Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«___»_____2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Учебно-исследовательский портал для поддержки
архитектурно-независимого параллельного программирования
тема

09.04.01 Информатика и вычислительная техника
код и наименование направления

09.04.01.04 Технология разработки программного обеспечения
код и наименование магистерской программы

Научный руководитель	_____	профессор, д.т.н.	А. И. Легалов
	подпись, дата	должность, учёная степень	инициалы, фамилия
Выпускник	_____		Д. Е. Костыгин
	подпись, дата		инициалы, фамилия
Рецензент	_____	доцент, д.ф.-м.н.	К. В. Сафонов
	подпись, дата	должность, учёная степень	инициалы, фамилия
Нормоконтролер	_____	профессор, д.т.н.	А. И. Легалов
	подпись, дата	должность, учёная степень	инициалы, фамилия

Красноярск 2020

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия

«___»_____2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме магистерской диссертации
бакалаврской работы, дипломного проекта, дипломной работы, магистерской диссертации

Студенту Костыгину Денису Евгеньевичу
фамилия, имя, отчество

Группа КИ18-01-4М Направление (специальность) 09.04.01
номер код

«Информатика и вычислительная техника»
наименование

Тема выпускной квалификационной работы Учебно-исследовательский портал для поддержки архитектурно-независимого параллельного программирования

Утверждена приказом по университету № 19113/с от 19.12.2018

Руководитель ВКР А. И. Легалов, профессор, д.т.н.
инициалы, фамилия, должность, учебное звание и место работы

Исходные данные для ВКР: задание на магистерскую диссертацию

Перечень разделов для ВКР: 1. Анализ предметной области; 2. Инструментальные средства разработки; 3. Описание структуры системы; 4. Описание разработанной системы.

Перечень графического материала: структурная схема системы, диаграмма прецедентов, диаграмма описания хранения данных, презентация, видео-демонстрация работы системы

Руководитель ВКР _____ А. И. Легалов
подпись инициалы и фамилия

Задание принял к исполнению _____ Д. Е. Костыгин
подпись инициалы и фамилия

«20» декабря 2018 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Учебно-исследовательский портал для поддержки архитектурно-независимого параллельного программирования» содержит 65 страниц, 30 рисунков, 2 таблицы, 20 использованных источников.

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ, ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ С ИСПОЛЬЗОВАНИЕМ ИНТЕРНЕТ ТЕХНОЛОГИЙ, УПРАВЛЯЮЩИЙ ГРАФ, ЯЗЫК ПРОГРАММИРОВАНИЯ ПИФАГОР, РЕВЕРСИВНЫЙ ИНФОРМАЦИОННЫЙ ГРАФ, ACE EDITOR.

Целью выпускной работы в форме магистерской диссертации, является создание интегрированной среды разработки с использованием Интернет технологий для функционально-потокового языка параллельного программирования «Пифагор», которая позволит производить разработку программ без необходимости в установке каких-либо средств на компьютере пользователя.

В процессе данной работы были рассмотрены существующие решения в областях встраиваемых редакторов кода, интегрированных сред разработки, реализованных при помощи Интернет технологий, проведен выбор используемых средств разработки, а также описание этих средств разработки.

В результате работы была создана система, позволяющая производить разработку программ на языке «Пифагор» без необходимости в установке каких-либо средств на компьютере пользователя.

Содержание

Введение

Интенсивное развитие вычислительной техники всегда влекло за собой как открытие новых возможностей, так появление новых проблем. Одна из таких проблем была связана с достижением пиков роста тактовой частоты процессов. В качестве выхода из сложившегося положения было произведено смещение в сторону параллельных систем, а именно распространение многоядерных процессоров. Это повлекло за собой создание средств разработки параллельных программ. И одним из перспективных направлений в этой области является развитие языков потока данных, которые изначально более приспособлены для параллельных задач. В этом случае написание параллельных программ производится с помощью функционально – потокового языка, который задает программу в виде информационного графа, обладающего рядом специфических особенностей. В его основе лежит управление по готовности данных, определяемое для процессов, протекающих внутри бесконечных ресурсов. Это позволяет описать параллелизм задачи без ресурсных конфликтов. Одним из подобных решений является функционально-потоковый язык параллельного программирования «Пифагор».

В настоящее время для языка Пифагор разработаны и продолжают разрабатываться инструментальные средства, обеспечивающие поддержку функционально-потокового параллельного программирования. Существуют консольные приложения для языка Пифагор, обеспечивающие трансляцию и выполнение программ. Разрабатывается следующая версия языка со статической типизацией данных. Формируется концепция распределенной разработки таких программ с возможностью подключения репозиториев функций, размещенных на удаленных узлах. Вместе с тем следует отметить, что для последних разработок отсутствуют средства разработки, учитывающие все возможности по быстрому формированию исходных

текстов в рамках интегрированной среды. Необходимость таких средств становится очевидной при разработке больших проектов.

Исходя из вышеперечисленных доводов, является актуальным вопрос о реализации интегрированной среды разработки для функционально-потокового языка «Пифагор».

Существует два вида интегрированных сред разработки (далее ИСР). Настольная (традиционная) ИСР и ИСР с использованием Интернет-технологий. Основным отличием ИСР с использованием Интернет технологий от традиционных ИСР является их переносимость: они не требуют от пользователя загрузки и установки каких-либо инструментальных средств. Ввиду этого довода было принято решение о реализации ИСР для функционально-потокового языка «Пифагор» с использованием Интернет технологий.

Целью магистерской диссертации является создание интегрированной среды разработки с использованием интернет-технологий для функционально-потокового языка параллельного программирования «Пифагор», которая позволит производить разработку программ без необходимости в установке каких-либо средств на компьютере пользователя.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Анализ интегрированных сред разработки с выбором архитектуры для реализации системы
2. Анализ существующих инструментальных средств для поддержки разработки с использованием функционально-потокового языка «Пифагор»
3. Анализ существующих встраиваемых редакторов кода
4. Анализ существующих интегрированных сред разработки с использованием интернет-технологий

5. Разработка архитектуры интегрированной среды, ориентированной на поддержку средств функционально-потокового параллельного программирования.
6. Проектирование и реализация среды разработки
7. Описание использования разработанных программных средств

1 Анализ предметной области

1.1 Интегрированные среды разработки

Интегрированная среда разработки представляет собой программное обеспечение, которое обеспечивает всесторонние возможности для программистов для разработки программного обеспечения. Обычно среда ИСР состоит из редактора исходного кода, средств автоматизации сборки и отладчика. Большинство современных ИСР имеют интеллектуальное дополнение кода. Некоторые ИСР, такие как NetBeans и Eclipse, содержат компилятор, интерпретатор или и то, и то другое; другие, такие как SharpDevelop и Lazarus, не содержат компилятора или интерпретатора [1].

Граница между интегрированной средой разработки и другими частями более широкой среды разработки программного обеспечения не является четко определенной. Иногда интегрируется система контроля версий или различные инструменты, упрощающие построение графического интерфейса пользователя (Graphical User Interface - GUI) [2]. Многие современные ИСР имеют обозреватель классов, обозреватель объектов, и диаграмму иерархии классов, для помощи в разработке объектно-ориентированного программного обеспечения.

Необходимым общепринятым минимумом для ИСР являются [3]:

1) Редактор кода
2) Запуск компилятора
3) Показ результата компиляции

А также несколько дополнительных функций:

1) Подсветка синтаксиса
2) Руководство пользователя
3) Файловый менеджер проекта

1.2 ИСР, реализованные с использованием Интернет-технологий

Существуют также ИСР, реализованные с использованием Интернет технологий. Такие ИСР, без исключений, создаются с использованием клиент-серверной архитектуры [4].

Неоспоримыми преимуществами ИСР, реализованных с использованием Интернет технологий, перед традиционными ИСР являются:

1) Доступ из любой точки мира (при наличии подключения к сети Интернет) при помощи любого устройства с установленным современным браузером;
2) Отсутствие необходимости загрузки и установки.
3) Все действия, связанные с запуском разрабатываемых программ, происходят на сервере, что приводит к снижению нагрузки на компьютер пользователя.

Благодаря этим фактам ИСР, реализованные с использованием Интернет технологий, пользуются большой популярностью у программистов по всему миру. Такие ИСР обычно включают в себя те же инструменты, что и традиционные, что позволяет не тратить время на обучение новым средствам разработки и приступать к работе сразу после регистрации в одной из таких систем, с условием, что на устройстве с которого будет проводиться работа, есть соединение с сетью Интернет.

Клиентской частью ИСР, реализованных с использованием Интернет технологий, является окно браузера с редактором кода, представление древа файлов текущего проекта, а также различные элементами интерфейса, отвечающие за вызов инструментов (генерацию запросов, отправку их на сервер).

Серверная часть такой ИСР, реализованных с использованием Интернет технологий, является обработчик запросов клиентской части.

1.3 Язык «Пифагор»

Одним из представителей семейства языков потока данных является язык функционально-потокового программирования «Пифагор», среди особенностей которого можно выделить следующие:

– параллелизм на уровне операций;

– архитектурная независимость, достигаемая за счёт описания в программе только информационных связей;

– асинхронный параллелизм, поддерживаемый выполнением операций по готовности данных;

– отсутствие переменных, что позволяет избежать конфликтов, связанных с совместным использованием памяти параллельными процессами;

– отсутствие операторов цикла, что позволяет избежать конфликтов при использовании различными данными одних и тех же фрагментов параллельной программы [5].

Для достижения поставленной цели в разработанной системе используются следующие инструментальные средства языка «Пифагор»

Инструментальные средства языка «Пифагор» включают в себя:

1. Транслятор;
2. Генератор управляющих графов;
3. Интерпретатор;
4. Дополнительные средства визуализации реверсивных информационных и управляющих графов.

Обзор данных средств языка «Пифагор» и пример их запуска представлен в главе 2.1 и ее подглавах.

1.4 Обзор существующих решений

1.4.1 Обзор встраиваемых редакторов кода

На сегодняшний день существует множество встраиваемых редакторов кода, используемых в различных средах. Эти редакторы обладают всеми функциями, необходимыми для комфортного написания кода, что делает разработку собственного редактора нецелесообразной. Обзор существующих разработок, их характеристики представлены далее.

1.4.1.1 Ace Editor

Ace Editor (рисунок 1) – это встраиваемый редактор кода, написанный на Javascript. Его производительность и возможности сопоставимы с нативными редакторами, такими как Vim, Sublime, TextMate. Ace является преемником проекта Mozilla Skywriter (Bespin), был разработан Ajax.org и поддерживается для проекта Cloud9 (Ace расшифровывается буквально Ajax Cloud9 Editor). Плюсом является устойчивость к большим объемам кода [6]. В этом Ace Editor выигрывает у своих конкурентов с большим отрывом.

Ace Editor используется в следующих проектах:

1. Github;
2. Tumblr (редактор тем);
3. Wikia (редактор CSS);
4. Cloud9

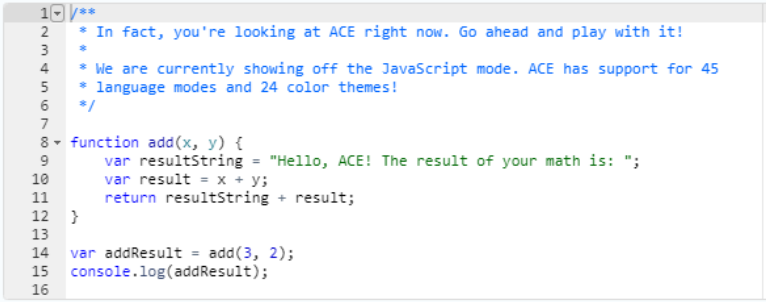

Рисунок 1 – Демонстрация Ace Editor

1.4.1.2 Monaco

Monaco – редактор кода, разработанный Microsoft для Visual Studio Online, впоследствии стал использоваться как редактор для Visual Studio Code [7], ввиду чего получает активную поддержку и доработку. Основной плюс – подробная документация, чем другие редакторы похвастаться не могут.

Визуальная составляющая редактора Monaco, встроенного в веб-приложение, представлена на рисунке 2 [8].

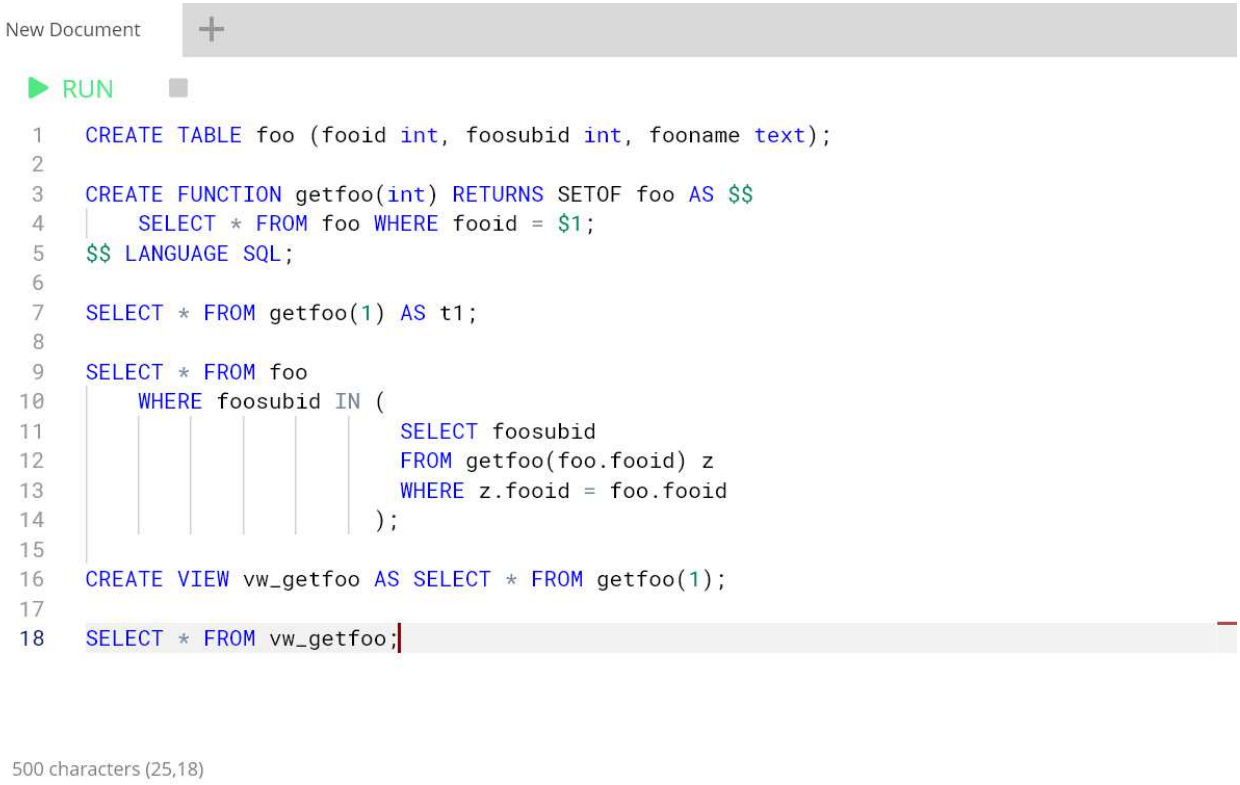

Рисунок 2 – Демонстрация Monaco

1.4.1.3 CodeMirror

CodeMirror – редактор кода, впервые разработанный в 2007 году. Первая его версия была основана на таком аттрибуте языка HTML, как contentEditable, и обладал низкой производительностью. После появления Ace, который продемонстрировал, что даже в JavaScript можно обрабатывать документы с тысячами строк без снижения производительности – автор

CodeMirror разработал версию 2. На данный момент CodeMirror позиционируется как гибкий текстовый редактор, специализированный для редактирования кода [9].

CodeMirror используется в следующих проектах:

1. CodeAnywhere;
2. Mozilla Firefox (веб консоль)
3. uBlock Origin
4. Google Chrome DevTools

Рисунок 3 – Демонстрация CodeMirror

1.4.1.4 Сравнение обозреваемых редакторов

Сравнение обозреваемых редакторов проводится для выбора одного из них как инструментальное средство разработки.

Сравнение редакторов приведено в таблице 1.

Таблица 1. Сравнение редакторов кода [6]

	Ace	CodeMirror	Monaco
Поддержка браузеров	Firefox => 3.5; Chrome, Safari => 4.0; IE => 8.0; Opera => 11.5	Firefox => 3.0; Chrome, Safari => 5.2; IE => 8.0; Opera => 9.2	Firefox => 4.0; Chrome, Safari => 5.2; IE => 11.0; Opera => 15.0

Окончание таблицы 1

Количество поддерживаемых языков (подсветка синтаксиса, автодополнение)	120	100	20
Размер последней версии	2.147 KB	1.411 KB	10.898 KB
Устойчивость к большим объемам текста	Около 200000 строк	Около 100000 строк	Около 100000 строк
Поддержка сторонних средств синтаксического контроля (подсветка, автодополнение)	Есть	Есть	Есть

1.4.2 Обзор ИСР, реализованных при помощи Интернет технологий

Существует множество ИСР разработанных реализованных при помощи интернет технологий и позволяющих вести разработку везде, где есть интернет. Самыми популярными примерами таких ИСР являются: Cloud9, CodeAnywhere, Eclipse Che, Neutron Drive, Orion. У каждой из них есть свои преимущества и недостатки, которые будут описаны далее.

1.4.2.1 Cloud9

Cloud9 – это облачная ИСР, включающая в себя редактор кода, отладчик, терминал, предустановленный набор инструментов для следующих языков программирования: JavaScript, Python, PHP, Ruby, Go, C++, и т.д. Сервис также включает в себя такие функции, как подсветка синтаксиса, режим структуры, подсказки, авто дополнение кода. Редактор кода Cloud9 основан на **Ace Editor** [10]. Разработан полностью на языке программирования Javascript, серверная часть на NodeJS.

Преимущества:

1) Удобный контроль версий (Git, SVN);
2) Встроенные инструменты для контроля качества CSS и Javascript кода.

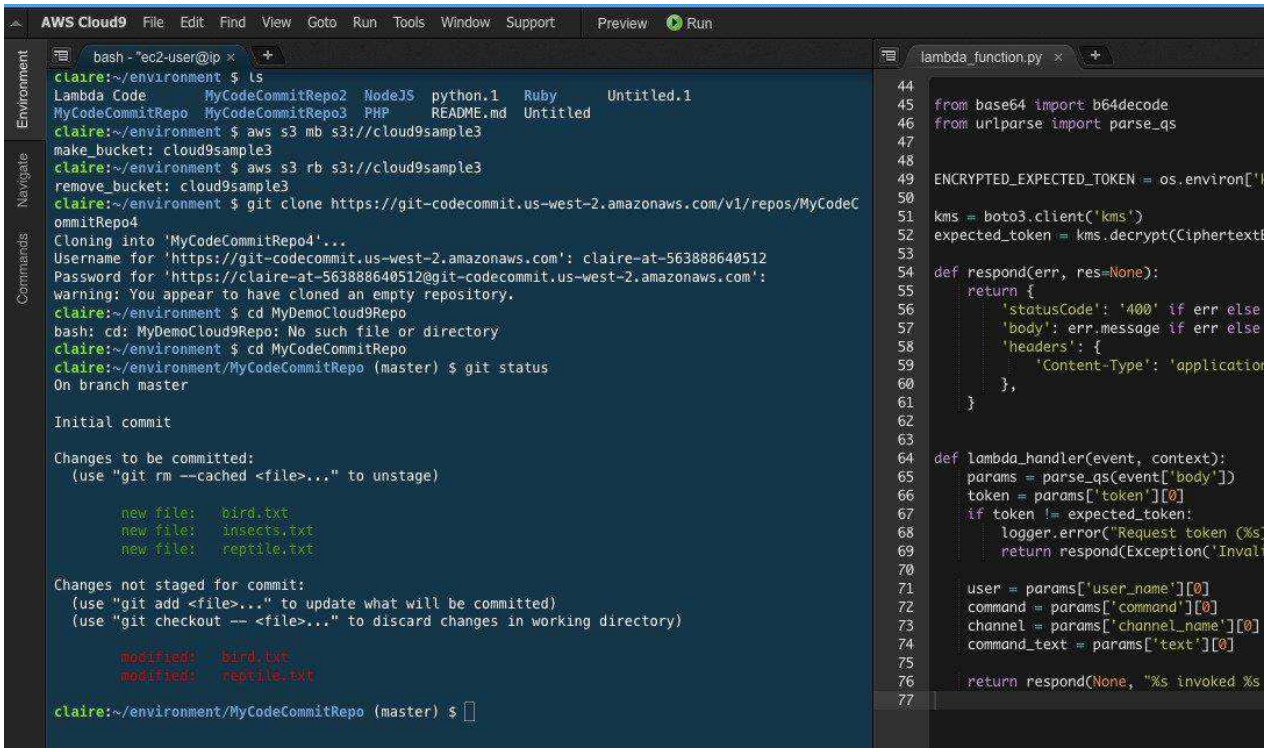

Рисунок 4 – Cloud9 ИСР

1.4.2.2 CodeAnywhere

CodeAnywhere – также как и Cloud9 является облачной ИСР с похожими на него функциями. Редактор кода основан на **CodeMirror** [11], и использует контейнеры OpenVZ для своих сред разработки. Включает в себя поддержку более 75 языков программирования. Разработан полностью на языке программирования Javascript.

Главным преимуществом CodeAnywhere является интегрированный Dropbox и SFTP-клиент, которые позволяют быстро обмениваться файлами с другими разработчиками и осуществлять резервное копирование.

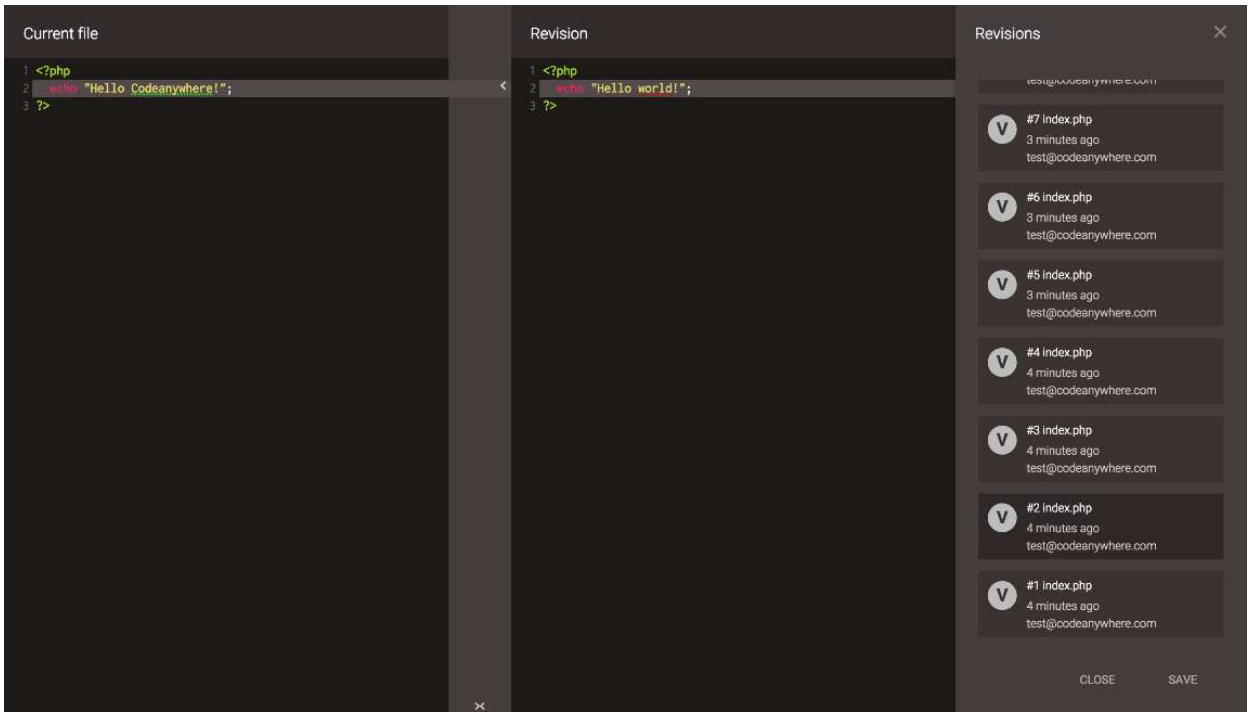

Рисунок 5 – CodeAnywhere ИСР

1.4.2.3 Eclipse Che

Eclipse Che – это сервер рабочего пространства разработчика и браузерная ИСР. Eclipse Che имеет открытый исходный код и разработан на языке программирования Java. Использует собственный редактор кода.

Плюсом Eclipse Che является обслуживание нескольких изолированных рабочих мест и управление доступом пользователей с различными правами [12].

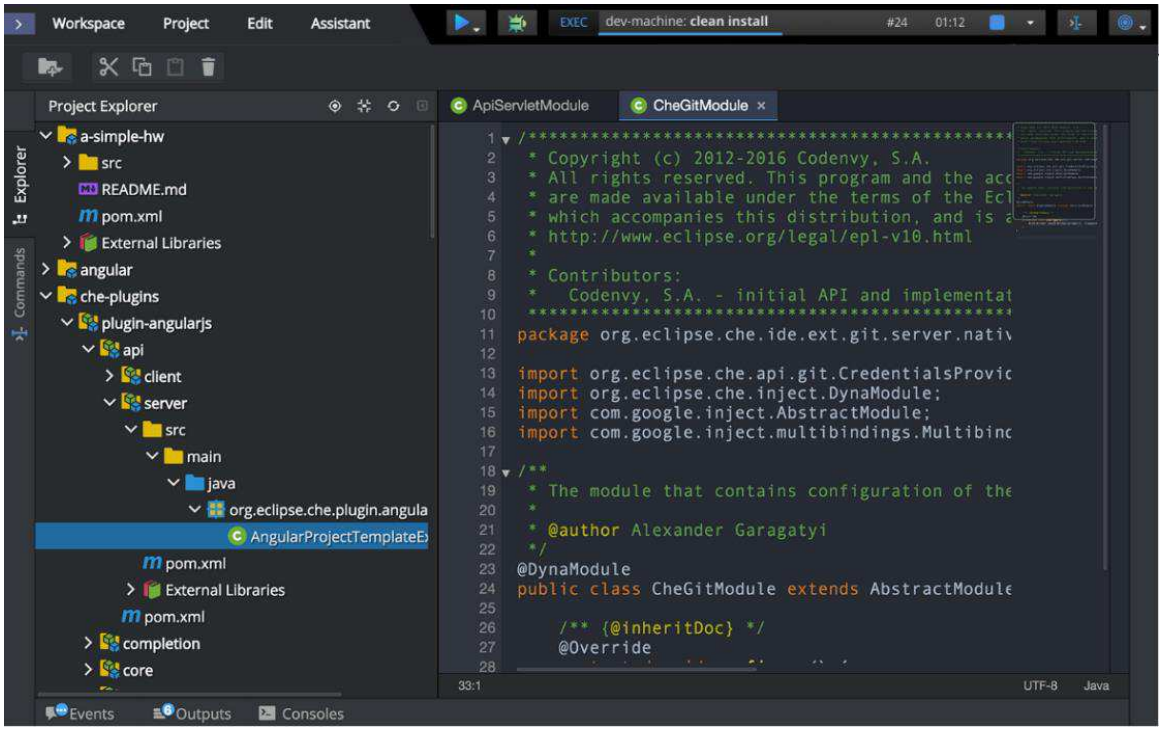

Рисунок 6 – Eclipse Che ИСР

1.4.2.4 Orion

Orion – облачная ИСР, возникшая на базе ИСР Eclipse. Редактор кода Orion базирован на Eclipse Theia. Основная сфера применения среды – разработка интерфейсов для сайтов, поэтому имеется хорошая поддержка HTML, CSS, Javascript с различными надстройками [13].

На текущий момент работа над Orion еще ведется, ввиду этого нет отличающих ее от других ИСР, представленных в данном обзоре.

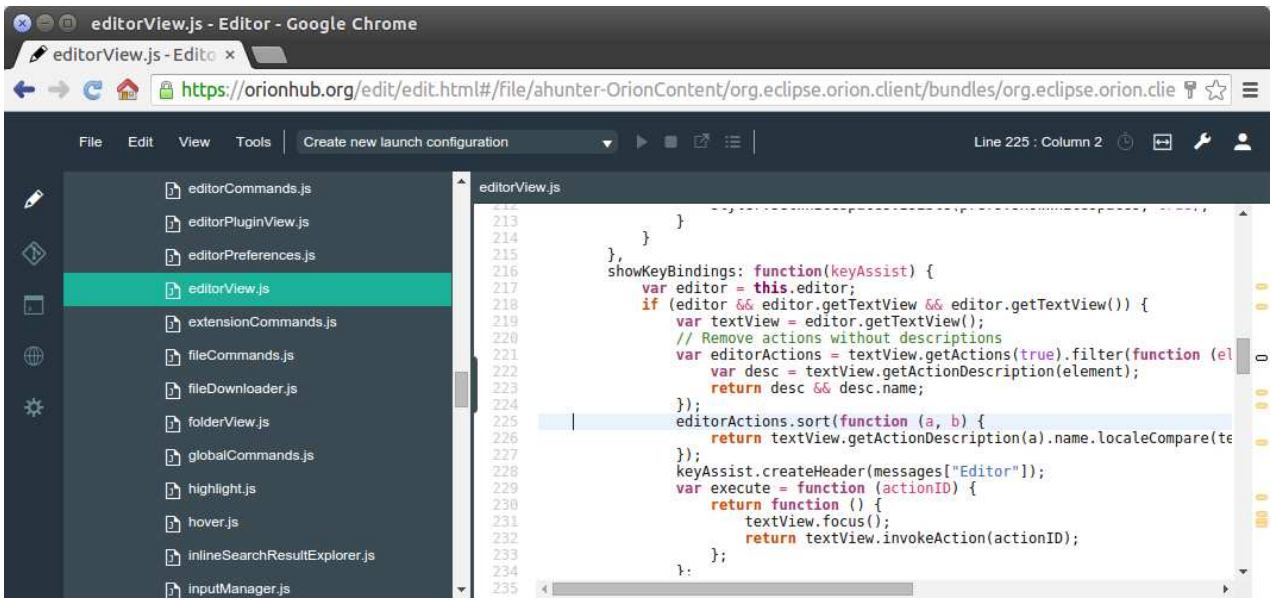

Рисунок 7 – Orion ИСР

1.4.2.5 Сравнение обозреваемых ИСР

Сравнение обозреваемых ИСР представлено в таблице 2.

Таблица 2 – сравнение обозреваемых ИСР [14]

	Cloud9	CodeAnywhere	Eclipse Che	Orion
Поддерживаемые языки	PHP, Python, Ruby, Java, Go, Javascript, HTML, CSS и др.	PHP, Python, Ruby, Perl, Java, Go, SQL, Javascript, XML, HTML, CSS и др.	PHP, Python, Ruby, Java, SQL, Javascript и др.	HTML, Javascript, CSS и др.
Подсветка синтаксиса	+	+	+	+
Автодополнение	+	+	+	+
Горячие клавиши	+	+	+	+
Работа с файлами на удаленном сервере	+	+	+	+
Макросы	-	-	+	-
Вкладки	+	+	+	+
Поддержка систем контроля версий	+	+	+	+

Окончание таблицы 2

Отладка программы	+	+	+	+
Анализаторы кода	+	+	-	+
Работа с облачными дисками/хранилищами	+	+	-	-
Встроенный терминал	+	+	+	-
Клиент Android/iOS	-	+	-	-
Навигация по проекту	+	+	+	+
Необходимость регистрации для полноценного использования системы	+	+	+	+

1.4.3 Обзор популярных PHP фреймворков

Так как работа с ИСР подразумевает использование файловой системы, а также запуск уже существующих инструментальных средств языка «Пифагор». Запуск существующих средств происходит с использованием консольных команд.

1.4.3.1 Symfony

Symfony – это бесплатный PHP фреймворк, нацеленный на создание надежных веб-приложений, с возможностью полностью контролировать его конфигурацию: от структуры каталогов до сторонних библиотек. Symfony поставляется с дополнительными инструментами, помогающими разработчикам тестировать отлаживать и документировать проекты.

Отличительные черты Symfony включают в себя:

1. Быструю скорость загрузки. Symfony 4.2 загружает свой REST API в течение двух миллисекунд, что делает его самым быстрым PHP-фреймворком [15];

2. Гибкость;

3. Подробная документация. Symfony имеет исчерпывающую и подробную документацию, на каждый компонент также существуют примеры использования;

4. Стабильная поддержка. Разработка Symfony ведется уже на протяжении 15 лет. Компания-разработчик оперативно реагирует на появление новых проблем, связанных с работой фреймворка, решая их в сроки, не нарушающие процесс разработки.

1.4.3.2 Laravel

Laravel – это бесплатный PHP фреймворк с открытым кодом, основанный на Symfony [16]. До версии 5.0 фреймворк предназначался для разработки с использованием архитектурной модели MVC (Model-View-Controller). Laravel нацелен на создание многофункциональных веб-приложений, поставляется с шаблонизатором Blade и имеет такие функции, как RESTful маршрутизацию, кэширование, аутентификация.

Отличительными чертами Laravel являются:

1. Безопасность. При веб-разработке всегда существует опасность различного рода атак, к примеру SQL-инъекций, подмены параметров консольных команд на команды другого рода. Laravel гарантирует получение безрисковой среды, защищенной от непреднамеренных и скрытых атак

2. Интеграция почтовых сервисов. Благодаря встроенной почтовой службе разработанное веб-приложение может отправлять уведомления пользователям.

3. Аутентификация. Laravel имеет расширенные возможности по аутентификации любой информации, поступающей на сервер.

4. Генерация общих функций. Laravel имеет возможность автоматической генерации такой функциональности, как регистрация и авторизация пользователя.

1.4.3.3 Yii2

Yii2 известен как фреймворк, имеющий повышенную безопасность и быстрое время загрузки. Yii2 тесно связан с такой библиотекой, как JQuery [17], что на сегодняшний день является самым большим его минусом.

Отличительные черты Yii2 включают в себя:

1. Расширенная безопасность. Yii2 обладает многочисленными мерами безопасности для предотвращения различных атак, таких как: подделка cookie, SQL инъекции, подделка межсайтовых запросов и межсайтовый скриптинг.

2. Ускоренное время разработки. Yii2 генерирует базовые CRUD (создание, чтение, обновление и удаление) операции, что позволяет сократить время разработки.

1.5 Выбор используемых средств разработки

На основе проведенного обзора необходимо произвести выбор используемых средств разработки.

Транслятор языка «Пифагор» работает с одним файлом, что приводит к необходимости обеспечения максимальной устойчивости редактора к большим объемам кода. На этом основании для разработки клиентской части используется редактор кода **Ace Editor**, так как он превосходит другие редакторы по этому показателю в два раза.

Разработка серверной части веб-приложений часто связана с использованием уже существующих средств, позволяющих сгенерировать общую функциональность, такую как регистрация и авторизация, расширенные средства шаблонизации для лучшей структуризации кода, а также проверкой данных, присылаемых пользователем, на предмет SQL-инъекций, вредоносных параметров для командной строки. На основе проведенного обзора популярных PHP-фреймворков можно заметить, что данным критериям удовлетворяет только фреймворк **Laravel**. Исходя из

этого, данный фреймворк был выбран как средство разработки серверной части системы.

1.6 Выводы по главе 1

Проведенный анализ предметной области позволил выявить необходимую функциональность разрабатываемой системы, а также выбрать используемые средства разработки, помимо средств языка «Пифагор».

Также на основе обзора существующих ИСР, реализованных и использованием Интернет технологий, выявленной необходимой функциональностью является:

1) Регистрация новых пользователей;
2) Авторизация;
3) Возможность создавать и сохранять файлы;
4) Возможность разбивать разрабатываемые программы на отдельные проекты для структуризации файлов;
5) Редактор кода;
6) Возможность использовать средства языка «Пифагор» для трансляции, генерации управляющих графов, интерпретации программ, написанных пользователем;
7) Возможность использовать средства языка «Пифагор» для визуализации сгенерированных графов;
8) Краткое руководство пользователя, объясняющее работу элементов интерфейса;
9) Импорт проектов других пользователей.

2 Инструментальные средства разработки

Ввиду необходимости реализации ИСР с использованием интернет-технологий была выбрана клиент-серверная архитектура.

Сервер отвечает за обработку входящих запросов клиента, таких как:

1) Регистрация пользователя;
2) Авторизация;
3) Генерация страниц;
4) Создание проектов и генерация их древовидной структуры;
5) Создание, сохранение и загрузку файлов;
6) Запуск средств языка «Пифагор», вывод результатов запуска.

Для разработки серверной части был использован язык программирования PHP вкупе с фреймворком Laravel.

Для разработки клиентской части были использованы языки программирования Javascript, HTML, наряду с модулем, отвечающим за редактор кода под названием Ace Editor (глава 1.4.1).

Так как PHP является кроссплатформенным языком программирования – сервер может быть запущен на любой машине под управлением следующих ОС: Linux, Mac OS, Windows.

Клиентская часть имеет требования к версии браузера и может быть использована в браузерах Firefox версии 3.5 или выше; Chrome, Safari версии 4.0 или выше; Internet Explorer версии 8.0 или выше; Opera версии 11.5 или выше. Данные ограничения накладываются текстовым редактором Ace Editor.

2.1 Особенности подключения разрабатываемой системы к существующим инструментальными средствам языка «Пифагор»

2.1.1 Компилятор «trans»

Компилятор переводит исходный текст заданной функции в Реверсивный Информационный Граф (далее – РИГ). Синтаксическая ошибка в любом месте файла с программой полностью прерывает работу компилятора. При этом результаты для уже оттранслированных констант и функций не сохраняются. [18].

Синтаксис запуска транслятора представлен далее:

trans2 <key> <in_file> [<error_log> [<debug_log>]],

где

<*key*> – ключ режима работы транслятора

-*t* для обработки файла с одной или несколькими функциями с выводом результата в репозиторий

-*i* для обработки файла, содержащего строго одну функцию с выводом результата в тот же каталог

-*c* для трансляции констант

<*in_file*> – файл с программой на языке Пифагор

<*error_log*> – файл для сообщений об ошибках. Значение по умолчанию – “error.txt”

<*debug_log*> – файл для отладочных сообщений. Значение по умолчанию – “debug.txt”

Транслятор может возвращать следующие значения:

0 – трансляция прошла успешна

1 – подано неверное количество аргументов

2 – ошибка открытия входного файла

3 – ошибка открытия файла для ошибок

4 – ошибка открытия файла отладочной информации

5 – ошибка трансляции

2.1.2 Генератор управляющих графов «cgen»

Генератор управляющих графов осуществляет анализ реверсивного информационного графа, формируя на его основе управляющий граф (далее УГ) [18].

Запуск генератора управляющих графов требует наличия сгенерированного транслятором РИГ и проводится следующим образом:

cgen2 -n <function_name>

где

<*function_name*> – полное имя обрабатываемой функции

Генератор УГ может вернуть следующие значения:

0 – генерация прошла успешно

1 – подано неверное количество аргументов

2 – ошибка открытия входного файла

2.1.3 Интерпретатор промежуточного представления «inter»

Интерпретатор осуществляет выполнение заданной функции. Используя список функций и констант, полученных в результате компоновки, он загружает для каждой функции ее РИГ и УГ в оперативную память, образуя тем самым законченную функционально-потоковую программу [19]. После этого начинается процесс выполнения (интерпретации) программы.

Для корректного запуска интерпретатора требуются сгенерированные транслятором и генератором УГ, РИГ и УГ соответственно. Запуск проводится следующей командой:

inter2 <function_name>,

где

<*function_name*> – полное имя обрабатываемой функции

В зависимости от результата транслятор возвращает следующие значения:

0 – трансляция прошла успешна

1 – подано неверное количество аргументов

2 – ошибка открытия файла с аргументом

3 – ошибка открытия файла информационного графа

4 – ошибка открытия файла управляющего графа

5 – ошибка открытия файла отладочной информации

6 – ошибка подгрузки дерева модулей

7 – ошибка открытия файла для результата

2.1.4 Генератор графического представления РИГ «rig2dot»

Данное средство производит генерацию .dot файла на основе РИГ, сгенерированного транслятором. Этот файл можно использовать для визуализации РИГ в виде изображения в формате «.png», если средство было запущено с необязательным ключом -png.

Запуск данного средства производится следующей командой:

rig2dot [-png] <in_file>

где

<*in_file*> – файл с РИГ функции

В зависимости от результата rig2dot возвращает следующие значения:

0 – порождение dot-файла прошло успешно, либо подано некорректное количество аргументов

-1 – ошибка открытия входного файла

2.1.5 Генератор графического представления УГ «cg2dot»

Данное средство имеет практически идентичную функциональность с средством rig2dot, за исключением того, что для генерации .dot файла ему требуется УГ.

Запуск данного средства и возвращаемые значения также идентичны rig2dot.

2.1.6 Пример работы с инструментальными средствами языка «Пифагор»

Пусть дана программа, производящая сортировку списка методом Хоара

```
sort.hoar.mind << funcdef Y
{
  X << Y:2;
  cind << Y:1;
  fl << ( (X:|,cind):[=,>] ):?;
  act << ( cind,
           {
            block
            {
              ncind << ((cind,1):+,X):sort.hoar.mind;
              fl2 << ( (X:cind, X:ncind ):[>,<=]):?;
              act2 << (ncind,cind);
              break << act2:fl2:.;
            }
           }
         );
  return << act:fl:.;
}

sort.hoar.getmind << funcdef X
{
  minind << sort.hoar.mind^(1,X);
  return << minind;
}

sort.hoar.getsort << funcdef X
{

  fl << ( (X:|,1):[=,>] ):?;
  act << (X,
           {
            block
            {
              gm << X:sort.hoar.getmind;
              mgm << (0,gm):-;
              minel << X:gm;
              tail << X:mgm;
              ktail << tail:sort.hoar.getsort;
              break << (minel,ktail:[]);
              }
            }
          );
  return << act:fl:.;
}
```

Здесь функция *getsort* осуществляет собственно сортировку, используя в процессе две вспомогательные функции – *getmind* и *mind*. Сохранена программа в файле *sort.pfg* в директории с *.exe*-файлами. Все функции

принадлежат к пространству имен *hoar*, в свою очередь входящем в пространство имен *sort*.

1. Транслируем программу

   trans2 -t sort.pfg err.log dbg.log

Если все прошло без ошибок, в текущей директории появятся пустой файл *err.log* и файл *dbg.log*, содержащий отладочную информацию; в папке *repository* должны появиться папки *sort/hoar/mind*,*sort/hoar/getmind*, *sort/hoar/getsort*. В этих папках должны содержаться файлы *1.rig* с текстовым представлением соответствующих РИГ.

2. Создаем управляющие графы

   cgen2 -n sort.hoar.getsort
   cgen2 -n sort.hoar.getmind
   cgen2 -n sort.hoar.mind

Если все прошло без ошибок, в папках *sort/hoar/mind*,*sort/hoar/getmind*, *sort/hoar/getsort* должны содержаться файлы *1.cg* с текстовым представлением соответствующих УГ.

3. Подготавливаем аргумент.

Сохраним аргумент для нашей функции – список *(1,2,3,4,3,2,1,2,3)* в файл *arg.pfg*.

trans2 -c arg.pfg erra.log dbga.log

Если все прошло без ошибок, в текущей директории появятся пустой файл *erra.log*, файл *dbga.log*, содержащий отладочную информацию и файл *arg.rig*

4. Запускаем интерпретацию.

inter sort.hoar.getsort

Если все прошло без ошибок, на экран будет выведена отладочная информация и результат выполнения программы, список *(1,1,2,2,2,3,3,4)*.

2.2 Ace Editor

Ace Editor обладает рядом функций, позволяющих организовать среду разработки. Такими функциями являются: встроенный UndoManager, возможность получить строку по ее номеру, а также возможность заменять документ (содержащий код) без потери позиции курсора и прокрутки. Обобщенная схема работы Ace Editor представлена на рисунке 8.

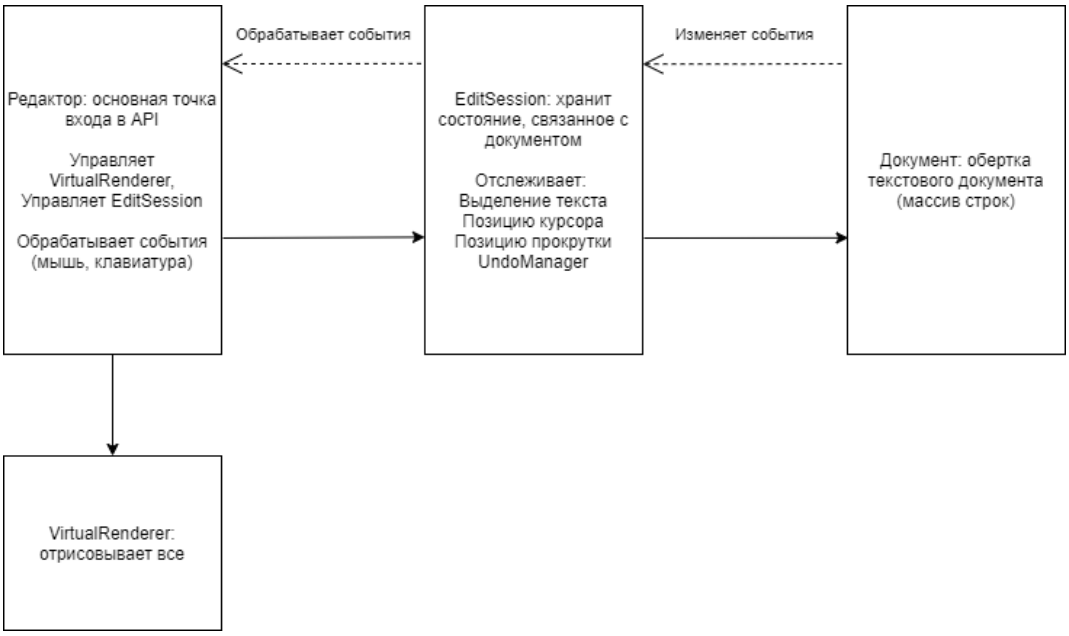

Рисунок 8 – Структурная схема Ace Editor

2.3 Выводы по главе 2

В ходе описания инструментальных средств языка «Пифагор» были выявлены особенности их использования:

1. Инструментальные средства языка «Пифагор» работают только с файлами;

2. Их запуск происходит при помощи командной строки эксклюзивно;

3. Некоторые части запуска инструментальных средств в рамках интегрированной среды можно автоматизировать: trans, cgen, так как не требуют выбора функции для интерпретации, а также cg2png, rig2png, так как требуют на вход только файлы .cg и .rig соответственно.

Также был проведен краткий анализ средств редактора кода Ace Editor, которые позволят организовать среду разработки.

3 Описание структуры системы

В данной главе представлено описание разработанной системы: ее классов, методов классов, а также функций.

3.1 Прецеденты рабочей системы

Среда разработки должна предоставлять стандартный для систем такого рода набор операций и функциональность: список проектов, создание проектов, файловый проводник, создание, сохранение и редактирование файлов. Диаграмма прецендентов системы представлена на рисунке 9.

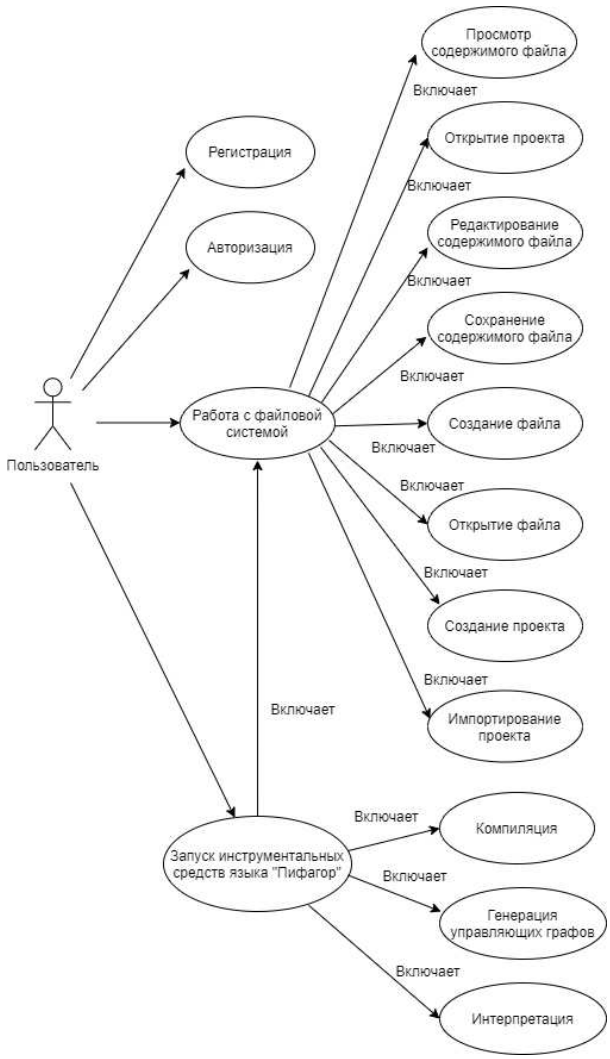

Рисунок 10 – Диаграмма прецендентов

На основе диаграммы прецендентов также была разработана структурная схема системы, представленная на рисунке 10.

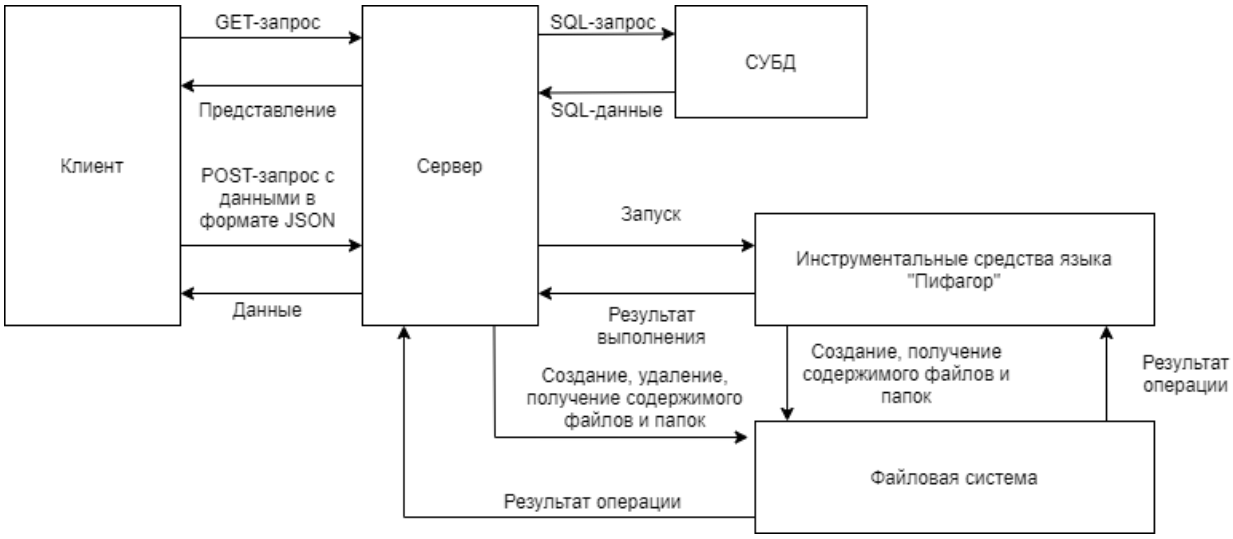

Рисунок 10 – Структурная схема системы

3.2 Хранение данных

Для хранения данных используется гибридный подход: информация о пользователях и их проектах хранится в базе данных MySQL, когда как файлы проектов хранятся в отдельных директориях, название которых совпадает с названием проекта. Такой подход был выбран ввиду специфики инструментальных средств языка «Пифагор». Все инструментальные средства языка «Пифагор» работают с файлами. Если хранить содержимое файлов в базе данных, для запуска разработанных пользователем программ понадобится делать запрос к базе данных, создавать новый файл, и только затем запускать инструментальные средства. Использование гибридного подхода хранения данных позволяет обойти дополнительные действия, предшествующие запуску инструментальных средств. Диаграмма, описывающая хранение данных представлена на рисунке 11.

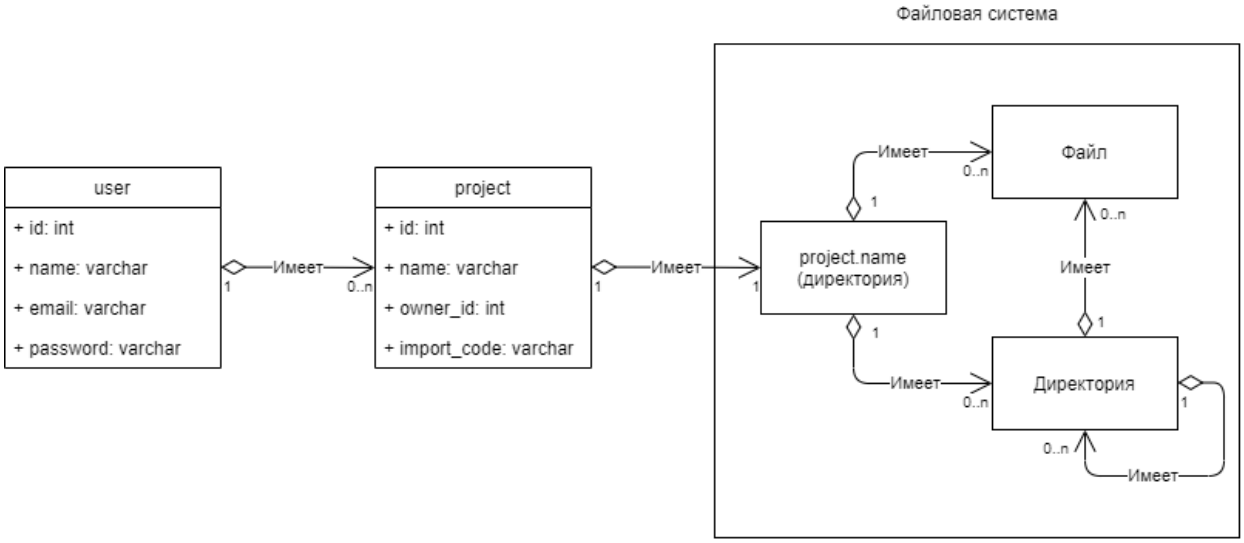

Рисунок 11 – Диаграмма описания хранения данных

3.3 Клиентская часть

3.3.1 Структура Tab

Структура «Tab» представляет собой набор переменных, описывающих вкладку редактора. Список переменных, сгруппированных данной структурой представлен далее:

1. Contents – хранит объект Ace Editor – EditSession;
2. Filename – имя файла, за который отвечает данная вкладка;
3. Element – указатель на HTML-представление вкладки;
4. Project – проект пользователя, в котором хранится Filename;
5. UnsavedChanges – булева переменная, обозначающая несохраненные изменения в данной вкладке;
6. Ext – расширение файла.

3.3.2 Класс TabManager

Класс TabManager сочетает в себе функции управления интерфейсом, а также массив структур «Tab». TabManager обладает следующими функциями:

1. Функция addTab;

2. Функция deleteTab;

3. Функция selectTab;

4. Функция configureButtons;

3.3.2.1 Метод addTab

Метод addTab отвечает за создание новой вкладки редактора и принимает на вход переменные, идентичные переменным структуры «Tab», за исключением значений переменных UnsavedChanges и Element. При ее вызове создается новый экземпляр структуры «Tab» и переданные функции значения переменной записываются в соответствующие поля. Данный экземпляр структуры «Tab» добавляется в конец массива структур класса TabManager. Далее генерируется HTML-представление вкладки редактора, представленное на рисунке 12.

Рисунок 12 - Активная вкладка редактора

HTML-представление вкладки состоит из элемента с именем файла (рисунок 13) и элемента, отвечающего за удаление вкладки (рисунок 14).

Рисунок 13 - Элемент с именем файла

Рисунок 14 - Элемент, отвечающий за удаление вкладки

После генерации HTML-представления происходит его размещение на странице.

3.3.2.2 Метод deleteTab

Данный метод вызывается по клику на элемент HTML-представления вкладки, отвечающего за ее удаление. При ее вызове происходит удаление HTML-представления со страницы, а также удаление соответствующего экземпляра структуры «Tab» из массива структур класса tabManager.

3.3.2.3 Метод selectTab

Метод selectTab вызывается по клику на элемент с именем файла HTML-представления вкладки. При вызове данной функции происходит замена объекта EditSession (глава 2.2) текущей вкладки на объект выбираемой вкладки.

3.3.2.4 Метод configureButtons

Метод configureButtons отвечает за отображение кнопок редактора кода в зависимости от расширения файла, который соответствует вкладке. Если расширение файла соответствует расширениям языка «Пифагор», кнопки будут отображены. Иначе функция скроет кнопки.

3.3.3 Функция getTabFromArray

Данная функция возвращает экземпляр структуры «Tab», соответствующий активной вкладке.

3.3.4 Функция getFuncNames

Функция getFuncNames предназначена для анализа кода, написанного пользователем. Возвращает имена функций, описанные пользователем.

3.3.5 Функция runTool

Функция runTool принимает на вход один параметр: url. При вызове функции runTool происходит проверка на наличие несохраненных изменений в текущей вкладке редактора. Если таковые присутствуют, функция выведет

диалоговое окно с просьбой сохранить изменения и прекратится ее выполнение. Иначе функция подготавливает данные, необходимые для запроса. Для запуска транслятора и генератора управляющих графов, генераторов .dot файлов – это имя файла и проект, ассоциируемый с файлом. Для запуска интерпретатора и всех инструментальных средств по порядку – это имя файла, проект, ассоциируемый с файлом, функция, которую необходимо интерпретировать, а также, если необходимо, аргумент для функции. После подготовки данных происходит запрос к серверу по url, который передается функции в виде параметра. Результатом вызова функции является вывод информации, полученной путем запуска инструментальных средств «Пифагор», в поле для вывода результатов запуска (транслятор, генератор УГ, интерпретатор), либо в поле для вывода изображений (cg2dot, rig2dot). Если данные, полученные сервером, не пройдут валидацию, выведется ошибка.

3.3.6 Функция displayImage

Функция displayImage принимает на вход один параметр – изображение в кодировке base64. Вызывается функцией runTool, если результатом выполнения является изображение. Данная функция отвечает за вывод полученного изображения на страницу, путем преобразования его из кодировки base64 в формат .png, с последующим добавлением его в поле для вывода изображений.

3.3.7 Функция displayResult

Данная функция вызывается функцией runTool, если результатом выполнения является результат запуска транслятора, генератора УГ, интерпретатора языка «Пифагор». Результатом выполнения функции displayResult является вывод полученной информации в поле, предназначенное для ее вывода.

3.3.8 Функция getPngFile

При вызове функции getPngFile происходит запрос к серверу по маршруту editor/getPngFile. Данными, посылаемыми на сервер, являются имя файла и проект, ассоциируемый с этим файлом. Успешным результатом работы функции является вывод полученного изображения при помощи функции displayImage. В случае, если такого файла не существует, выводится ошибка.

3.3.9 Функция getFileContents

При вызове функцции getFileContents происходит запрос к серверу по маршруту editor/getFileContents. Данными, посылаемыми на сервер, являются имя файла и проект, ассоциируемый с этим файлом. Успешным результатом работы функции является создание новой вкладки редактора при помощи функции addTab класса TabManager с содержимым файла, по которому кликал пользователь. В случае, если такого файла не существует, выводится ошибка.

3.3.10 Функция fileClicked

Данная функция отвечает за проверку расширения файла, по которому кликнул пользователь. Если расширение файла – «.png» данная функция вызывает функцию getPngFile. Если расширение файла отличается от «.png» вызывается функция getFileContents.

3.3.11 Функция saveFileContents

Данная функция отвечает за подготовку запроса к серверу с целью сохранения содержимого файла. Данными, посылаемыми на сервер, являются имя файла, имя проекта, ассоциируемого с файлом, а также содержимое файла. При успешном сохранении содержимого файла выводится сообщение об успешности операции. В случае, если файл не существует или произошла ошибка записи, выводится ошибка.

3.3.12 Функция refreshProjectContents

При вызове функции refreshProjectContents происходит подготовка запроса к серверу с целью получения древовидного представления файлов проекта. Данная функция вызывается каждый раз при каком-либо изменении (создании нового файла, папки) проекта. Результатом ее работы является обновленное древовидное представление файлов проекта.

3.3.13 Функция submitFile

Целью данной функции является создание файла с именем, введенным пользователем, в текущем открытом проекте. Если имя файла не прошло валидацию – выводится ошибка с просьбой ввести имя еще раз. Иначе посылается запрос к серверу, при успешной обработке которого выводится сообщение об успешности операции, созданный файл появляется в древовидном представлении файлов проекта. При ошибке обработки запроса (невозможно создать файл, файл с таким именем уже существует) выводится ошибка.

3.3.14 Функция submitProject

Функция submitProject предназначена для отправки запроса к серверу с целью создания нового проекта с именем, введенным пользователем. Если такой проект уже существует выводится ошибка. Иначе созданный проект появляется в списке проектов.

3.3.15 Функция displayFuncNames

Данная функция отвечает за вывод имен функций, описанных пользователем в текущей вкладке редактора в меню выбора функции запуска. Вызывается функцией runTool, если необходимо запустить интерпретатор языка «Пифагор» или все инструментальные средства поочередно.

3.3.16 Функция exportProjectList

Данная функция выводит лист проектов и их кодов импорта в соответствующее диалоговое окно.

3.4 Серверная часть

При разработке серверной части системы использовался фреймворк Laravel. В контексте разработанной системы фреймворк отвечает за маршрутизацию, подготовку запросов к базе данных (борьбу с SQL-инъекциями), шаблонизацию и работу с миграциями. В ходе проектирования серверной части системы были разработаны структурная схема работы серверной части системы (рисунок 15) и диаграмма классов системы (рисунок 16).

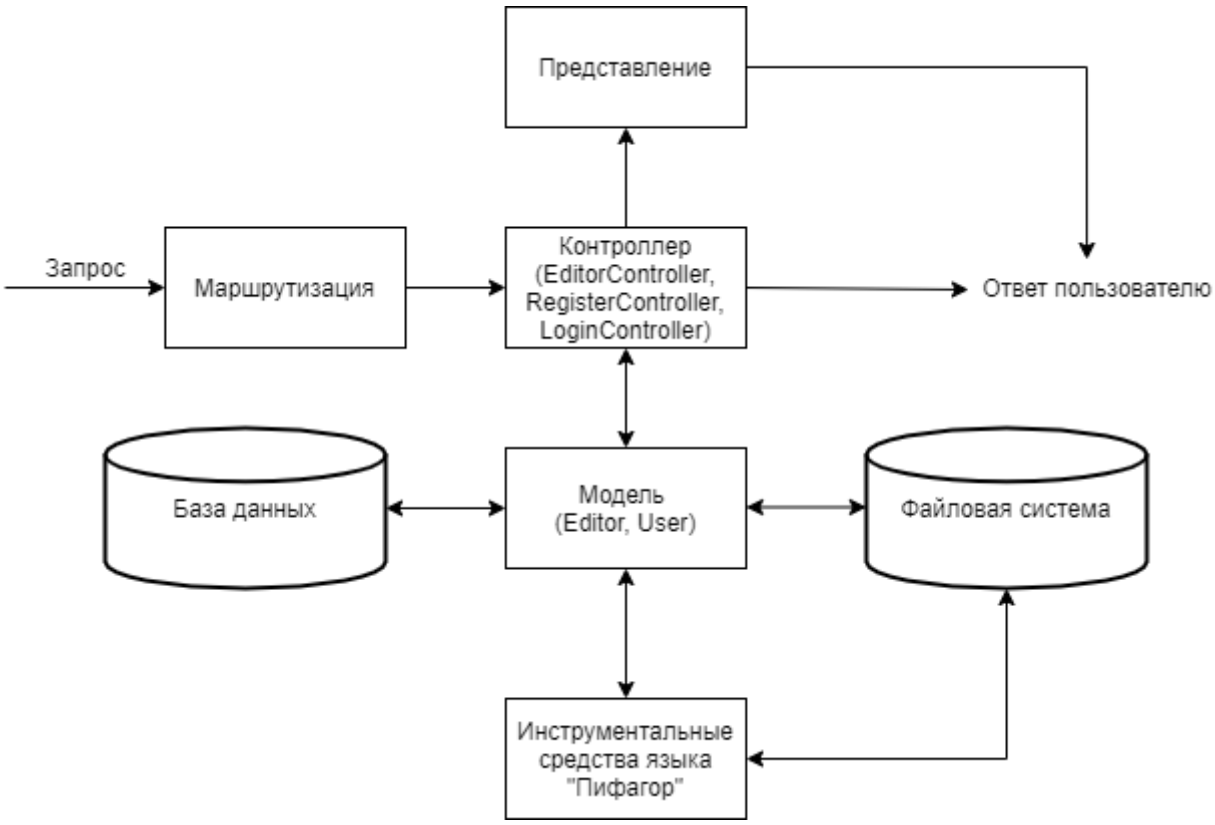

Рисунок 15 – Структурная схема работы серверной части системы

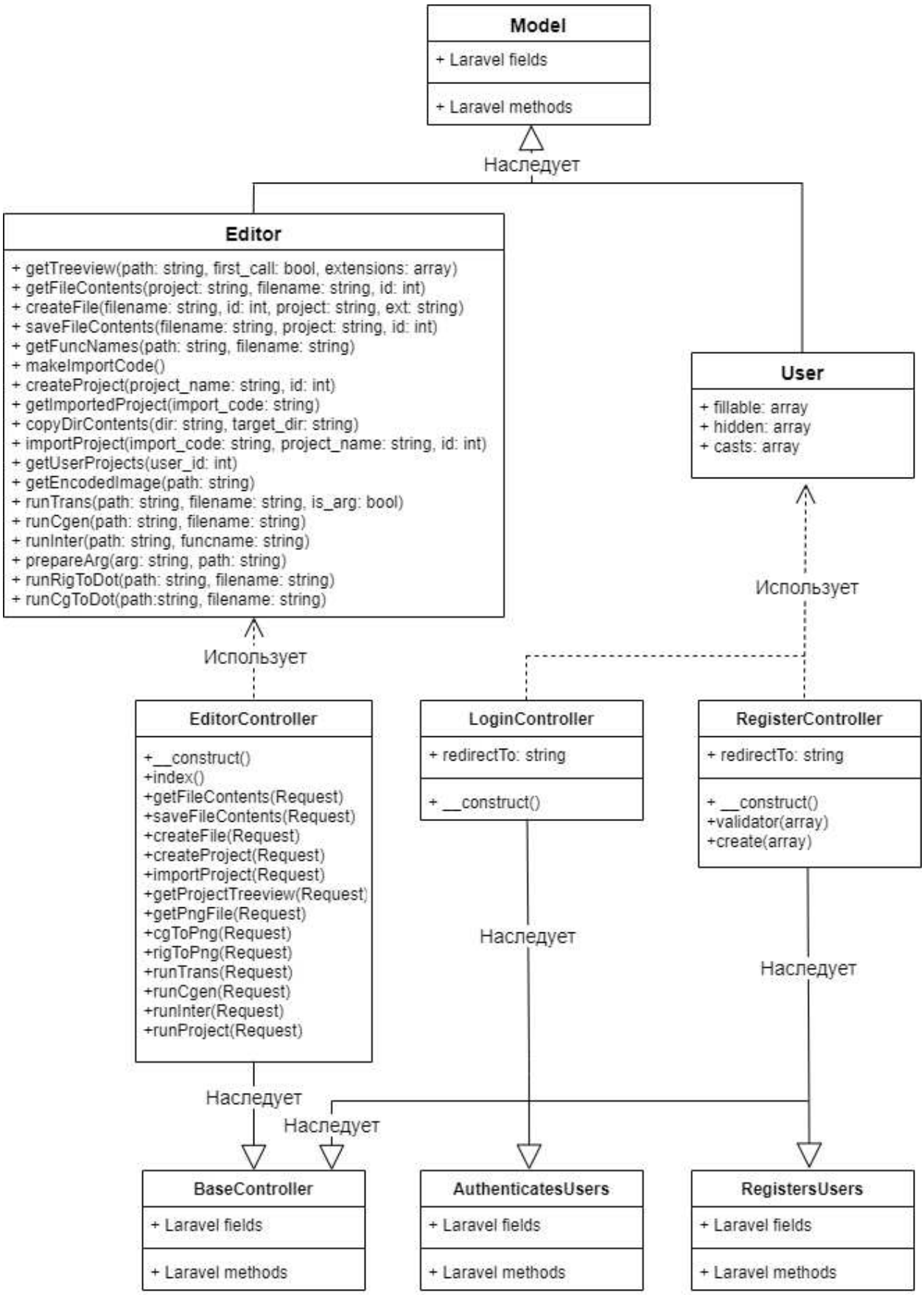

Рисунок 16 – Диаграмма классов серверной части системы

3.4.1 Класс User

Данный класс является моделью пользователя и не имеет собственных методов. Используется классами LoginController и RegisterController для реализации паттерна ActiveRecord.

3.4.2 Класс LoginController

Данный класс предназначен для использования встроенных средств фреймворка Laravel описывающих авторизацию пользователя. Имеет единственный метод _construct, а также множество методов фреймворка, полученных при помощи механизма языка PHP под названием «trait» [20].

3.4.3 Класс RegisterController

Также, как и LoginController, данный класс предназначен для использования встроенных средств фреймворка Laravel описывающих регистрацию пользователя. Имеет три собственных метода - _construct, validator и create, а также множество методов фреймворка.

3.4.3.1 Метод validator

Данный метод используется при валидации информации, полученной от пользователя. Создает объект Validator с необходимыми правилами валидации.

3.4.3.2 Метод create

Данный метод использует модель User для создания записи в базе данных при успешной регистрации пользователя, а также создания соответствующей созданному пользователю директории.

3.4.4 Класс Editor

Данный класс отвечает за работу с данными, полученными от контроллера EditorController. Описание его методов представлено далее.

3.4.4.1 Метод getTreeview

Метод getTreeView это рекурсивный метод прохода директорий. Данный метод принимает на вход три параметра: path (путь к директории), булеву переменную first_call (изначальное значение true) и extensions (массив переменных, описывающих расширения файлов, игнорирующиеся методом). Возвращаемым значением метода является массив строк, соответствующий именам файлов и их расширениям, а также поддиректории параметра path.

3.4.4.2 Метод getFileContents

Данный метод предназначен для получения содержимого файла. Принимает на вход три параметра: project (строка, имя проекта, ассоциируемого с файлом), filename (путь к файлу в проекте), id (идентификатор пользователя, который создал проект). Возвращает 0, если файл не существует или строку с содержимым файла при успешном чтении файла.

3.4.4.3 Метод createFile

Цель данного метода – создание файла. Принимает на вход четыре параметра: filename (строка, имя файла), id (целочисленное значение, идентификатор пользователя), project (строка, имя проекта, ассоциируемого с создаваемым файлом), ext (строка, расширение создаваемого файла). Возвращает одно из следующих значений:

1) 0 – директория, в которой необходимо создать файл не существует;
2) 1 – файл с таким именем уже существует;
3) 2 – невозможно создать файл (ошибка прав записи);
4) Массив с двумя значениями: путь к файлу в проекте и его расширение. Используется для создания представления файла для древовидного представления файлов проекта.

3.4.4.4 Метод saveFileContents

Метод saveFileContents сохраняет содержимое contents в файл filename проекта project пользователя id. Возвращает одно из следующих значений:

1) 0 – файла не существует;
2) 1 – невозможно записать в файл (ошибка прав записи);
3) 2 – операция успешно выполнена.

3.4.4.5 Метод getFuncNames

Данный метод предназначен для построчного чтения файла с целью получения имен функций для генератора УГ языка «Пифагор». Возвращает 0, если файла, который необходимо прочитать, не существует или, при успешном чтении, массив строк, где каждая строка – имя функции.

3.4.4.6 Метод makeImportCode

Данный метод генерирует случайную строку символов (минимум 15 символов, строчные и заглавные буквы и цифры) и возвращает ее.

3.4.4.7 Метод createProject

Метод createProject предназначен для создания проекта. Метод принимает два параметра: project_name (строка, имя создаваемого проекта) и id (идентификатор пользователя, создающего проект). Ввиду особенности хранения данных, описанной в главе 2.2, задачей метода является создание директории с именем project_name в директории, ассоциируемой с пользователем (id). Данный метод возвращает следующие значения:

1) 0 – проект с таким именем уже существует;
2) 1 – произошла ошибка при создании директории;
3) 2 – произошла ошибка при создании записи в базе данных;
4) Массив строк, состоящий из строки с кодом импорта проекта (генерируется функцией makeImportCode) и строки с именем

проекта. Используется для создания визуального представления проекта.

3.4.4.8 Метод getImportedProject

Метод getImportedProject предназначен для получения информации о импортируемом проекте из базы данных. Принимает один параметр: import_code (строка, код импорта проекта). Возвращает массив с информацией о проекте с соответствующим import_code кодом импорта, либо NULL, если записи не было найдено.

3.4.4.9 Метод copyDirContents

Данный метод производит рекурсивное копирование всего содержимого директории dir в директорию target_dir.

3.4.4.10 Метод importProject

Данный метод предназначен для импортирования проекта. Принимает три параметра: import_code (строка, код импорта), project_name (строка, имя проекта после импортирования) и id (целочисленное значение, идентификатор пользователя, импортирующего проект). Вызывает методы getImportedProject, createProject и copyDirContents и возвращает следующие значения:

1) Все возвращаемые значения метода createProject;
2) 3 – не существует проекта с таким кодом импорта.

3.4.4.11 Метод getUserProjects

Данный метод производит запрос к базе данных, с целью получения всех проектов с соответствующим user_id идентификатором пользователя, создавшим проекты. Возвращает массив с информацией о проектах пользователя.

3.4.4.12 Метод getEncodedImage

Данный метод возвращает изображение с расширением «.png», закодированное в base64.

3.4.4.13 Метод runTrans

Данный метод отвечает за запуск инструментального средства языка «Пифагор» – «trans». Принимает следующие параметры: path (строка, путь к файлу), filename (строка, имя файла), is_arg (булева переменная, является ли файл файлом аргумента для функции, изначальное значение – false). Запуск инструментального средства trans происходит при помощи функции языка PHP – exec. Для корректной работы инструментального средства (нюансы описаны в главе 2.1.6) происходит переход в директорию path, и только затем его запуск (порождаемые файлы, директории описаны в главе 2.1.6). Возвращает 0, если файла не существует, либо массив с целочисленным значением output_var, а также строкой output, где output_var – значение, возвращаемое инструментальным средством trans, output – отладочная информация, выводимая инструментальным средством trans.

3.4.4.14 Метод runCgen

Данный метод отвечает за запуск инструментального средства языка «Пифагор» - «cgen». Принимает следующие параметры: path (строка, путь к файлу), filename (строка, имя файла). Вызывает метод getFuncNames, так как для работы «cgen» необходимы имена функций (глава 2.1.2). Полученный массив строк с именами функций используется в цикле foreach для запуска «cgen» при помощи функции exec. Результатом работы метода является двумерный массив, где первый подмассив это все значения, возвращаемые каждым запуском «cgen», а второй – отладочная информация, выводимая «cgen».

3.4.4.15 Метод runInter

Данный метод отвечает за запуск инструментального средства языка «Пифагор» – «inter». Принимает следующие параметры: funcname (строка, имя интерпретируемой функции), path (путь к директории проекта). Запуск инструментального средства происходит также при помощи функции exec. Возвращаемым значением является также массив содержащий output_var и output.

3.4.4.16 Метод prepareArg

Данный метод предназначен для подготовки аргумента функции к использованию инструментальным средством «inter». Принимает следующие параметры: arg (строка, аргумент функции), path (путь к директории проекта). Вызывает методы createFile, saveFileContents, runTrans. В случае неудачной работы одного из методов возвращает массив с двумя значениями: номер метода, вызвавшего ошибку и значение, возвращаемое этим методом. В случае успешной работы всех методов возвращаемое значение совпадает с возвращаемым значением метода runTrans.

3.4.4.17 Метод runRigToDot

Данный метод отвечает за запуск инструментального средства языка «Пифагор» – «rig2dot». Принимает следующие параметры: path (строка, путь к файлу) и filename (строка, имя файла). Запуск инструментального средства происходит при помощи функции exec. При успешном запуске «rig2dot» вызывает метод getEncodedImage и возвращает полученное значение. При ошибке работы инструментального средства возвращает -1 – ошибка открытия входного файла (глава 2.1.4).

3.4.4.18 Метод runCgToDot

Данный метод идентичен методу runRigToDot, за исключением запускаемого инструментального средства, которым в этом случае является «cg2dot».

3.4.5 Класс EditorController

Класс EditorController отвечает за валидацию данных, полученных из запроса, вызовы методов модели Editor, формирование ответа клиенту.

Данный класс обладает рядом методов, каждый из которых соответствует отдельному маршруту.

3.4.5.1 Метод index

Метод index вызывается при получении GET-запроса по маршруту /editor и отвечает за генерацию представления основной страницы маршрута /editor и отправку его пользователю. Вызывает метод модели getUserProjects.

3.4.5.2 Метод createProject

Данный метод вызывается при получении POST-запроса по маршруту /editor/createProject. Запрос должен содержать имя создаваемого проекта. Имя создаваемого проекта экранируется при помощи стандартной функции PHP escapeshellarg и впоследствии передается на обработку методу модели Editor - createProject. В зависимости от значения, возвращенного методом модели, формируется сообщение об ошибке выполнения операции или о ее успешности. После формирования сообщения оно посылается клиенту.

3.4.5.3 Метод saveFileContents

Метод saveFileContents вызывается при получении POST-запроса по маршруту /editor/saveContents. Запрос должен содержать содержимое файла, имя файла и имя проекта, ассоциируемое с файлом. При успешном прохождении валидации данными вызывается метод модели Editor

saveFileContents. В зависимости от значения, возвращенного методом модели, формируется ответ на запрос. При ошибке сохранения содержимого файла (запись в файл невозможна или файла не существует) формируется сообщение об ошибке выполнение операции. При успешном сохранении содержимого файла формируется сообщение о успешности операции. После формирования сообщения о результате операции оно посылается клиенту.

3.4.5.4 Метод getFileContents

Метод getFileContents вызывается при получении POST-запроса по маршруту /editor/getFileContents. Запрос должен содержать имя файла и имя проекта, ассоциируемое с файлом. При успешном прохождении валидации данными вызывается метод модели getFileContents. В зависимости от значения, возвращенного методом модели, формируется ответ на запрос. Если файла не существует или произошла ошибка чтения формируется сообщение об ошибке выполнения операции. В случае успешности операции метод посылает ответ клиенту, содержащий содержимое файла.

3.4.5.5 Метод createFile

Метод createFile вызывается при получении POST-запроса по маршруту /editor/createFile. Запрос должен содержать имя создаваемого файла и имя проекта, в котором создается файл. При успешной валидации данных вызывается метод модели createFile. В зависимости от значения, возвращенного моделью, формируется ответ на запрос. В случае возникновения ошибки при создании файла (директория, в которой необходимо создать файл не существует, невозможно создать файл, файл с таким именем уже существует) формируется сообщение об ошибке операции. В случае успешного создания файла метод создает представление созданного файла для древа файлов проекта и посылает его клиенту.

3.4.5.6 Метод importProject

Метод importProject вызывается при получении POST-запроса по маршруту /editor/importProject. Запрос должен содержать код импорта, имя проекта после импорта. При успешной валидации данных вызывается метод модели importProject. В зависимости от значения, возвращенного методом модели, формируется ответ на запрос. Если произошла ошибка при импорте проекта посылается ответ, содержащий описание ошибки. Если импорт прошел успешно посылается представление проекта для списка проектов.

3.4.5.7 Метод getProjectTreeview

Метод getProjectTreeview вызывается при получении POST-запроса по маршруту /editor/getProjectTreeview. Запрос должен содержать имя проекта. При успешной валидации имени проекта вызывается метод модели getTreeview. Полученный массив используется для генерации представления древа файлов проекта (рисунок 17), которое вследствие отправляется клиенту.

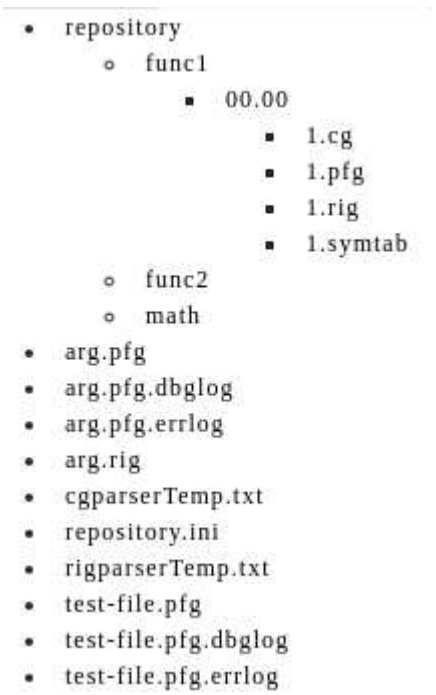

Рисунок 17 - демонстрация представления древа файлов проекта

3.4.5.8 Метод getPngFile

Метод getPngFile вызывается при получении POST-запроса по маршруту /editor/getPngFile. Запрос должен содержать два значения: путь до файла в проекте и имя проекта. При успешной валидации данных вызывается метод модели getEncodedImage, результат работы которой отправляется клиенту.

3.4.5.9 Метод cgToPng

Метод cgToPng вызывается при получении POST-запроса по маршруту /editor/cgToPng. Запрос должен содержать два значения: путь до файла с расширением .cg, а также имя проекта. При успешной валидации полученных из запроса данных вызывается метод модели cgToPng. В зависимости от значения, возвращенного методом модели, отправляется один из следующих вариантов ответа клиенту:

1) Произошла ошибка при запуске инструментального средства «cg2png»;
2) Закодированное в base64 изображение.

3.4.5.10 Метод rigToPng

Метод rigToPng вызывается при получении POST-запроса по маршруту /editor/rigToPng. Алгоритм его работы фактически идентичен методу cgToPng, за исключением вызываемого метода модели, которым в данном случае является rigToPng.

3.4.5.11 Метод runTrans

Метод runTrans вызывается при получении POST-запроса по маршруту /editor/runTrans. Запрос должен содержать два значения: имя транслируемого файла и проект, ассоциируемый с файлом. Результатом работы данного метода является отладочная информация, выводимая инструментальным средством языка «Пифагор» – «trans».

3.4.5.12 Метод runCgen

Метод runCgen вызывается при получении POST-запроса по маршруту /editor/runCgen. Алгоритм работы данного метода фактически идентичен методу runTrans, за исключением вызываемого метода модели, которым в данном случае является runCgen.

3.4.5.13 Метод runInter

Метод runInter вызывается при получении POST-запроса по маршруту /editor/runCgen. Запрос должен содержать имя запускаемой функции, имя проекта, булеву переменную arg_required, отвечающую за необходимость использования функцией аргумента, а также аргумент функции. Если булева переменная arg_required имеет значение true вызывается метод модели prepareArg, в случае ошибки работы которого формируется сообщение, описывающее ошибку и отправляющееся клиенту. Если arg_required имеет значение false или метод prepareArg завершил работу без ошибок, вызывается метод модели runInter. Результатом работы метода в таком случае является информация, выводимая инструментальным средством языка «Пифагор» – «inter».

3.4.5.14 Метод runProject

Метод runProject вызывается при получении POST-запроса по маршруту /editor/runProject. Запрос должен содержать все те же значения, что и значения запроса метода runInter, а также имя файла. Данный метод отвечает за запуск следующих методов модели: runTrans, runCgen, prepareArg, runInter. Если при выполнении любого из методов происходит ошибка, формируется сообщение об ошибке которое отправляется клиенту. В случае успешного завершения выполнения всех методов результатом работы метода является информация, выводимая всеми инструментальными средствами языка «Пифагор», запуск которых происходит при вызове методов модели.

3.5 Выводы по главе 3

Для осуществления инструментальной поддержки написания и выполнения программ на функциональном языке параллельного программирования «Пифагор» была реализована интегрированная среда разработки с использованием Интернет технологий.

В соответствии с проектными решениями система должна выполнять следующие функции:

1. Предоставляет возможности по работе с файловой системой;
2. Предоставляет возможность запуска и работы с инструментальными средствами языка «Пифагор»;
3. Обладает возможностями по регистрации и авторизации;
4. Предоставляет возможности по разработке в рамках интегрированной среды.

4 Описание разработанной системы

4.1 Интерфейс пользователя

Интерфейс пользователя включает в себя:

- Редактор кода;

- Вкладки редактора;

- Список проектов и файловый проводник;

- Семь диалоговых окон;

- Двенадцать кнопок;

- Поле для вывода изображений, сгенерированных средствами языка «Пифагор»;

- Поле для вывода информации о результате выполнения кода;

- Информационное поле с просьбой ознакомиться с руководством пользователя.

При открытии страницы пользователю представляется список проектов (пуст изначально) и предложение ознакомиться с руководством пользователя.

Форма страницы с основной функциональностью представлена на рисунке 18.

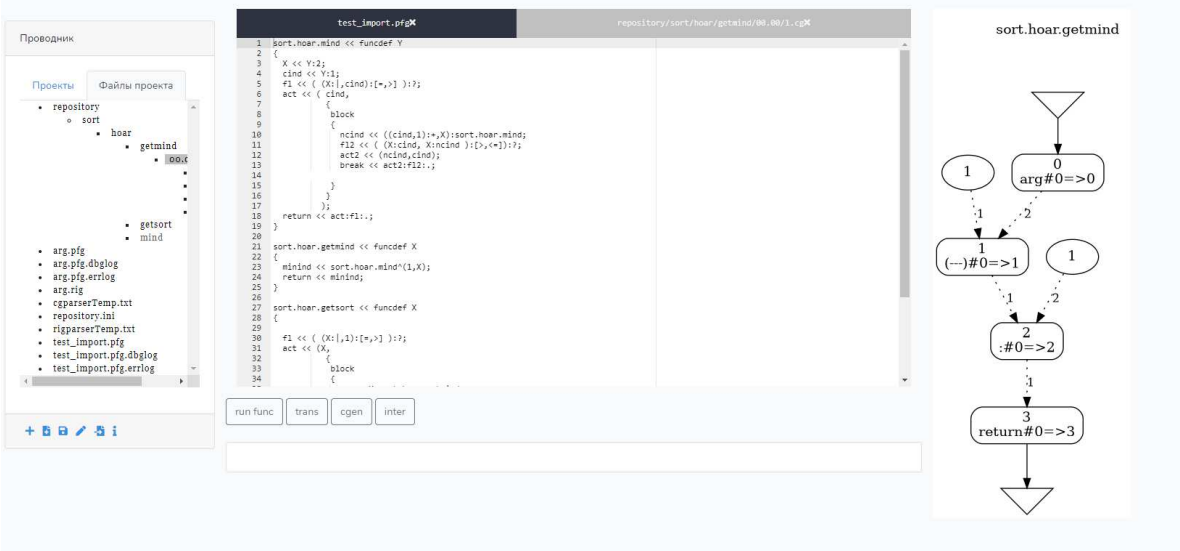

Рисунок 18 – Страница ИСР с частью видимых элементов

Основными элементами интерфейса являются:

1) Проводник и его блок кнопок;
2) Редактор кода;
3) Вкладки редактора;
4) Блок кнопок для запуска средств языка «Пифагор»;
5) Поле для вывода изображения;
6) Поле для вывода результата выполнения запуска средств языка «Пифагор».

А также диалоговые окна:

1) Окно создания проекта;
2) Окно создания файла;
3) Окно со списком проектов и их кодов импорта;
4) Окно импорта проекта;
5) Окно руководства пользователя;
6) Окно ввода информации для средств языка «Пифагор».

4.1.1 Проводник

Проводник имеет два режима – режим списка проектов и режим древовидного отображения структуры проекта. Эти режимы, а также блок кнопок проводника представлены на рисунках 19, 20 и 21.

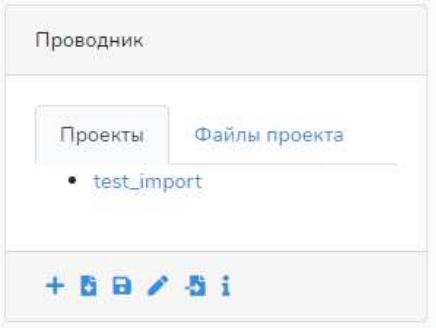

Рисунок 19 – Проводник, режим списка проектов

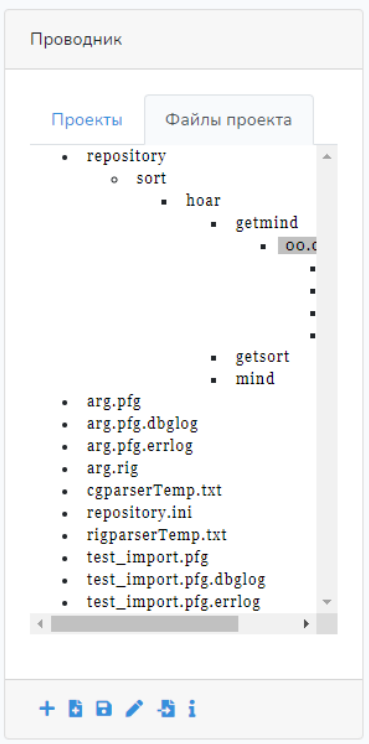

Рисунок 20 – Проводник, режим древовидного отображения структуры проекта

Рисунок 21 – Блок кнопок проводника

Кнопка 1 называется кнопкой создания проекта и отвечает за вызов диалогового окна создания проекта (представленное на рисунке 22).

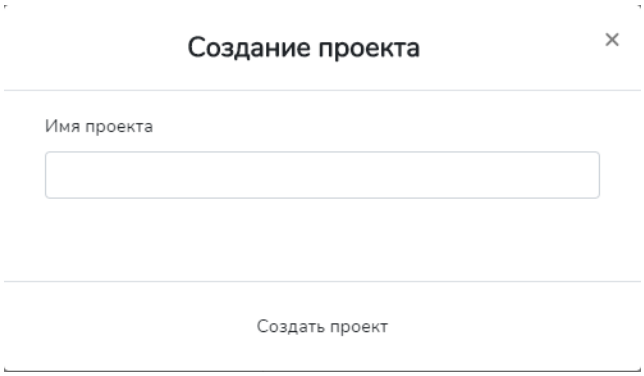

Рисунок 22 – Диалоговое окно создания проекта

Кнопка 2 называется кнопкой создания файла и отвечает за вызов диалогового окна создания файла (представленное на рисунке 23).

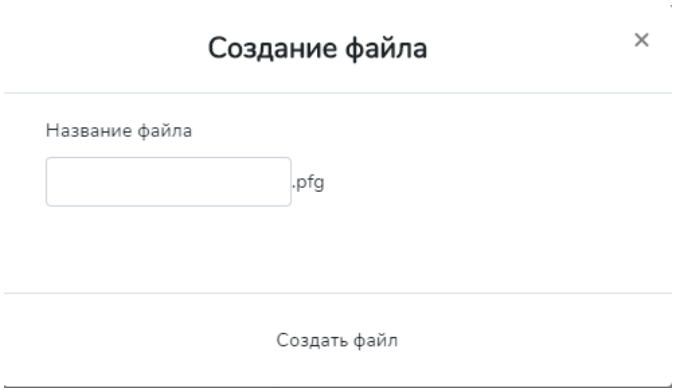

Рисунок 23 – Диалоговое окно создания файла

Кнопка 3 называется кнопкой сохранения файла и отвечает за сохранение написанного в редакторе кода в соответствующий файл.

Кнопка 4 называется кнопкой окна экспорта и отвечает за вызов окна экспорта проектов, представленное на рисунке 24.

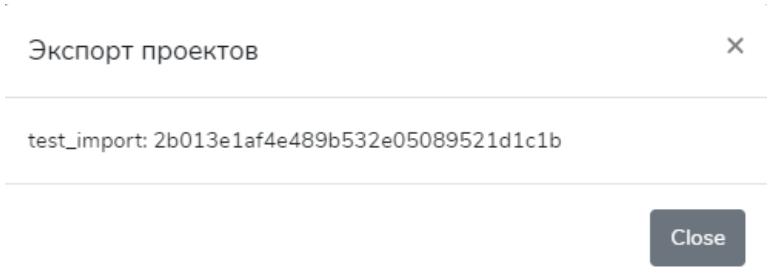

Рисунок 24 – Окно экспорта проектов

Кнопка 5 называется кнопкой импорта проекта и отвечает за вызов окна импорта проекта, представленное на рисунке 25.

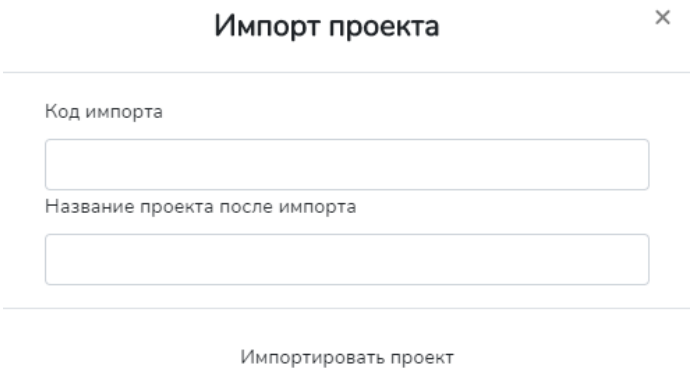

Рисунок 25 – Окно импорта проекта

Кнопка 6 называется кнопкой руководства пользователя и вызывает окно с кратким руководством по использованию ИСР. Окно руководства представлено на рисунке 26.

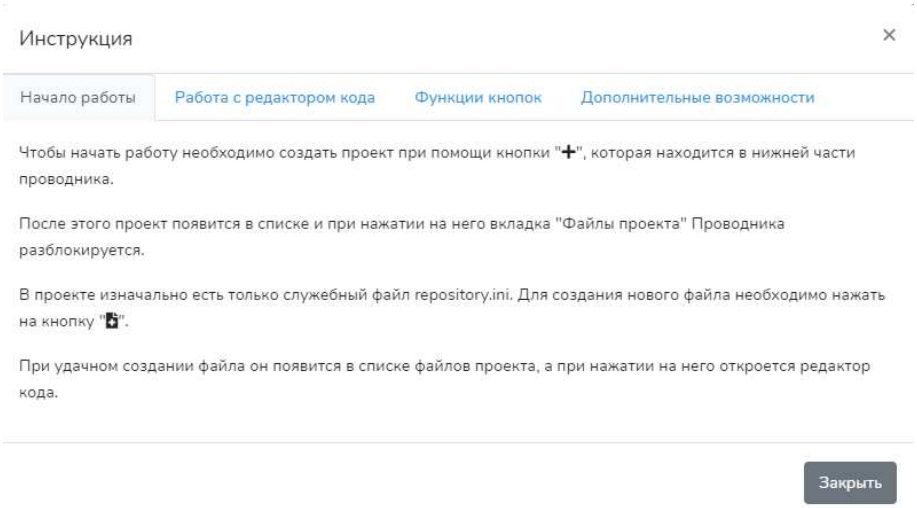

Рисунок 26 – Окно руководства пользователя

4.1.2 Поля вывода

Поля вывода предназначены для вывода результатов запуска инструментальных средств языка «Пифагор», описанных в главе 2.1 и ее подглавах, и представлены на рисунках 27 и 28.

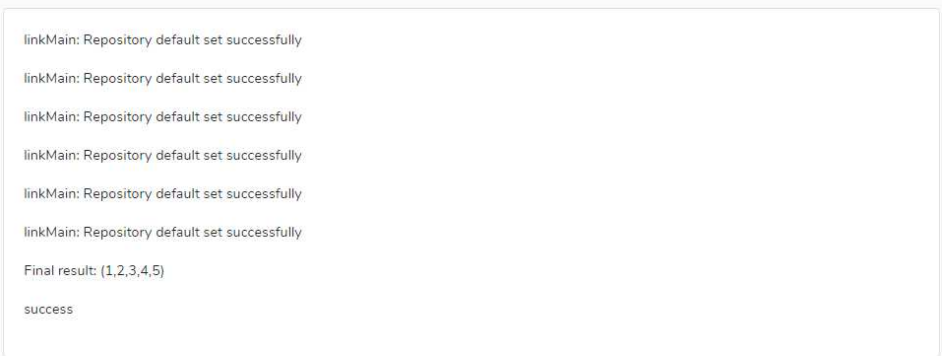

Рисунок 27 – Поле вывода результатов запуска trans, inter, cgen

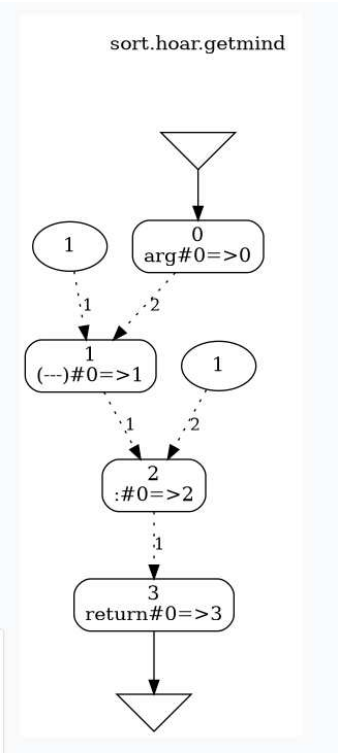

Рисунок 28 – Поле вывода результата запуска rig2dot, cg2dot

4.1.3 Блок кнопок редактора кода

Блок кнопок редактора кода изображен на рисунке 29.

Рисунок 29 – Блок кнопок редактора кода

Кнопка «run func» отвечает за вызов диалогового окна запуска функции, представленного на рисунке 30.

Кнопка «inter» также отвечает за вызов диалогового окна запуска функции, ее отличием от кнопки «run func» является маршрут, по которому отправляется запрос.

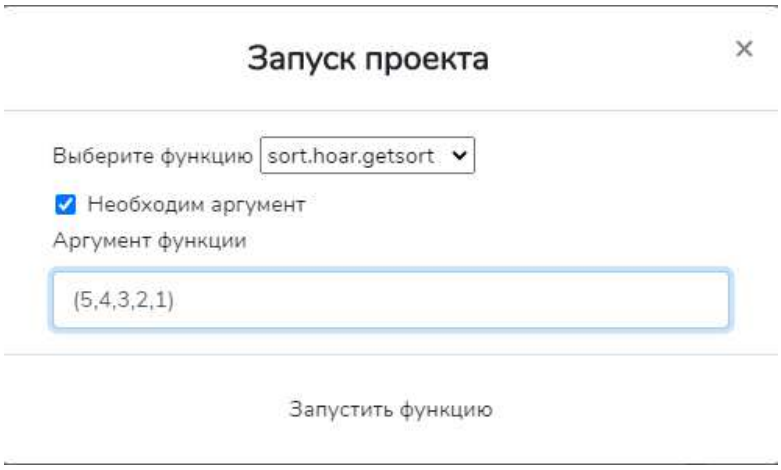

Рисунок 30 – Диалоговое окно запуска функции

Кнопка «Запустить функцию» диалогового окна запуска функции отвечает за вызов функции runTool (глава 3.1.5).

Кнопки «trans», «cgen», «rig2png», «cg2png» также вызывают функцию runTool.

4.2 Описание сценария работы с системой

При открытии портала пользователя встречает домашняя страница, на которой присутствует только навигационная панель с пунктами «регистрация», «авторизация» и переход к ИСР. Переход к ИСР возможен только для авторизованных пользователей. Не авторизованных пользователей система автоматически перенаправляет на окно авторизации.

После регистрации пользователь переходит к странице интегрированной среды разработки (/editor) и создает свой первый проект,

путем нажатия на кнопку создания проекта и ввода его названия в появившемся диалоговом окне.

Следующим действием пользователь открывает созданный проект и создает новый файл, также вводя его название в специальное поле, отведенное для этого.

После создания нового файла пользователь кликает по его представлению в древовидном списке файлов проекта, что вызывает загрузку файла с сервера, и, в свою очередь, появление на экране редактора кода, его блока кнопок и первой вкладки редактора.

После написания пользователем программы, он сохраняет изменения и нажимает на кнопку run func, вследствие чего появляется диалоговое окно с просьбой выбрать имя функции, которая будет запущена инструментальным средство языка «Пифагор» – «inter», а также по необходимости, поле ввода аргумента функции. После ввода информации пользователь нажимает на кнопку «Запустить функцию» и наблюдает информацию, выведенную в процессе запуска инструментальных средств языка «Пифагор», а также результат работы функции в поле для вывода данной информации.

Следующим действием пользователь импортирует проект другого пользователя, путем нажатия на кнопку, отвечающую за вызов диалогового окна импорта проекта и ввода кода импорта в соответствующее поле. Пользователь открывает импортированный проект и находит в древовидном представлении файлов проекта файл с расширением .cg, открывает его и нажимает на кнопку «cg2png», что приводит к выводу графического представления управляющего графа в соответствующее поле.

4.3 Выводы по главе 4

Для разработанной системы было проведено описание интерфейса пользователя, а также сценария работы с системой. Из описания видно, что использование среды обеспечило инструментальную поддержку процесса

разработки функционально-потоковых параллельных программ без применения командной строки, что позволило повысить эффективность.

Заключение

В результате выполнения работы были получены следующие результаты:

1. Проведен анализ существующих решений в области ИСР, реализованных с использованием Интернет технологий, на основе которого была выявлена необходимая функциональность;
2. Проведен анализ существующих решений в области встраиваемых редакторов кода, на основе которого был сделан выбор одного из них как инструментального средства разработки;
3. Проведен анализ существующих решений в области фреймворков языка PHP, на основе которого был сделан выбор одного из них как инструментального средства разработки;
4. Проведено описание инструментальных средств, в ходе которого были выявлены особенности их использования;
5. Реализована ИСР для языка «Пифагор» с использованием Интернет технологий;
6. Проведено описание реализованной ИСР.

Реализованная ИСР позволяет производить разработку без необходимости в загрузке каких-либо инструментальных средств и дополнительной нагрузки, связанной с работой этих инструментальных средств.

Список использованных источников

1. Ajantha Dahanayake. Integrated Development Environments – [Электронный ресурс] / Режим доступа: https://onlinelibrary.wiley.com/doi/full/10.1002/9781118785317.weom070024 (дата обращения: 24.05.2019)
2. Конников Павел Владимирович. Интегрированная инструментальная среда как средство обучения гибким технологиям разработки программного обеспечения [Электронный ресурс] / Режим доступа: https://elibrary.ru/item.asp?id=17780420 (дата обращения 10.06.2019)
3. Неизвестный автор, 16 лучших сред для веб-разработки [Электронный ресурс] / Режим доступа: https://proglib.io/p/webdev-editors/ (дата обращения 14.06.2019)
4. Неизвестный автор, Web IDEs: The Future of Coding [Электронный ресурс] / Режим доступа: https://devrix.com/tutorial/web-ides-future-coding/ (дата обращения: 21.06.2020)
5. Легалов А.И. Функциональный язык для создания архитектурно – независимых параллельных программ / А. И. Легалов // Вычислительные технологии / Институт вычислительных технологий СО РАН – Новосибирск, 2005. – Т.10, №1. – С. 71–89.
6. Неизвестный автор, IDE нормального человека или почему мы выбрали Monaco [Электронный ресурс] / Режим доступа https://habr.com/ru/company/Voximplant/blog/445390/ (дата обращения: 03.06.2019)
7. Документация Monaco Editor [Электронный ресурс] / Режим доступа: https://microsoft.github.io/monaco-editor/ (дата обращения: 05.06.2020)
8. Mikhail Khorpyakov, Embedding Monaco Editor into a Web Application [Электронный ресурс] / Режим доступа: https://database.blog/integrating-monaco-editor/ (дата обращения: 30.05.2020)

9. Документация CodeMirror [Электронный ресурс] / Режим доступа: https://codemirror.net/ (дата обращения: 18.06.2020)
10. Документация Ace Editor [Электронный ресурс] / Режим доступа: https://ace.c9.io/ (дата обращения 14.06.2020)
11. Wikipedia, CodeAnywhere [Электронный ресурс] / Режим доступа: https://en.wikipedia.org/wiki/Codeanywhere (дата обращения: 12.06.2020)
12. Неизвестный автор, Инструменты для веб-разработки: как выбрать IDE? [Электронный ресурс] / Режим доступа: https://webformyself.com/instrumenty-dlya-veb-razrabotki-kak-vybrat-ide/ (дата обращения (22.06.2020)
13. Eclipse Orion Version History [Электронный ресурс] / Режим доступа: https://projects.eclipse.org/projects/ecd.orion (дата обращения: 21.06.2020)
14. Bart Baesens, Aimee Backiel, Seppe vanden Broucke. Setting Up Your Development Environment / Режим доступа: https://onlinelibrary.wiley.com/doi/10.1002/9781119209416.ch3 (дата обращения 01.06.2019)
15. The App Solutions, Top 4 PHP frameworks to use in 2019 [Электронный ресурс] / Режим доступа: https://theappsolutions.com/blog/reviews/top-4-php-frameworks-2019/ (дата обращения: 15.06.2020)
16. Документация Laravel 6.x [Электронный ресурс] / Режим доступа: https://laravel.com/docs/6.x/ (дата обращения: 12.06.2020)
17. Yii Framework 2.0 API documentation [Электронный ресурс] / Режим доступа: https://www.yiiframework.com/doc/api/2.0 (дата обращения: 12.06.2020)
18. И.В. Матковский, А.И. Легалов. Инструментальная поддержка трансляции и выполнения функционально-потоковых параллельных программ [Электронный ресурс] / Режим доступа: http://ikit.sfu-kras.ru/files/ikit/06_Instrumentalnaya_podderzhka.pdf (дата обращения: 11.06.2020)

19. Легалов А.И., Васильев В.С., Матковский И.В., Ушакова М.С. Инструментальная поддержка создания и трансформации функционально-потоковых параллельных программ. Труды ИСП РАН, том 29, вып. 5, 2017 г., стр. 165-184. DOI: 10.15514/ISPRAS-2017-29(5)-10
20. Документация языка PHP [Электронный ресурс] / Режим доступа: https://www.php.net/docs.php (дата обращения: 12.06.2020)

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«___»_____2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Учебно-исследовательский портал для поддержки
архитектурно-независимого параллельного программирования
тема

09.04.01 Информатика и вычислительная техника
код и наименование направления

09.04.01.04 Технология разработки программного обеспечения
код и наименование магистерской программы

Научный руководитель _____ профессор, д.т.н. А. И. Легалов
подпись, дата должность, учёная степень инициалы, фамилия

Выпускник Косм 29.06.20 Д. Е. Костыгин
подпись, дата инициалы, фамилия

Рецензент _____ доцент, д.ф.-м.н. К. В. Сафонов
подпись, дата должность, учёная степень инициалы, фамилия

Нормоконтролер _____ профессор, д.т.н. А. И. Легалов
подпись, дата должность, учёная степень инициалы, фамилия

Красноярск 2020