com.ru/wp-content/uploads/2020/04/The_eduScrum_Guide_RU_2_0.pdf (дата обращения 15.04.2023)
18. Лукашенко, М. А. Научить студента думать: SCRUM как метод продуктивного обучения в учебном заведении / М. А. Лукашенко, Т. В. Телегина // Азимут научных исследований: педагогика и психология. – 2019. – Т. 8, № 2(27). – С. 138-142.
19. Алексеев, Г. С. Использование методологии AGILE в командной разработке проекта «Экология» с применением системы контроля версий GIT / Г. С. Алексеев, В. В. Лавров, И. А. Гурин // Теплотехника и информатика в образовании, науке и производстве : Сборник докладов VI Всероссийской научно-практической

- конференции студентов, аспирантов и молодых учёных (ТИМ'2017) с международным участием, Екатеринбург, 11–12 мая 2017 года / Редакционная коллегия сборника докладов: Спирин Н. А., Лавров В. В., Бурыкин А. А., Воронов Г. В., Гольцев В. А., Гурин И. А., Казяев М. Д., Киселев Е. В., Куделин С. П., Лошкарев Н. Б., Матюхин В. И., Носков В. Ю., Швыдкий В. С., Ярошенко Ю. Г.. – Екатеринбург: Уральский федеральный университет имени первого Президента России Б.Н. Ельцина, 2017. – С. 184-187.
20. Барвенов, С. А. Опыт использования GIT и GITHUB при проведении занятий со студентами / С. А. Барвенов, А. А. Станкевич // Веб-программирование и интернет-технологии WebConf 2015 : Материалы 3-й Международной научно-практической конференции, Минск, 12–14 мая 2015 года / Белорусский государственный университет, механико-математический факультет. – Минск: Белорусский государственный университет, 2015. – С. 44-46.
21. Маклецов, С. В. Интеграция учебной и профессиональной деятельности в подготовке студентов ИТ-направлений на основе сервиса GITHUB / С. В. Маклецов, Т. А. Старшинова, Р. Н. Зарипов // Управление устойчивым развитием. – 2020. – № 5(30). – С. 100-104.
22. Сидякин, И. М. Применение системы контроля версий GitLab для обучения программированию / И. М. Сидякин // Наука и образование: научное издание МГТУ им. Н.Э. Баумана. – 2016. – № 10. – С. 168-179.
23. Гаспарян, А. В. Совместная разработка ПО с использованием GIT / А. В. Гаспарян, Н. В. Тимошина // ИТПОРТАЛ. – 2017. – № 1(13). – С. 3.
24. Кукарцев, В. В. Сравнение систем контроля версий: Git, Mercurial, CVS и SVN / В. В. Кукарцев, С. А. Бадарчы // СИНЕРГИЯ НАУК. – 2018. – № 19. – С. 538-548.
25. Грузин, Н. А. Обзор и сравнение хостингов для git-репозиторий: Bitbucket, Github и Gitlab / Н. А. Грузин, А. А. Голубничий // E-Scio. – 2021. – № 1(52). – С. 588-596.
26. Иващенко, Н. Н. Непрерывная интеграция программного кода с помощью инструментов Docker и github Actions / Н. Н. Иващенко, С. С. Емельянова // Научно-технический вестник Поволжья. – 2022. – № 12. – С. 286-289.
27. Онлайн калькулятор Math. URL: <https://math.semestr.ru/> (дата обращения: 02.05.2023).

28. Онлайн калькулятор Calculat.org. URL: <https://www.calculat.org/ru/> (дата обращения: 02.05.2023).
29. Онлайн калькулятор Geleot. URL: <https://geleot.ru/> (дата обращения: 02.05.2023).
30. Онлайн калькулятор OnlineMSchool. URL: <https://ru.onlinemschool.com/math/assistance/> (дата обращения: 02.05.2023).
31. Онлайн калькулятор Zaochnik. URL: <https://zaochnik.com/online-calculators/> (дата обращения: 02.05.2023).
32. Онлайн калькулятор Calculatorium. URL: <https://calculatorium.ru/> (дата обращения: 02.05.2023).
33. Лайкова А.А. Юзабилити сайта: принципы и методы оценки // Актуальные проблемы авиации и космонавтики. – 2016. – №12. URL: <https://cyberleninka.ru/article/n/yuzabiliti-sayta-printsipy-i-metody-otsenki> (дата обращения: 02.05.2023).
34. Бакаев М.А. Современные тенденции в автоматизированной оценке юзабилити и поведенческие факторы в алгоритмах поисковых систем // Программные продукты и системы. – 2017. – №3. URL: <https://cyberleninka.ru/article/n/sovremennyye-tendentsii-v-avtomatizirovannoy-otsenke-yuzabiliti-i-povedencheskie-factory-v-algoritmah-poiskovyh-sistem> (дата обращения: 02.05.2023).
35. Генералова А. Чек-лист юзабилити сайта: как сделать сайт удобным и перестать терять клиентов. – 2021. URL: <https://roistat.com/rublog/check-list-usability/> (дата обращения: 02.05.2023).
36. Проверка скорости сайта. URL: https://pr-cy.ru/speed_test/ (дата обращения: 02.05.2023).
37. Алферова З.В. Теория алгоритмов // Учебн. пособие. - М.: Статистика, 1973. – 164 с.

Приложение А

Таблица А.1. Просмотр списка алгоритмов

Название	1. Просмотр списка алгоритмов
Актеры	Основной: Пользователь
Описание	Прецедент дает возможность просматривать список существующих алгоритмов в онлайн калькуляторе
Предварительные условия	Нет.
Постусловия	Нет.
Основной поток	1. Пользователь открывает раздел с перечнем доступных алгоритмов. 2. Система получает перечень доступных алгоритмов. 3. Система выводит перечень доступных алгоритмов. 4. Пользователь просматривает список доступных алгоритмов.
Исключения	1. Недоступность Сервера: 2. На сервере нет доступных алгоритмов.
Альтернативный поток	1. Недоступность Сервера - поток выполняется после шага 1 основного потока: а. Система выводит уведомление о недоступности сервера. 2. На сервере нет доступных алгоритмов - поток выполняется после шага 2 основного потока: а. В разделе со список алгоритмов выводится уведомление, что нет доступных для выполнения алгоритмов.
Приоритет	Высокий
Частота использования	Постоянно при использовании Системы

Таблица А.2. Выполнить алгоритм

Название	2. Выполнить алгоритм
Актеры	Основной: Пользователь
Описание	Прецедент дает возможность Пользователю запускать алгоритм на выполнение
Предварительные условия	PRE-1. Пользователь выбрал алгоритм из списка алгоритмов PRE-2. Пользователь задал входные параметры и они соответствуют требуемому формату ожидаемых входных данных
Постусловия	POST-1. Результат выполнения POST-2. Решение
Основной поток	1. Пользователь запускает алгоритм на выполнение. 2. Система выполняет расчет по выбранному алгоритму и заданным входным параметрам. 3. Система выводит результат.
Исключения	1. Недоступность Сервера 2. Ошибка во время расчета алгоритмом
Альтернативный поток	1. Недоступность Сервера - поток выполняется после шага 1 основного потока: а. Система выводит уведомление о недоступности сервера. 2. Ошибка во время расчета алгоритмом - поток выполняется после шага 2 основного потока: а. Система выводит уведомление об ошибке с текстовым описанием содержания ошибки.
Приоритет	Высокий
Частота использования	Постоянно при использовании Системы

Таблица А.3. Задать входные параметры

Название	3. Задать входные параметры
Актеры	Основной: Пользователь
Описание	Прецедент дает возможность Пользователю задать входные параметры для выбранного алгоритма
Предварительные условия	PRE-1. Пользователь выбрал алгоритм
Постусловия	Нет.
Основной поток	1. Пользователь задает входные параметры для выбранного алгоритма. 2. Система проверяет входные параметры на соответствие требованиям к входным данным для выбранного алгоритма
Исключения	1. Недоступность Сервера 2. Ошибка в входных данных
Альтернативный поток	1. Недоступность Сервера - поток выполняется после шага 1 основного потока: а. Система выводит уведомление о недоступности сервера. 2. Ошибка в входных данных - поток выполняется после шага 1 основного потока: а. Система выводит уведомление об ошибке в входных параметрах и сообщает, какие ожидаются входные параметры: тип данных, ограничения и в формат (для данных из файла)
Приоритет	Средний
Частота использования	Постоянно после выполнения выбранного алгоритма

Таблица А.4. Получить результат

Название	4. Получить результат
Актеры	Основной: Пользователь
Описание	Прецедент дает возможность Пользователю получить результат по выполненному выбранному алгоритму
Предварительные условия	PRE-1. Пользователь запустил на выполнение выбранный алгоритм
Постусловия	POST-1. Результат выполнения
Основной поток	Система выводит результат (ответ) выполненного выбранного алгоритма.
Исключения	1. Недоступность Сервера 2. Ошибка во время расчета алгоритмом
Альтернативный поток	1. Недоступность Сервера - поток выполняется после шага 1 основного потока: а. Система выводит уведомление о недоступности сервера. 2. Ошибка во время расчета алгоритмом - поток выполняется после шага 2 основного потока: а. Система выводит уведомление об ошибке с текстовым описанием содержания ошибки.
Приоритет	Средний
Частота использования	Постоянно после выполнения выбранного алгоритма

Приложение Б

Таблица Б.1. Критерии тестирования

№	Критерий	T1	T2	T3	T4	T5	T6	T7	T8
1	Функции								
1.1	Просмотр списка алгоритмов	+	+	+	+	+	+	+	+
1.2	Просмотр описания алгоритма		+	+	+	+	+	+	+
1.3	Ввод параметров		+	+	+	+	+	+	+
1.4	Просмотр информации о параметре		+	+	+	+	+	+	+
1.5	Запуск алгоритма на выполнение		+	+	+	+	+	+	+
1.6	Просмотр результата выполнения алгоритма		+	+	+	+	+	+	+
2	Алгоритмы								
2.1	N-е число Фибоначчи		+						
2.2	Числа Фибоначчи			+					
2.3	Расход топлива для поездки на заданное расстояние				+				
2.4	Вычитание матриц					+			
2.5	Количество подстрок в строке						+		
2.6	Корни квадратного уравнения							+	
2.7	Проверка ряда чисел на совершенность								+
3	Входные данные (параметры)								
3.1	Форма данных								
3.1.1	Скалярное значение				+				
3.1.2	Список						+		+
3.1.3	Матрица					+			
3.2	Тип данных								
3.2.1	Целое число		+	+					
3.2.2	Число с плавающей точкой				+	+		+	
3.2.3	Строка						+		+
3.2.4	Логическое				+				
3.3	Количество								
3.3.1	1 параметр		+	+					+
3.3.2	Больше 1 параметра				+	+	+	+	
4	Выходные данные (результаты)								
4.1	Форма данных								
4.1.1	Скалярное значение				+				
4.1.2	Список			+			+		+
4.1.3	Матрица					+			
4.2	Тип данных								
4.2.1	Целое число		+						
4.2.2	Число с плавающей точкой				+	+			
4.2.3	Строка			+			+	+	+
4.2.4	Логическое								+
4.3	Количество								
4.3.1	1 результат		+	+		+	+	+	
4.3.2	Больше 1 результата				+				+
4.4	Результат								
4.4.1	Результат выполнения алгоритма		+	+	+	+	+	+	+
4.4.2	Сообщение об ошибке		+						

Приложение В

Предварительное действие для выполнения функционального тестирования продукта:

- открыть в браузере сайт test.ommat.ru

Таблица В.1. Сценарий тестирования T1

Действие	Ожидаемый результат
Открыть главную страницу	На странице приведена основная информация о сервисе и о его возможностях: «Онлайн калькуляторы ОММАТ.ru - бесплатный сервис онлайн-калькуляторов. С помощью нашего сервиса Вы сможете без регистрации и прочих проверок быстро и точно произвести необходимые вычисления. Мы постоянно работаем над улучшением предоставляемого Вам сервиса и созданием новых калькуляторов, чтобы каждый пользователь смог оперативно и максимально точно произвести нужные расчеты. Мы прилагаем много усилий для улучшения нашего сервиса и искренне надеемся, что с помощью представленных онлайн-калькуляторов Вы смогли решить поставленные перед Вами задачи. Если Вам понравился наш сервис онлайн-калькуляторов, то добавляйте его в закладки и расскажите про него друзьям через Вашу социальную сеть."
	На странице представлен полный перечень алгоритмов* в виде плиток в центральной части страницы и в меню, расположенном в левой части страницы. *Перечень алгоритмов: -Расход топлива для поездки на заданное расстояние -Количество подстрок в строке -Корни квадратного уравнения -Вычитание матриц -Проверка ряда чисел на совершенность -Числа Фибоначчи -N-е число Фибоначчи
Скроллинг главной страницы	Страница сайта пролистывается вверх и вниз, плитки, содержащие ссылки на алгоритмы - не накладываются друг на друга

Таблица В.2. Покрытие критериев (T1)

Покрытие критериев	
1.1	Просмотр списка алгоритмов

Таблица В.3. Сценарий тестирования T2

Действие	Ожидаемый результат
Выбрать алгоритм «N-е число Фибоначчи» в меню в левой части главной страницы сайта	Выполняется переход на страницу алгоритма «N-е число Фибоначчи"
	В верхней части страницы представлено описание выбранного алгоритма: «N-е число Фибоначчи Числа Фибоначчи - последовательность чисел, каждый член которой равен сумме двух предыдущих. Введите порядковый номер числа Фибоначчи и калькулятор выдаст вам соответствующее значение."
	Непосредственно под блоком описания находится блок ввода входных данных с заголовком «Номер числа Фибоначчи». В поле ввода данных приведена информация (подсказка) о типе входных данных «Введите целое положительное

Действие	Ожидаемый результат
	<p>число». В нижней части блока ввода входных данных расположена кнопка «ПОЛУЧИТЬ РЕЗУЛЬТАТ».</p> <p>Непосредственно под блоком ввода входных данных расположен блок вывода результатов, имеющий заголовок «Число Фибоначчи». Блок вывода результатов визуально выделен более темным цветом.</p> <p>На странице представлен полный перечень алгоритмов* в меню, расположенном в левой части страницы</p> <p>*Перечень алгоритмов: -Расход топлива для поездки на заданное расстояние -Количество подстрок в строке -Корни квадратного уравнения -Вычитание матриц -Проверка ряда чисел на совершенность -Числа Фибоначчи -N-е число Фибоначчи</p>
Выбрать алгоритм «N-е число Фибоначчи» в плитках, расположенных под блоком описания инструмента на главной странице сайта	<p>Выполняется переход на страницу алгоритма «N-е число Фибоначчи»</p> <p>В верхней части страницы представлено описание выбранного алгоритма: «N-е число Фибоначчи Числа Фибоначчи - последовательность чисел, каждый член которой равен сумме двух предыдущих. Введите порядковый номер числа Фибоначчи и калькулятор выдаст вам соответствующее значение.»</p> <p>Непосредственно под блоком описания находится блок ввода входных данных с заголовком «Номер числа Фибоначчи». В поле ввода данных приведена информация (подсказка) о типе входных данных «Введите целое положительное число». В нижней части блока ввода входных данных расположена кнопка «ПОЛУЧИТЬ РЕЗУЛЬТАТ».</p> <p>Непосредственно под блоком ввода входных данных расположен блок вывода результатов, имеющий заголовок «Число Фибоначчи». Блок вывода результатов визуально выделен более темным цветом.</p> <p>На странице представлен полный перечень алгоритмов* в меню, расположенном в левой части страницы</p> <p>*Перечень алгоритмов: -Расход топлива для поездки на заданное расстояние -Количество подстрок в строке -Корни квадратного уравнения -Вычитание матриц -Проверка ряда чисел на совершенность -Числа Фибоначчи -N-е число Фибоначчи</p>
Скроллинг страницы алгоритма «N-е число Фибоначчи»	Страница сайта пролистывается вверх и вниз, поля описания, ввода и вывода данных не наезжают на друга
Ввести в поле ввода данных в качестве параметра n значение 4 и нажать кнопку «ПОЛУЧИТЬ РЕЗУЛЬТАТ»	<p>При клике на поле ввода входных данных для ввода значения параметра n подсказка, содержащая информацию о типе данных, которому должно соответствовать введенное значение ("Введите целое положительное число"), смещается вверх до границы поля ввода. Подсказка остается видна и по завершении ввода значения входного параметра.</p> <p>При нажатии кнопки «ПОЛУЧИТЬ РЕЗУЛЬТАТ» в блоке вывода ответов, а именно - в поле, расположенном под заголовком «Последовательность чисел Фибоначчи», выводится результат: «3»</p>
Ввести в поле ввода данных в качестве параметра n значение 50 и нажать кнопку «ПОЛУЧИТЬ РЕЗУЛЬТАТ»	Вывод сообщения об ошибке «Произошла ошибка! Время для выполнения алгоритма истекло». В окне уведомления об ошибке в правом нижнем углу есть кнопка «Закрыть»

Таблица В.4. Покрытие критериев (Т2)

Покрытие критериев	
1.1	Просмотр списка алгоритмов
1.2	Просмотр описания алгоритма
1.3	Ввод параметров
1.4	Просмотр информации о параметре
1.5	Запуск алгоритма на выполнение
1.6	Просмотр результата выполнения алгоритма
2.1	N-е число Фибоначчи
3.2.1	Целое число
3.3.1	1 параметр
4.2.1	Целое число
4.3.1	1 результат
4.4.1	Результат выполнения алгоритма
4.4.2	Сообщение об ошибке