МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра компьютерных технологий

Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Программная инженерия и компьютерные науки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Кузнецова Егора Владимировича

Тема работы:

РАЗРАБОТКА МОДУЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ ДЛЯ ОБЛАЧНОГО IDE ЯЗЫКА REFLEX

«К защите допущена»

Заведующий каф. КТ
ФИТ НГУ, д.т.н., доцент
Зюбин В.Е. /……..
(ФИО) / (подпись)
«……»……2024г.

Руководитель ВКР

д.т.н., доцент,
зав. каф. КТ ФИТ НГУ
Зюбин В.Е. /……...
(ФИО) / (подпись)
«……»……2024г.

Соруководитель ВКР
к.ф.-м.н., доцент
каф. КТ ФИТ НГУ
Гаранина Н.О./……..
(ФИО) / (подпись)
«…»……2024г.

Новосибирск, 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра компьютерных технологий
(название кафедры)

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль): Программная инженерия и компьютерные науки

УТВЕРЖДАЮ

Зав. кафедрой Зюбин В. Е.
(фамилия, И., О.)

…….
(подпись)

«02» ноября 2023г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студенту *Кузнецову Егору Владимировичу, группы 20204*
(фамилия, имя, отчество, номер группы)

Тема Разработка модуля управления проектами для облачного IDE языка Reflex
(полное название темы выпускной квалификационной работы)

утверждена распоряжением проректора по учебной работе от 02.11.2023 г. №0409

Срок сдачи студентом готовой работы «20» мая 2024 г.

Исходные данные (или цель работы): разработка модуля управления проектами для облачного IDE языка Reflex

Структурные части работы: анализ подходов к управлению проектами IDE, архитектура модуля и используемые методы, проектирование интерфейса, реализация, эксперимент

Консультанты по разделам ВКР отсутствуют

Руководитель ВКР
Заведующий кафедрой КТ
ФИТ НГУ, д.т.н., доцент
Зюбин В.Е./……..
(ФИО) / (подпись)
«02» ноября 2023г.

Задание принял к исполнению

Кузнецов Е.В./……..
(ФИО студента) / (подпись)

«02» ноября 2023г.

Соруководитель ВКР
Доцент кафедры КТ
ФИТ НГУ, к.ф.-м.н.
Гаранина Н.О./……..
(ФИО) / (подпись)

«02» ноября 2023г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Reflex – это процесс-ориентированный язык программирования, разрабатываемый в Институте автоматики и электрометрии СО РАН (ИАиЭ СО РАН), для киберфизических систем на базе программируемых логических контроллеров (ПЛК). Необходимость в таком языке резко возросла в связи с популярностью IoT, встраиваемых систем, умных устройств и т.д.. Идея разработки такого языка не нова, однако существующие решения не нашли своего широкого применения. С учетом накопленного опыта в индустрии, был произведен комплексный анализ, на базе которого был создан уникальный понятный и простой язык для людей уже работающих в отрасли.

Кроме того, чтобы стать популярным, удобным и современным языку необходима соответствующая среда разработки. Для Reflex уже разрабатывается IDE под названием RIDE построенная на базе Eclipse Theia в ИАиЭ СО РАН. Eclipse Theia была выбрана потому, что ее можно запускать в браузере, это проект с открытым исходным кодом. Theia - однопользовательское приложение, поэтому проект RIDE создается для одновременного использования командой разработчиков.

Для разработки программы на Reflex, пользователю RIDE необходимо создать проект. Внутри проекта уже есть поддержка трансляции в C, Python и Formula Node от LabVIEW, подсветка синтаксиса Reflex. Однако существуют трудности для совместной разработки, для создания своего первого проекта и не подсвечивается структура Reflex проекта. Уже была упомянута “простота и понятность”, как одна из идей разработки языка Reflex, то необходим модуль, упрощающий управление проектом.

Таким образом, цель нашей текущей работы – разработать модуль управления проектами в облачном IDE языка Reflex.

Задачи, которые необходимо решить:

- проанализировать существующие подходы к организации управления проектами в облачных IDE и специфику IDE языка Reflex;
- сформировать список требований к разрабатываемому модулю;
- спроектировать интерфейс модуля управления проектами;
- спроектировать архитектуру модуля управления проектами;
- реализовать модуль;
- интегрировать модуль в существующую IDE языка Reflex.

В результате проведенного исследования для web IDE языка Reflex был разработан модуль управления проектами, предоставляющий разработчику возможность: создавать

проекты по шаблону, определять область видимости проекта, условия переиспользования, подключать к проекту сторонних разработчиков и устанавливать им роли (редактор, комментатор, читатель), объединять в проект файлы, читать сопроводительную документацию, и осуществлять сборку проекта под определенную модель микроконтроллера. Разработанный модуль сокращает трудоемкость сопровождения проектов в web IDE языка Reflex.

Работа изложена в пяти главах. В первой главе анализируются существующие облачные среды разработки, как внутри каждой реализован модуль управления проектом, и приводятся требования, сформированные на основе полученной при анализе таблицы. Во второй главе описываются архитектура модуля и используемые методы. В третья глава посвящена вопросам проектирования графического интерфейса модуля. В четвертой подробно описывается реализация, какие технологии использовались. В пятой описывается эксперимент, включавший в себя тестирование разрабатываемых компонентов модуля и запуск на тестовом стенде.

1. Анализ существующих подходов к управлению проектами в IDE

Говоря про тренды в области комплексов программных средств, используемых программистами для разработки программного обеспечения (ПО), важно отметить следующий факт: реализация в виде нативного desktop приложения не принесёт инноваций и станет источником для наложения ряда ограничений для доступной совместной разработки. Во-первых с инженера снимается обязанность инсталляции и своевременного обновления IDE и версий poST и Reflex. Во-вторых пользователь получает все плюсы облачных решений. В статье “An overview of platforms for cloud based development” [1] в главе про среды программирования авторы отмечают следующее:

- Разработчики имеют возможность получить доступ к коду с любого устройства, где установлен веб-браузер;
- Доступность для любой организации, где существует проблема с ограниченными ресурсами;
- Экономия на специалистах, которые занимаются установкой и настройкой IDE;
- С клиента снимается ответственность за развертку в облаке;
- Удобство тестирования;
- Популярность не только среди студентов, но и специалистов. Например, AWS Cloud9 [2].

В-третьих, среде IIOT (Industrial internet of things) роль “облака” ключевая. Индустрию 4.0 можно описать, как большое количество информации, собираемое с большого количества датчиков и обрабатываемое в режиме реального времени [3]. Поэтому логичным шагом является разработка в “облаке” для “облака”.

В-четвертых, нужно брать во внимание тот факт, что индустрия - это прежде всего команда, которая должна активно взаимодействовать. Именно поэтому был сделан выбор в пользу создания облачного web IDE. Так, компания Red Hat активно использует Eclipse Che внутри своей платформы OpenShift [4], в результате чего это повышает производительность команд и помогает быстрой адаптации новых сотрудников [5].

Выше уже были упомянуты некоторые популярные среды для разработки в облаке. Конечно, это только малая часть, необходимая для последующего анализа существующих решений. Поэтому полный список будет выглядеть следующим образом: Codecollab.io, Replit.com, Visual Studio Code, IntelliJ IDEA, Remix Etherium IDE, Arduino Cloud Web Editor, Eclipse Theia, Tinkercad, Eclipse Che, Cloud9. Отметим, что справедливо будет возразить

следующее: Visual Studio Code[6] и IntelliJ IDEA[7] не являются облачными средами разработки. Включение их в текущий список было выполнено исходя из:

- Их широкой популярности [8];
- Возможности использования VS Code и IntelliJ в Eclipse Che, которая является облачным IDE;
- Мы имеем возможность использовать VS Code в web браузере для просмотра Azure и Github репозиториев. Кроме этого упомянутой среде разработки существует Live Share, который предлагает делиться сессией для просмотра в web браузере. В свою очередь IntelliJ IDEA предоставляет внутри себя Code with me, который очень похож на Live Share.

1.1. Codecollab.io

Codecollab.io – пример web IDE для совместной разработки [9]. Проект был создан студентами из Сингапура, достаточно простой и понятный в силу развитости своего графического интерфейса.

Встречает приятный сайт-презентация (рисунок 1). Регистрация: github, gmail, microsoft (рисунок 2). После входа можно начать создания проекта. В форме на создание проекта нужно указать: название проекта, язык программирования, страна сервера (доступно три), краткое описание (рисунок 3). После этого начинается загрузка: сопровождается анимацией (рисунок 4).

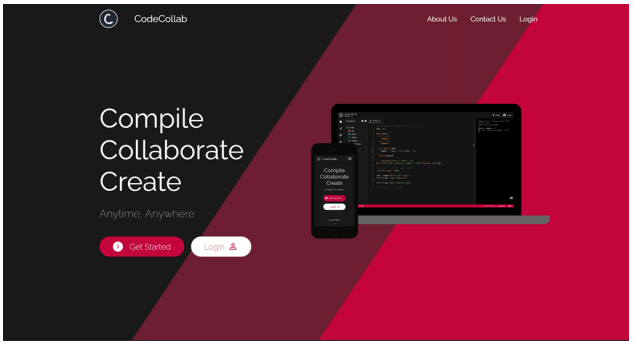


Рисунок 1 – Главная страница

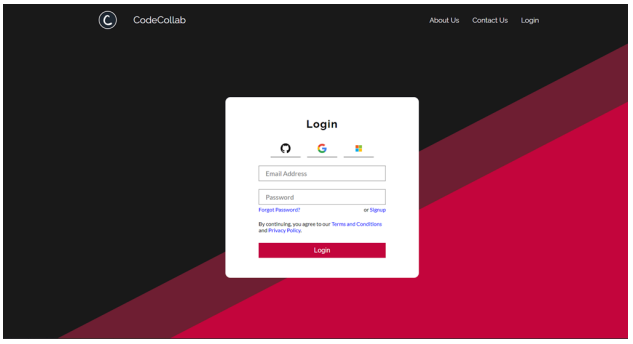


Рисунок 2 – Окно входа и регистрации

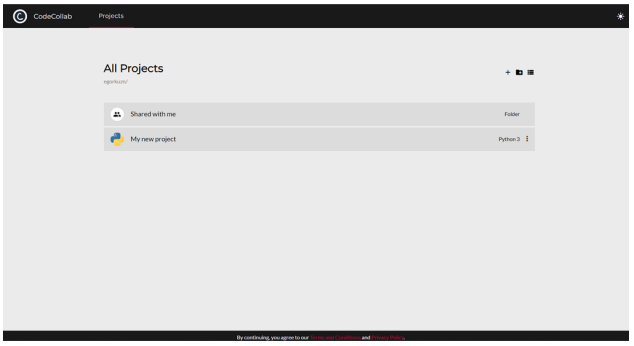


Рисунок 3 – Меню проектов

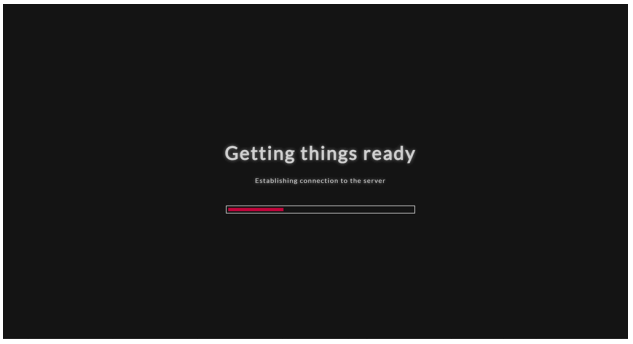


Рисунок 4 – Загрузка проекта

Опишем, что видит пользователь после загрузки проекта. Как видно на рисунке 5, слева - главная вертикальная панель. Сверху вниз: файлы ресурсов, кнопка поделиться (ссылка и возможность выбрать режим “только для чтения”), выгрузить на диск, настройка интерфейса. Снизу: окно терминала (shell) linux-подобной системы и консоль для ввода/вывода. Справа и снизу: чат для взаимодействия участников проекта. Вверху: run для запуска программы, “кружочки”, отражающие активных пользователей.

В самой нижней части рабочего пространства слева-направо на красной панели: выбор языка, настройка размера tab, размер файла, настройка темы (около 16 вариантов) и число “collaborators” (активных участников проекта).

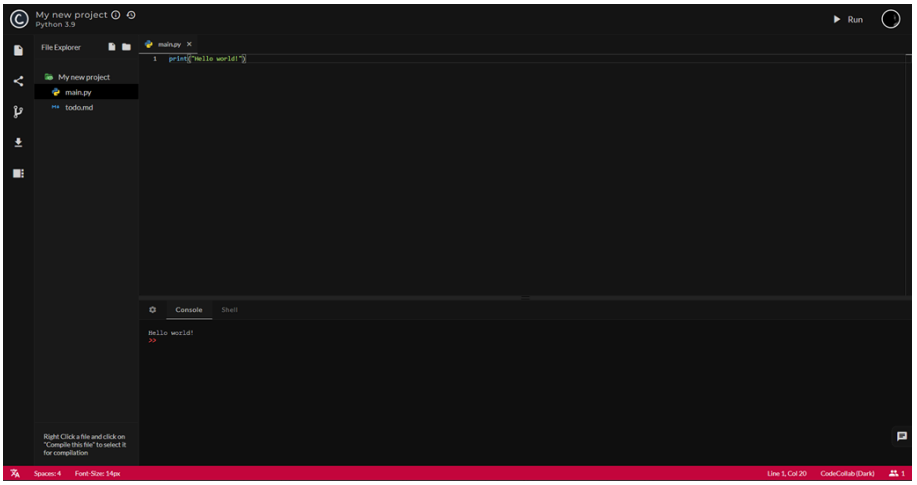


Рисунок 5 – Окно проекта

К большому сожалению, данный проект прекратил свою работу 31го мая 2023 года.

1.2. Replit.com

Replit.com в отличие от codecollab имеет другую философию: нацелен больше на обучение и коммуникацию между участниками Replit [10]. Интерфейс самой среды разработки один самых приятных.

Регистрация: отличается лишь тем, что сервис спрашивает цель использования инструмента. Как показано на рисунке 6, после данной процедуры входа, появляется лента с трендами и другими интересными репозиториями. Слева содержится блок, как в классических социальных сетях, но переработанное под логику IDE: home(домашняя), templates(шаблоны), my repl(мои репозитории), my cycles(баланс/подписка), community(форумы), bounties(возможность заработка за решение чужих задач), learn(курсы/лекции), teams(возможность четкой организации команд), curriculum(обучение/библиотека). Ещё ниже важный консультационный блок с Blog, About, Careers, Pricing, Discord, Terms и (что важно) большая кнопка Help. Рядом - смена темы светлая/тёмная.

После знакомства, перейдём к созданию проекта. Ничем не отличается от 1.1 и представлено на рисунке 7. Окно IDE (рисунок 8): разделяется на три главных вертикальных блока, которые динамичные по ширине: на максимум расширяются и полностью скрываются.

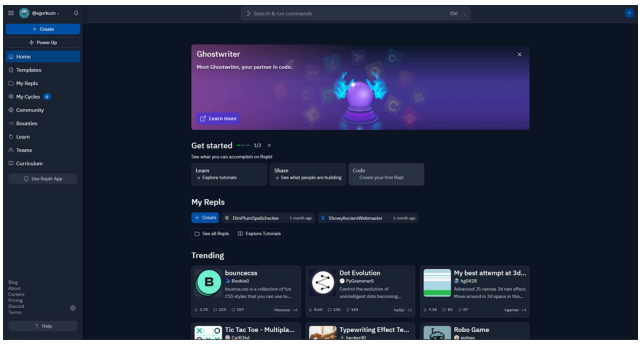


Рисунок 6 – Главная страница

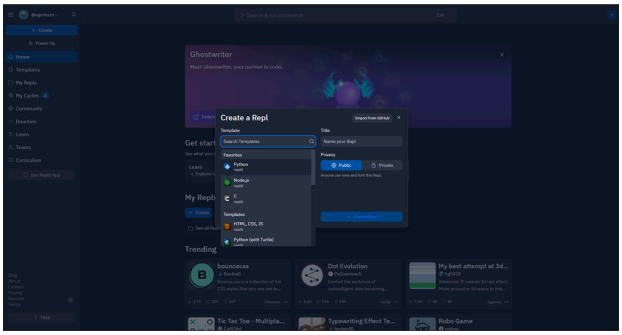


Рисунок 7 – Окно создания проекта

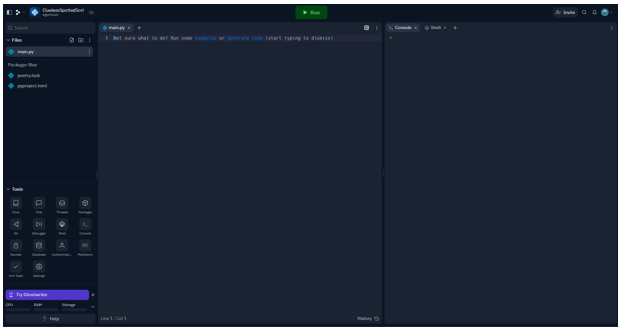


Рисунок 8 – Окно проекта

Первый блок в окне проекта: файлы/директории в верхней половине, инструменты (для документирования, система контроля версий, дебаггер, shell/console, unit-тестирование, базы данных, auth, настройки) в нижней половине. В самом низу блок отражающий мониторинг ресурсов, используемых пользователем, а также большой help, чтобы пользователь в случае проблемы не растерялся.

Второй блок: сама область ввода кода. Можно открывать несколько файлов и возможно разными способами конфигурировать их многооконное представление. Очень хорошо реализована анимация: крайне красиво, плавно и быстро. Над самим блоком большая зеленая кнопка run для запуска - это очень удобно, т.к. приходится часто запускать код.

Третий блок - shell и console. Предоставляет доступ к Linux и производит ввод и вывод, используя основные потоки.

Вверху справа кнопка для приглашения нового участника по почте, поиск по проекту, кнопка уведомлений и кнопка профиля пользователя. Слева - большая вытянутая горизонтальная кнопка, включающая в себя иконку название проекта, его создателя и индикатор синхронизации с облаком. И именно она скрывает под собой настройки:

- Изменение названия/описания проекта;
- Установление режима private/public, т.к. replit.com прежде всего соцсеть;
- Boost для добавления ресурсов;
- Publish кнопка, которая позволяет поделиться ссылкой для общего доступа к проекту. Т.е. не нужно отдельно каждого пользователя добавлять.

1.3. Visual Studio Code

Visual Studio Code [6] занимает второе место в списке TOP IDE index[8]. Его стоит рассматривать в силу того, что данный инструмент универсален, поддерживает большое количество языков и дополнений, open-source и также имеет возможности как для облачной, так и для совместной разработки.

VS Code достаточно гибок в настройке интерфейса, который представлен на рисунке 9. Около десятка стандартных тем и сотни плагинов для оформления. Сама система изначально представляет собой блокнот с подсветкой синтаксиса базовых языков программирования, поддержкой простого функционала. Однако, развитый магазин дополнений, продемонстрированный на рисунке 10, добавляет VS Code возможности для работы с различными языками, технологиями и фреймворками. Дополнения могут быть совершенно разными, что даёт свободу разработчику. Доступное API VS Code и умение

программировать на JavaScript - все, что требуется для создания своего дополнения для магазина VS Code. В силу большого числа приложений-дополнений и их разработчиков, было решение помечать верифицированных разработчиков: специальный значок можно найти у Microsoft, Red Hat, GitHub и других официальных разработчиков, и компаний. Можно сделать дополнение платным.

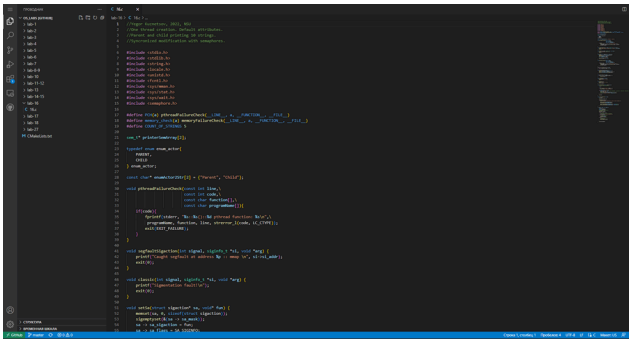


Рисунок 9 – Окно проекта

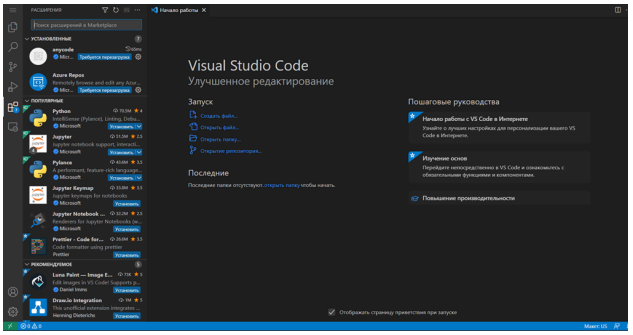


Рисунок 10 – Магазин дополнений

Идея того, что пользователь сам создает свое “идеальное” IDE высоко оценена сообществом, и это делает VS Code популярным. Он стал базой для многих других облачных IDE, т.к. уже был написан на JavaScript. Примерами могут быть Codecollab.io (п. 1.1.), Remix Etherium IDE (п. 1.5.), облачное IDE Theia, позволяющее использовать плагины VS Code и многие другие среды разработки.

Конкуренты: Atom IDE от GitHub (также JavaScript, ещё более развитые плагины и тоже open-source в Github) и Sublime Text (C++ и Python, имеет платные версии). VS Code поддерживается Eclipse Che [11].

1.4. IntelliJ IDEA

IntelliJ IDEA одна из самых продвинутых IDE на рынке, флагманский продукт JetBrains [12]. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

К сожалению, данное IDE представлено только в виде desktop-приложений, но у него можно позаимствовать логику для разработки крупных проектов. Так, в проекте разделяются файлы ресурсов, файлы тестов, целевые файлы (то, что создается в процессе сборки проекта и конечная программа). Как можно видеть на рисунке 11, стиль проектов в обычном IDEA проекта - размещение файлов по пакетам, а множество пакетов по модулям. Это все реализовано благодаря тому, что среда была создана для ООП языков (Java/Kotlin). Но это не

мешает тому, что мы можем разбивать крупные poST program как пакеты, а process как файлы в program-пакетах.

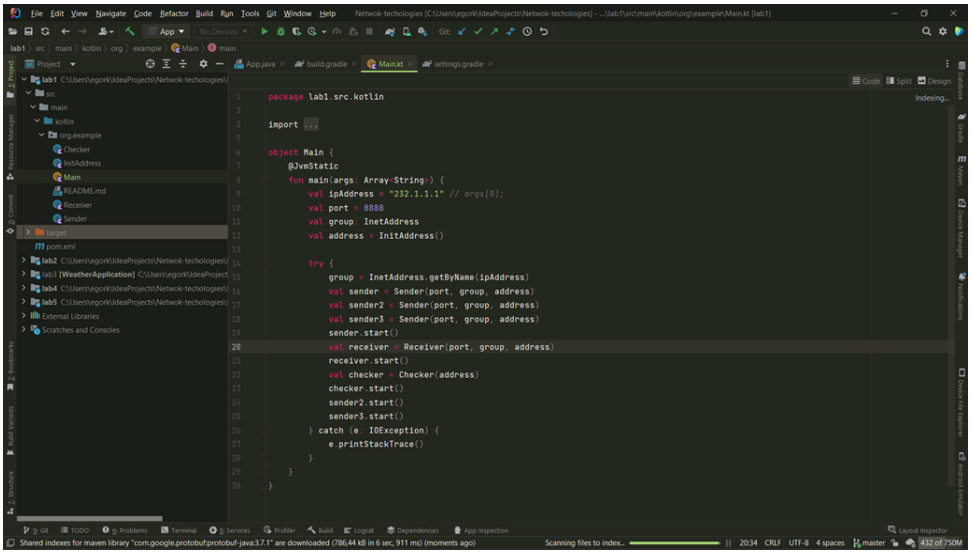


Рисунок 11 – Окно проекта

IntelliJ IDEA располагает анализатором кода, который помогает исправлять ошибки, подгружать зависимости, предлагает заменять участки кода на те, что по версии IDE, будут лучше. Магазин плагинов также содержит большое количество удобных инструментов и то, как он выглядит, можно увидеть на рисунке 12. Платный контент, приложения от верифицированных разработчиков в IDE присутствуют. IntelliJ IDEA поддерживается Eclipse Che [11].

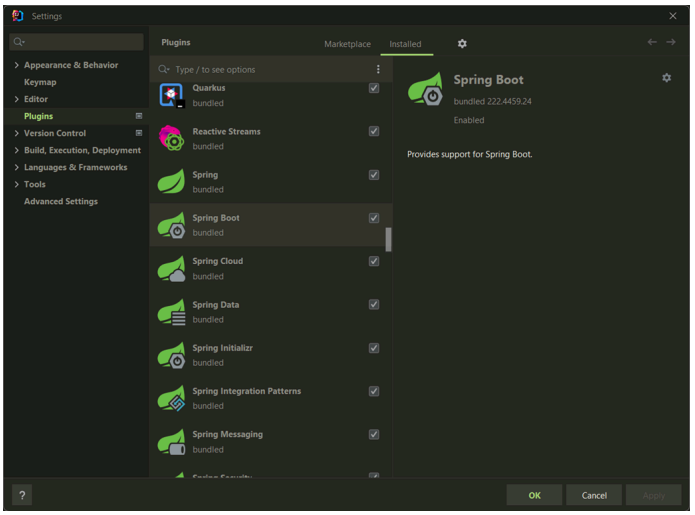


Рисунок 12 – Магазин плагинов

1.5. Remix Etherium IDE

Remix Etherium IDE - облачная среда разработки для описания смарт-контрактов. Стала достаточно популярной из-за повсеместного внедрения блокчейн технологий [13]. Базируется на open-source проекте Remix, разработанном на React JS.

Интерфейс: простой и практичный. Имеет много похожих с VS Code элементов: широкая вертикальная панель слева, на которой размещены разные приложения/плагины, магазин плагинов. Всё это изображено на рисунке 13.

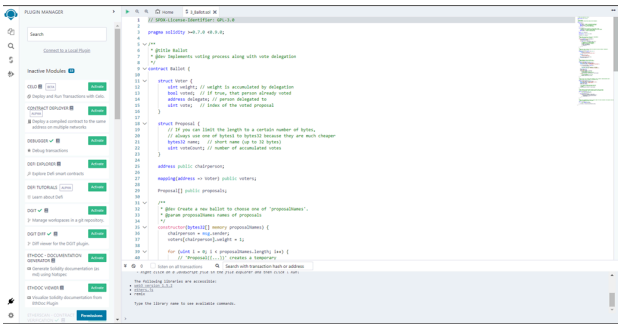


Рисунок 13 – Пример проекта с открытой панелью плагинов

Отдельно стоит обратить внимание на главную вкладку. Во-первых, сразу на рисунке 14 видим все языки, на которых можно описывать смарт-контракты. Во-вторых, там же есть список ресурсов, где можно решить возникшие проблемы (ссылки на соцсети и форумы). В-третьих, get started блок – информация об обновлениях и полезные советы для начинающих. Имеется быстрый поиск документации, а это значит, что не нужно тратить время на поиск информации на сторонних сайтах.

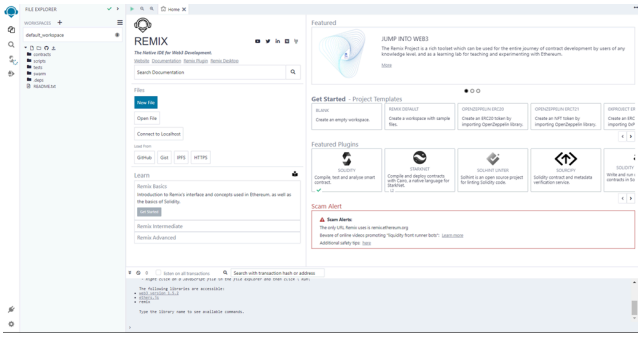


Рисунок 14 – Окно настроек проекта

1.6. Arduino Cloud Web Editor

Arduino Cloud Web Editor – облачная среда разработки для программирования микроконтроллеров Arduino [14]. Используется как для создания автономных объектов, так и

подключения к программному обеспечению через проводные и беспроводные интерфейсы. Подходит для начинающих пользователей с минимальным входным порогом знаний в области разработки электроники и программирования. Появление web версии было после успеха Arduino IDE для операционных систем Windows, Mac OS X и Linux. Для того, чтобы составить представление, как выглядит Arduino IDE, ниже были приведены рисунки 15-18.

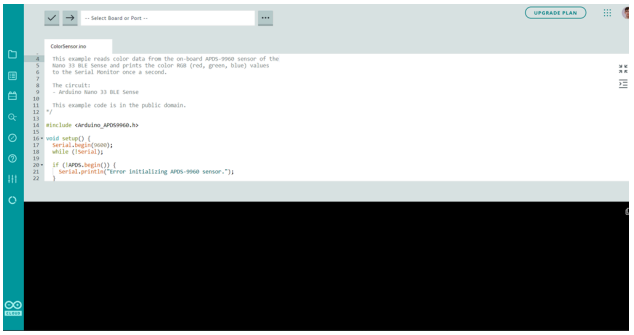


Рисунок 15 – Редактор кода и терминал

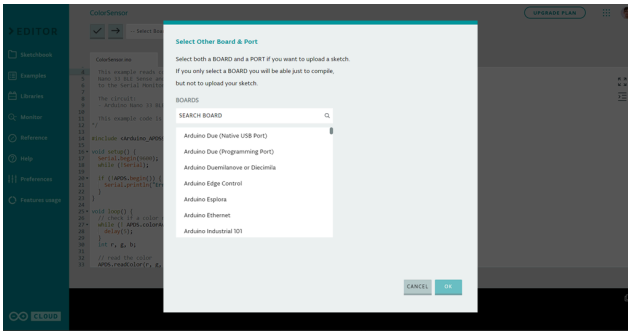


Рисунок 16 – Выбор Arduino платформы

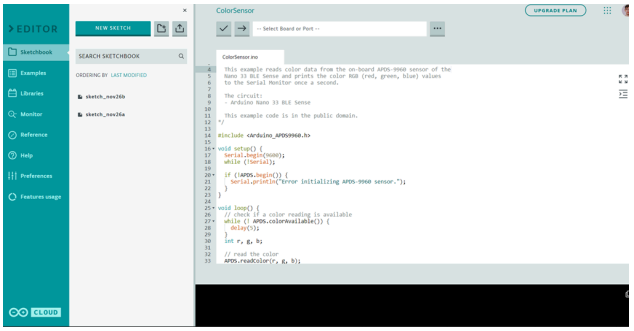


Рисунок 17 – Меню файлов проекта

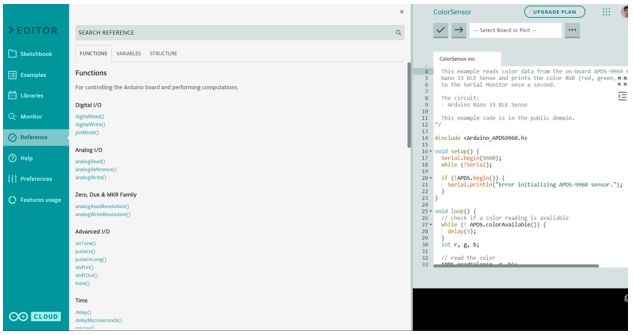


Рисунок 18 – Документация к библиотекам

После анализа были выявлены следующие особенности:

- Для выгрузки программы на устройство для исполнения и контроля версий драйверов, использует легковесный фоново исполняемый Arduino Create Agent;
- Есть возможность конструировать графическую модель исполняющего устройства;
- Поддержка несколько десятков плат (не только Arduino);
- Удобный раздел reference, где лежит документация по разным стандартным библиотекам;
- Исключительно для плат, что позволяет сконцентрироваться на важном.

Arduino web IDE не соответствует полностью всем требованиям IEC 61131, но все равно будет ключевым образцом, так как:

- Этот инструмент знаком многим, кто станет потенциальным специалистом (или уже стал) в области систем управления. Данное IDE изначально имеет

образовательную цель [15]. Обеспечение плавного перехода от обучения к промышленной разработке важно;

- На данный момент микроконтроллеры Arduino в силу своей простоты и дешевизны стали технологическим фактически фундаментом Индустрии 4.0 [16]. Важно принять во внимание историю и эволюцию данного IDE.

1.7. Eclipse Theia

Eclipse Theia - это расширяемая платформа для разработки полноценных многоязычных продуктов типа IDE для облака и настольных компьютеров с использованием передовых веб-технологий [17].

С технической точки зрения, Theia состоит из фронтенда, работающего в браузере или в локальном настольном приложении, и бэкенда, работающего на любом хосте или локально в настольном приложении. Фронтенд и бэкенд обмениваются данными через JSON RPC по веб-сокетам.

Данный продукт имеет подробную сайт-документацию для разработчиков, которые собираются использовать Eclipse Theia для построения собственных IDE. Интерфейс сайта можно видеть на рисунке 21. Предлагается воспользоваться демонстрационным стендом Theia Blueprint для знакомства с IDE. После того, как выполнен переход, пользователю предлагается войти или, если нет учетной записи, зарегистрироваться (рисунок 20). Меню проектов (рисунок 21) и магазин дополнений (рисунок 22) очень похожи на то, что есть в VS Code. Тоже можно сказать и про окно проекта, которое представлено на рисунке 23.

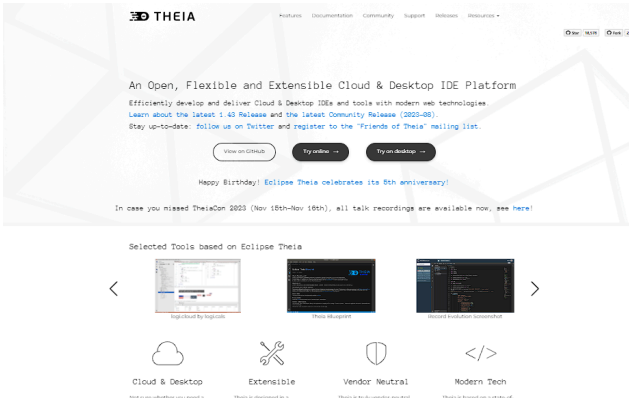


Рисунок 19 – Главная страница

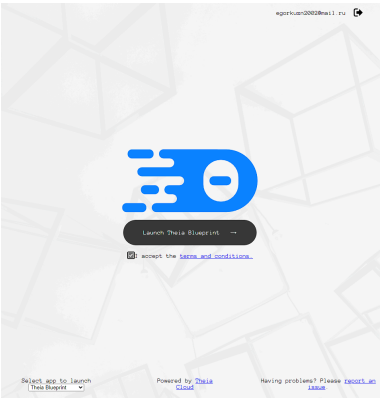


Рисунок 20 – Окно входа и регистрации

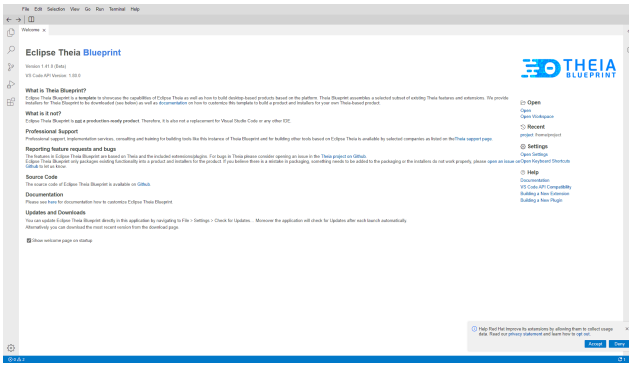


Рисунок 21 – Меню проектов

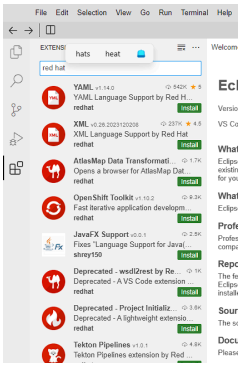


Рисунок 22 – Магазин дополнений

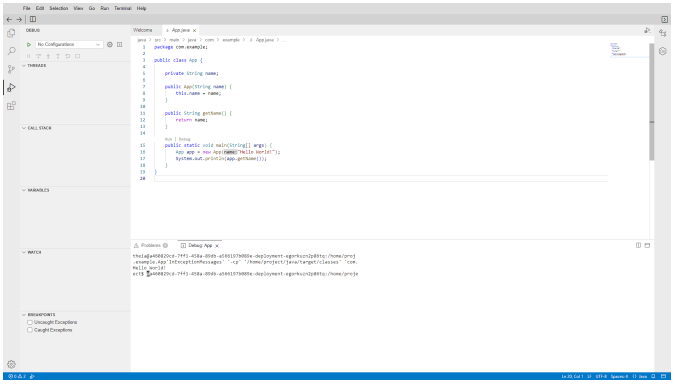


Рисунок 23 – Окно проекта

Приложение Theia состоит из ряда расширений, которые могут вносить свой вклад в фронтенд и бэкенд. Все в Theia построено с использованием этой системы расширений. Важно отметить, что есть возможность добавлять расширения из VS Code, которых на данный момент довольно большое и количество. Это существенный плюс в расширяемости IDE, которая будет основываться на Theia. Использование TypeScript фреймворка Electron позволяет производить сборку как Web версии приложения, так и Desktop.

Бесплатная свободно распространяемая с полностью открытым исходным кодом облачная среда разработки имеет крупную поддержку как со стороны IT компаний-гигантов, так и своего сообщества [18].

1.8. Tinkercad

Международный стандарт IEC 61131 включает в себя Ladder Diagram (Релейно-Контактные Схемы). Поэтому если ранее 1.1-1.7 пункты были с акцентом на то, как

сделать удобным представление и написание кода в облачном IDE с элементом групповой разработки, здесь уделим внимание целиком и полностью схемам.

Проектирование схем в Thinkercad крайне удобно [20]. Для этого нужно создать проект цепи и начать конструирование из имеющихся в меню компонентов. Создание нового или открытие существующего проектов представлено на рисунке 24. Само окно проекта (рисунок 25) состоит из области, где размещаются элементы схемы, и панели выбора компонентов схемы, которая находится в правой части окна. Компоненты в меню поделены на группы для простоты поиска нужного. К сожалению, пока нельзя добавлять свои компоненты и система ограничивает нас имеющимися.

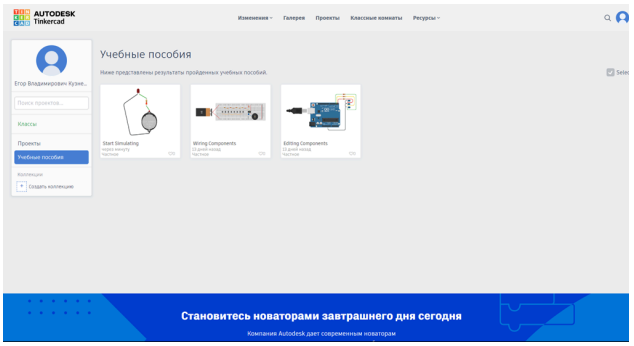


Рисунок 24 – Главная, выбор проекта

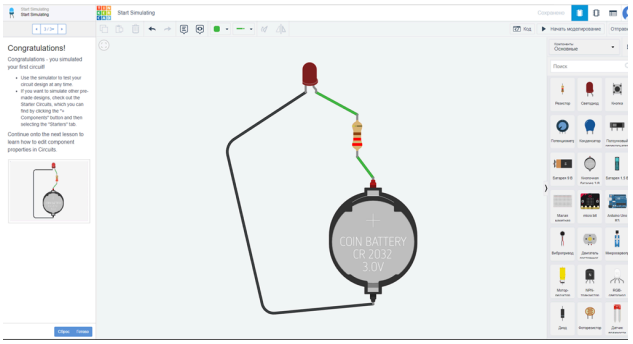


Рисунок 25 – Окно проекта

После того, как схема создана, можно запустить процесс моделирования нажатием на кнопку run, знакомую пользователю из других IDE. Возможно представление схемы в виде чертежа (скачивание в pdf формате, рисунок 26) и таблицы компонентов (выгрузка в csv файл, рисунок 27).

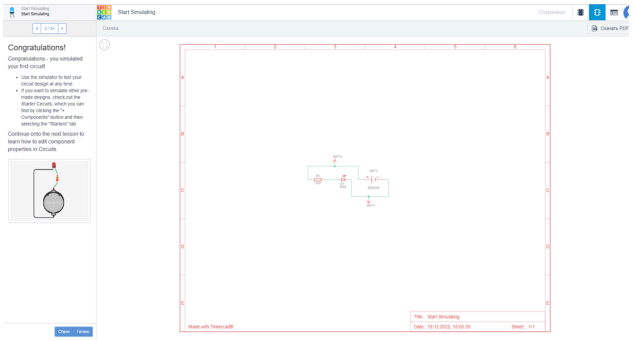


Рисунок 26 – Окно чертежа схемы

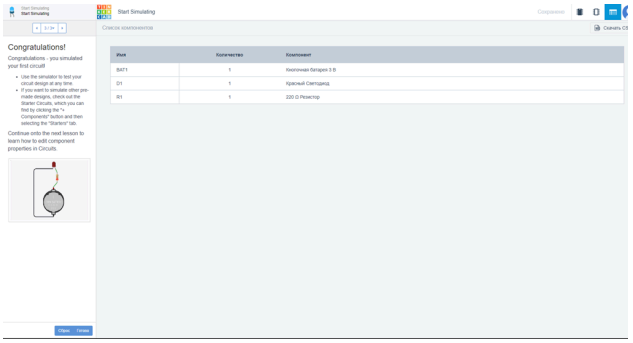


Рисунок 27 – Список компонентов схемы

К совместному проектированию схем можно пригласить других пользователей Tinkercad: приглашение производится по бессрочной ссылке и редактировать схему может каждый. Чтобы проще понять концепцию проекта, можно сравнить Tinkercad с Miro и прочими другими досками для совместной работы над каким-то виртуальным пространством. Касаемо программируемых компонентов, то здесь предусмотрено написание кода под платформы Arduino и in:Micro. Также есть возможность это все смоделировать. Т.к. проект образовательный, то есть интеграция с google classroom [21].

1.9. Eclipse Che

Данная система запускается в Kubernetes, поэтому имеется интеграция с Prometheus и Grafana [22]. У пользователя есть возможность попробовать для изучения Eclipse Che 30 дней бесплатно внутри Open Shift от Red Hat [23]. Также возможно использование в практически любом популярном поставщике облачных услуг, но если устанавливать самостоятельно, то только через Minikube. Eclipse Che предлагает воспользоваться IntelliJ IDEA, VS Code, Eclipse Theia для разработки проекта [11]. Т.к. данные среды разработки были описаны выше, то Eclipse Che проект не стал пересоздавать нужные и привычные пользователям инструменты, добавил возможность использования данных сред для облачной разработки.

Основные задачи, которые решает Eclipse Che:

- Ускорение адаптации сотрудников;
- Устранения несогласованности между средами разработки;
- Обеспечение встроенной безопасности и готовность для использования в компании.

При том, что был упомянут Red Hat, данная разработка относится к сообществу Eclipse Foundation, имеет мощную поддержку от open source сообщества GitHub, а также крупных корпораций. Важно отметить, что Eclipse Che очень хорошо подходила для решения наших задач.

Как было отмечено в работе [24], Eclipse Che предоставляет доступ к терминалу, однако мы не можем контролировать терминальную сессию так, как если бы запускали образ Theia в контейнере от пользователя контейнера правами, и защищал систему, исполняя приложение в виртуальном образе. Кроме того в этой работе отмечено, что Eclipse Che достаточно требовательна к ресурсам, именно поэтому было принято решение разрабатывать RIDE IDE.

Интерфейс системы опишем исходя из того, что предоставляет Red Hat внутри своей системы Open Shift. Окно настроек помогает разработчику для подготовки проекта:

настройка базы данных, пайплайна, брокеров сообщений и многие другие вещи, представленные на рисунке 28.

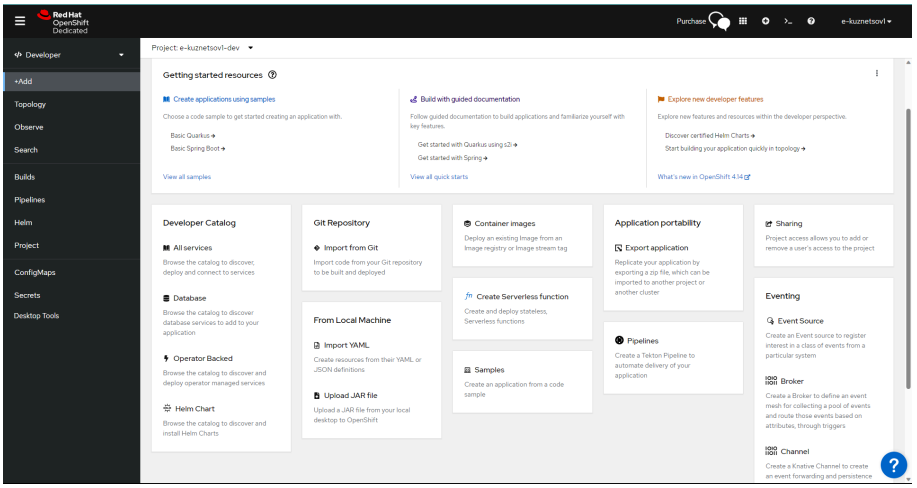


Рисунок 28 – Окно настроек проекта

После конфигурации окружения можно перейти к созданию самого сервиса. Например, здесь (рисунок 29) можно выбрать Spring Boot приложение. Если оно существует, то достаточно указать URL путь до его Git репозитория. Последним полем является имя проекта, после заполнения которого можно переходить к завершению создания проекта.

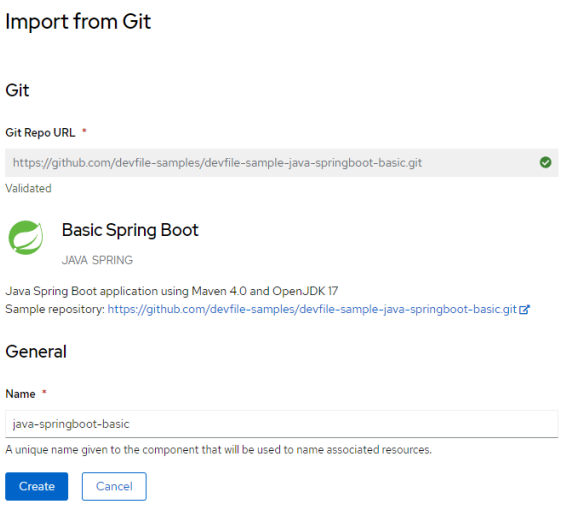


Рисунок 29 – Создание проекта

Происходящее далее открытие проекта зависит от выбранного на предыдущих этапах предпочитаемое IDE: IntelliJ IDEA или VS Code. Из того, что наиболее интересно для текущего объекта исследования: права доступа участников проекта. Здесь (рисунок 30) вводятся определения “Субъект”, “Имя” и “Роль”. “Субъект” – группа или пользователь, для которого применяется определение прав, “Имя” – название субъекта и “Роль” - роль, которая

под собой подразумевает определенный набор правил для конкретного субъекта. Из доступных ролей имеем администратора, редактора и читателя.

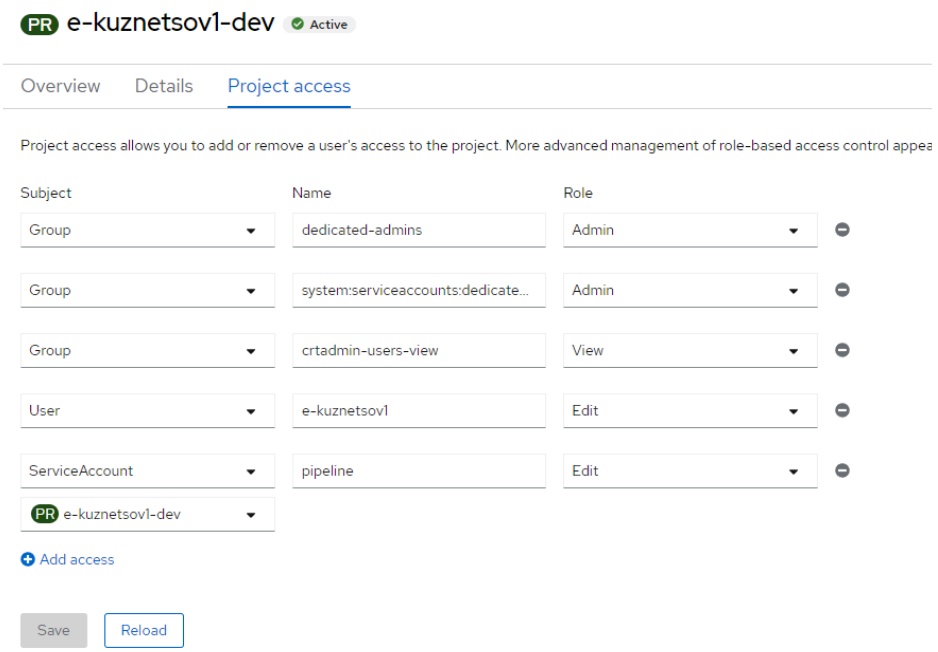


Рисунок 30 – Настройка прав доступа

Механизм предоставления доступа новым пользователям проекта представлен на рисунке 31. Для тех пользователей, которых ранее не существовало в проекте, администратором создается приглашение по почте.

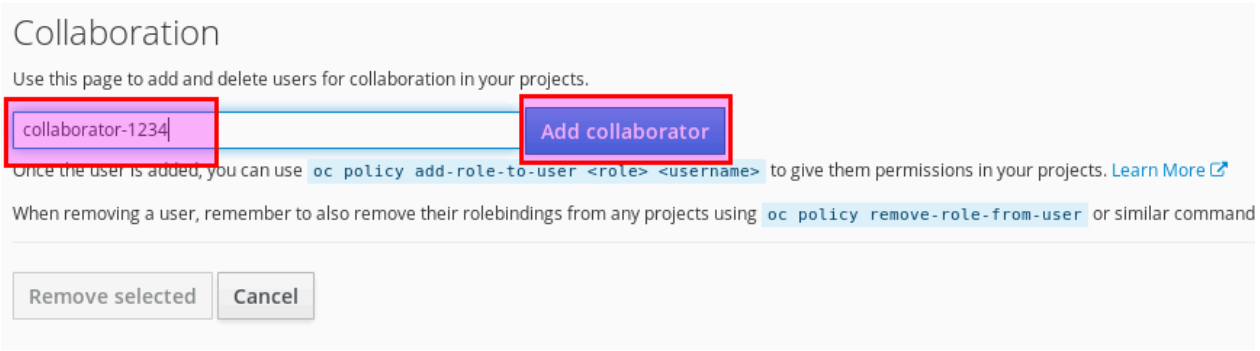


Рисунок 31 – Добавление нового участника проекта

1.10. AWS Cloud9

Компания Amazon - мировой лидер в области облачных услуг [25]. Поэтому было решено рассмотреть Cloud9 [26] как возможное лучшее коммерческое решение. Опишем устройство данного IDE.

Выбираем в поиске Amazon Cloud9. Создаем пространство: задаем его имя и приводим описание по необходимости. Затем для пространства выбирается тип среды, тип состояния, платформа, время неактивности (для перехода в режим гибернации), а также дополнительно настраивается сеть. После этого, нажимаем “Next step” для перехода на следующий этап.

Запускается окно проекта (рисунок 32), где в первой вкладке отображается приветствие, в котором содержатся полезные ссылки. В верхнем горизонтальном меню все ровно также, как в Eclipse Theia. В его правом углу расположена иконка участника, кнопка для “Поделиться”, кнопка настроек проекта. В левом блоке: папочная структура проекта с файлами. На панеле справа:

- Colaborate – модуль, отвечающий за совместную разработку, в котором представлены участники, информация о их правах и групповой чат;
- Outline – структуры проекта;
- AWS Resources – используемые сервисы Amazon, подключенные к проекту;
- Debugger – дебаггер.

На рисунке 33 показана открытая вкладка “Colaborate”: в верхней части пользователи с указанными правами и именами. Ниже располагается сам чат. На этом же рисунке в самом низу располагается терминал ОС, которая была выбрана на этапе инициализации системы.

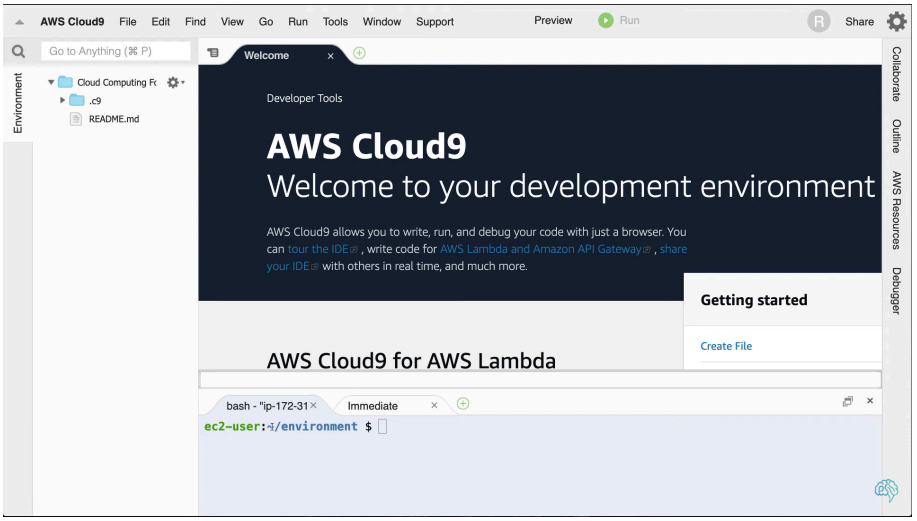


Рисунок 32 – Окно проекта

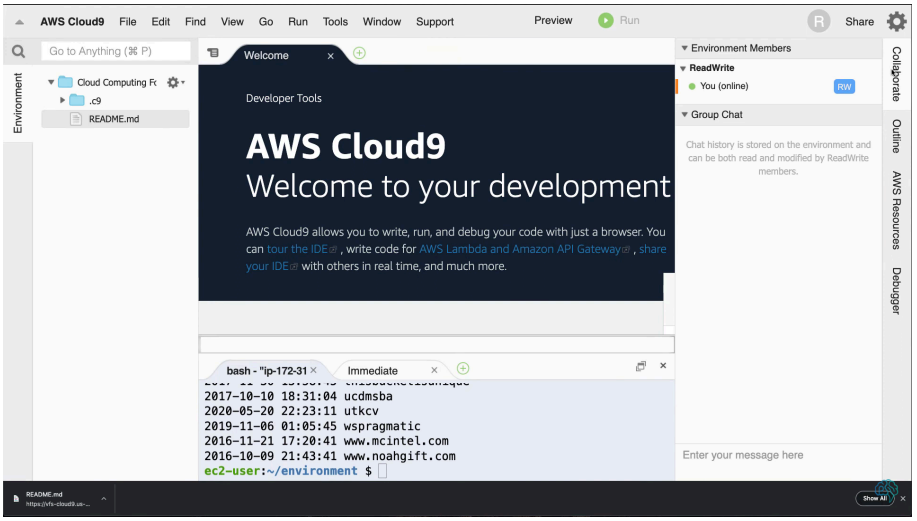


Рисунок 33 – Просмотр активных участников проекта

Механизм предоставления доступа к проекту в Amazon Cloud9 реализован через специальное окно, которое открывается с помощью кнопки “Share” и выглядит как на рисунке 34. Первый блок - ссылка на проект: ссылка на саму среду и приложение, для которого существует возможность разместиться во внешней сети. Второй блок отвечает за вывод информации о пользователях, подключенных к проекту. Имеется возможность внести изменения в права доступа пользователя. Существует специальный флажок, при установлении которого пользователи не будут иметь возможности сохранить состояние вкладки. Третий блок – приглашение IAM (AWS Identity and Access Management) пользователя. В поле необходимо ввести имя пользователя IAM, выбрать для него права на чтение или чтение запись. После чего пользователю поступает сообщение на почту о приглашении. Если IAM нужного пользователя не существует, то необходимо его создать.

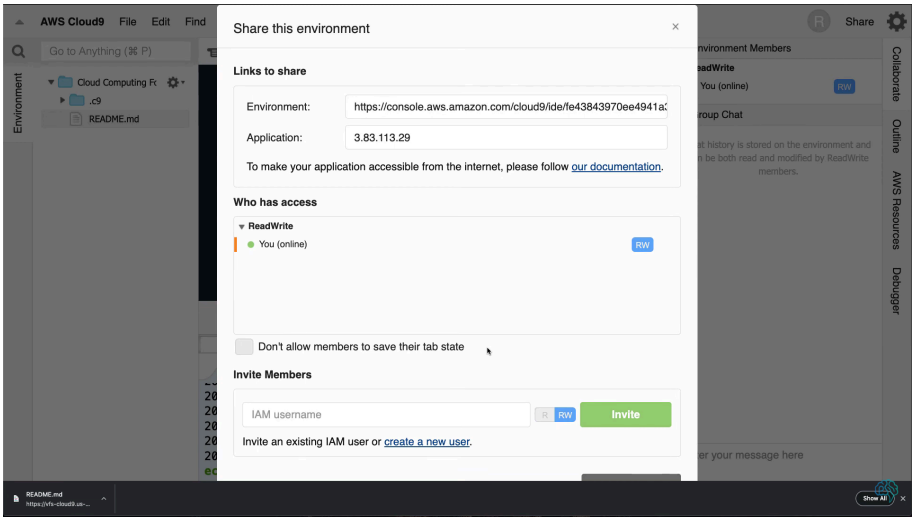


Рисунок 34 – Создание приглашения нового участника

Шаблоны для облачных проектов являются важной вещью из-за того, что позволяют сэкономить время на установке нужных программ и утилит для работы с определенной технологией. Поэтому шаблоны в Cloud9 есть, покрывают большую часть потребностей пользователя (рисунок 35).

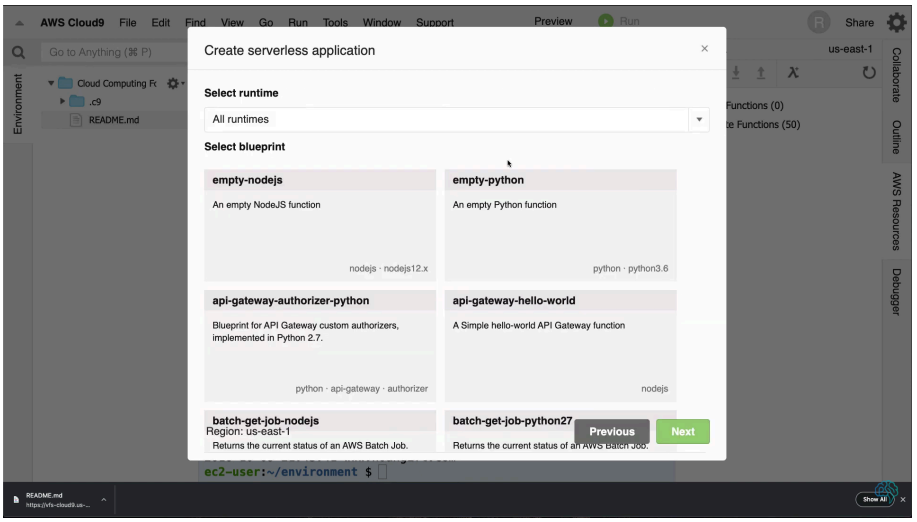


Рисунок 35 – Инициализация проекта шаблоном

Главный минус Amazon Cloud9 – доступно только платное использование, отсутствует возможность оплаты из России, кроме того невозможно использование без предоплаты.

1.11. Построение требований

В течение анализа существующих решений заполнялась таблица А.1 из приложения А. Было решено обобщить некоторые возможности в пункты и указать их отсутствие или наличие в конкретном исследуемом IDE, в последнем столбце привести вывод по каждому из пунктов.

С помощью таблицы из приложения А сформулированы требования. Модуль управления проектом Reflex должен обеспечивать возможность:

- создания проекта Reflex по шаблону;
- интеграции с git репозиторием;
- определения видимости проекта;
- определения условия использования проекта;
- документирования проекта;
- назначения ролей участникам проекта (владелец, администратор, редактор, комментатор, читатель);
- идентификации подключившихся пользователей;

- отображения текущего положения курсора онлайн-пользователей;
- установки области отображения на текущее положение курсора онлайн-пользователя;
- коммуникации между участниками проекта;
- использования сторонних библиотек;
- создания библиотек;
- сохранения копии проекта на локальный или облачный диск пользователя;
- восстановления проекта из сохраненной копии;
- запуска генератора Си файла для платформы ATmega.

2. Архитектура модуля и используемые методы

Для того, чтобы функционировать, модулю необходим доступ к серверу модуля и к приложению Theia. Сервер модуля предполагает собой что-то общее для всех создаваемых проектов. Theia клиент, запускаемый в отдельном Docker контейнере, может хранить в себе только те данные и запускать только те сервисы, которые отвечают за конкретный проект. Сервер модуля должен предоставлять информацию о пользователях, сохранять данные о проекте. Сервис Theia - это клиент сервера модуля, внутри которого встроен пользовательский интерфейс разрабатываемого модуля. Определенные выше сервисы должны обмениваться сообщениями формата JSON. Представить взаимодействие компонентов системы можно в виде схемы на рисунке 36.

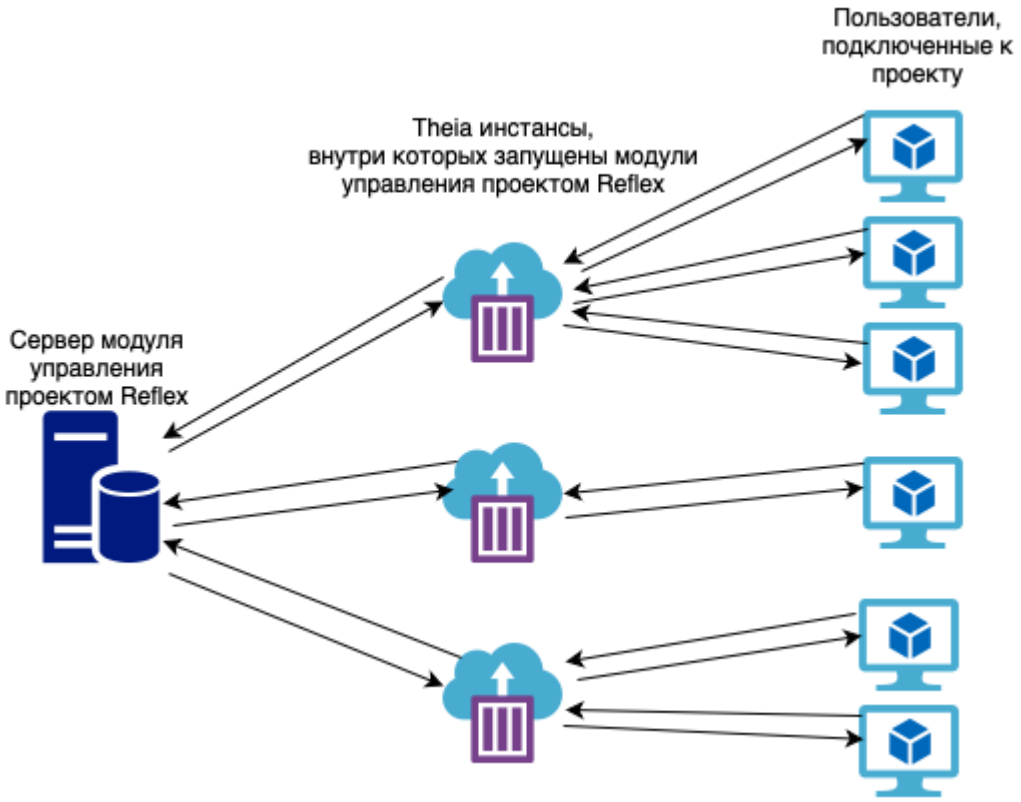


Рисунок 36 – Общая схема взаимодействия

Для выполнения каждого из требований было принято решение написать по приложению. Для части приложений, которые можно вынести на сервер модуля, будет использоваться микросервисная архитектура. Приложения, которые зависимы от конкретного проекта, имеет смысл реализовывать как набор приложений-утилит запускаемых внутри контейнера с Theia. Контейнер с Theia – контейнер Ubuntu версии 24, в которой запущено

приложение Theia. Каждое из реализуемых приложений объединяет графический интерфейс внутри Theia приложения.

Все описанное выше представляется в виде рисунка 37. Сервисы (утилиты и микросервисы) описываются ниже в пунктах 2.1-2.2. В главах объясняются причины реализации каждого из требований через утилиту или через микросервис.

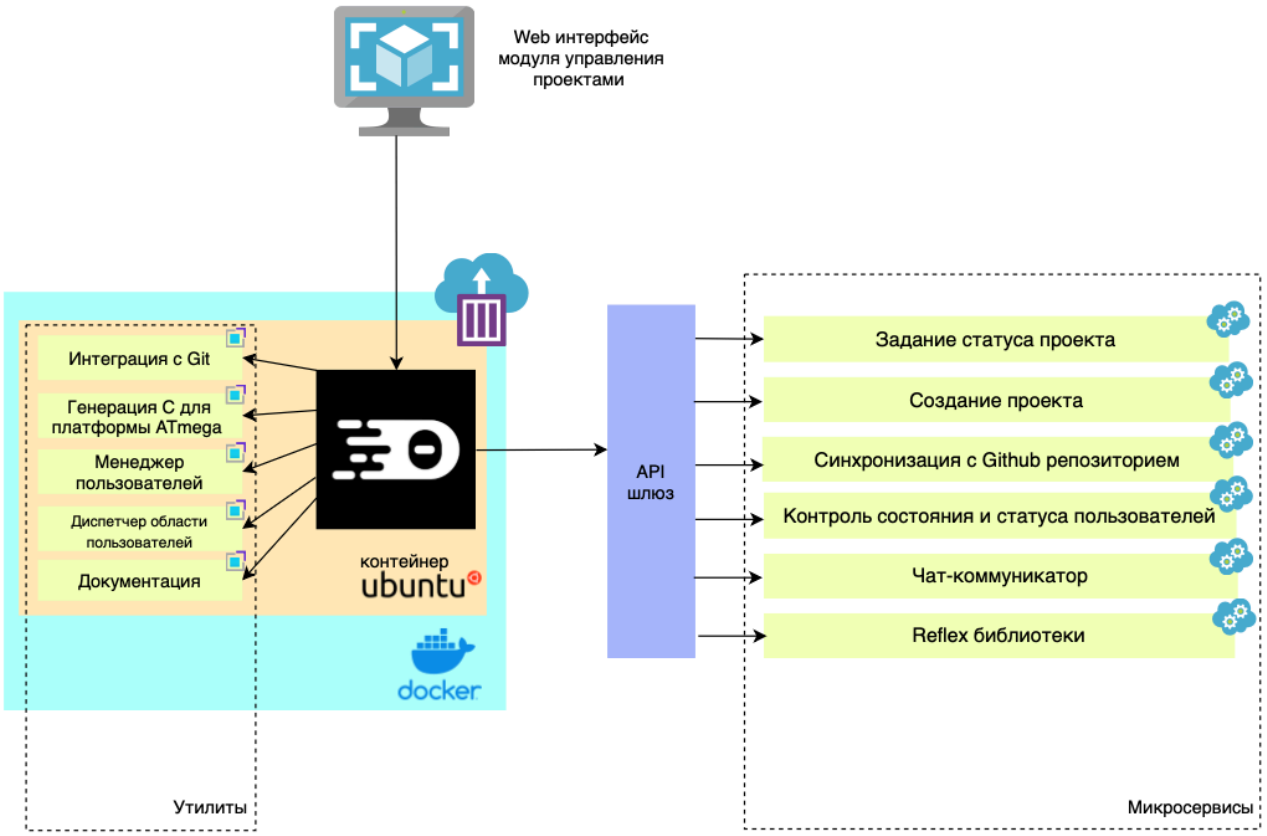


Рисунок 37 – Схема сервисов модуля

2.1. Сервер модуля

Перечислим микросервисы сервера модуля управления проектами облачного IDE языка Reflex:

- Задание статуса проекта;
- Создание проекта;
- Синхронизация с Github репозиторием;
- Контроль состояния и статуса пользователей;
- Чат-коммуникатор;
- Reflex библиотеки.

Микросервис “Задание статуса проекта” для получения информации о статусе проекта. Т.к. у RIDE имеется отдельный клиент, то при поиске проекта приватные, если не доступны пользователю, скрываются, а публичные видны всем. При создании проекта устанавливается его статус, который сохранится на сервере. Theia должна предоставлять возможность выбора статуса проекта, изменения списка людей, которым доступен проект, если он приватный. Данные о изменениях в настройках отправляются на сервер.

Микросервис “Создание проекта” для создания проекта по шаблону предполагает хранение на RIDE данных о логике для построения структуры проекта по шаблону. Т.к. она общая, то соответственно должна хранится на RIDE. Клиент Theia обращается к серверу RIDE для того, чтобы предоставить пользователю возможные варианты автогенерации. После того, как пользователь сделал выбор, информация о выборе и необходимые для генерации данные о проекте поступают на сервер RIDE, который выполняет логику генератора шаблона. Клиент Theia получает архив папки шаблона проекта, распаковывает и пользователь получает доступ к своему проекту внутри редактора кода.

Микросервис “Синхронизация с Github репозиторием” для резервного копирования кода проекта, сохранения кода в хранилище пользователя, выгрузка из хранилища в RIDE.

Микросервис “Контроль состояния и статуса пользователей” должен предоставлять клиенту информацию о пользователях системы RIDE, которую могут использовать другие приложения. Например, приложение, которое будет задавать роли участникам проекта.

Микросервис “Чат-коммуникатор” для взаимодействия участников проекта. Хранит в себе информацию о группах и сообщениях. Должен предоставлять методы создания группы, отправки сообщения в группе, ее изменения и удаления. В клиенте Theia должен быть реализован графический интерфейс, соответствующий классическому мессенджеру.

Микросервис “Reflex библиотеки” для хранение библиотек для Reflex. Предоставляет API для поиска подходящей библиотеки, добавление новой.

2.2. Утилиты модуля

Здесь будут перечислены приложения, которые не были определены на сервер по ряду причин. Сначала перечислим, а далее опишем каждое:

- Интеграция с Git;
- Генерация C для платформы ATmega;
- Менеджер пользователей;
- Диспетчер области пользователей;

- Документация.

Утилита “Интеграция с Git” должна реализовать требования, но для этого нужен сам git, который проще всего установить в контейнер. Папка с проектом будет уже локальным git репозиторием. Во-первых, это будет удобно участникам проекта, кто пользуется git из консоли, во-вторых, команда может установить любой графический git клиент, доступный из Open VSX Registry [27] - магазина дополнений для Eclipse Theia.

Утилита “Генерация C для платформы ATmega” включает в себя генератор из Reflex в Си. Так как он уже разработан командой языка Reflex, его стоит связать с графическим интерфейсом разрабатываемого модуля.

Утилита “Менеджер пользователей” определяет роли участников проекта. Ubuntu - многопользовательская ОС, и она обладает большим количеством инструментов для совместного использования. Каждой роли соответствуют определённые права. Кроме того, возможность запуска Theia процесса от пользователя A дает гарантии того, что пользователь запустив терминал или что-то внутри Theia проекта, то будут обеспечиваться гарантии безопасности на уровне ОС.

Утилита “Диспетчер области пользователей” собирает информацию о действиях участника внутри Theia проекта. Это нужно, чтобы отображать курсоры, нажатия и другие действия, чтобы члены команды были в курсе, кто что делает.

Утилита “Документация” прежде всего предполагает добавление возможности читать и создавать Markdown тексты, которые часто применяют для ведения документации. Крайне важно учитывать и тот факт, что кроме текстовых инструкций, необходимы чертежи и схемы устройств.

3. Проектирование интерфейса

IDE языка Reflex использует Theia, как основу для разработки своего решения. Поэтому нужно учитывать возможности предоставляемой платформы: все новые возможности реализуются в виде дополнений.

Дополнения Theia обычно располагаются в левой части экрана на специальной вертикальной панели. Выбрать дополнение можно нажав на него. Для того, чтобы не перекрывать область редактора кода, принято создавать интерфейсы вытянутыми вверх. Поэтому были получены следующие прототипы дизайна окна модуля управления проектом (рисунки 38 - 43).

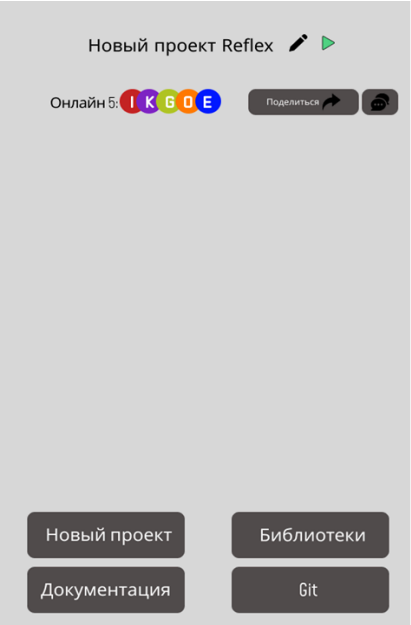


Рисунок 38 – Главный экран проекта

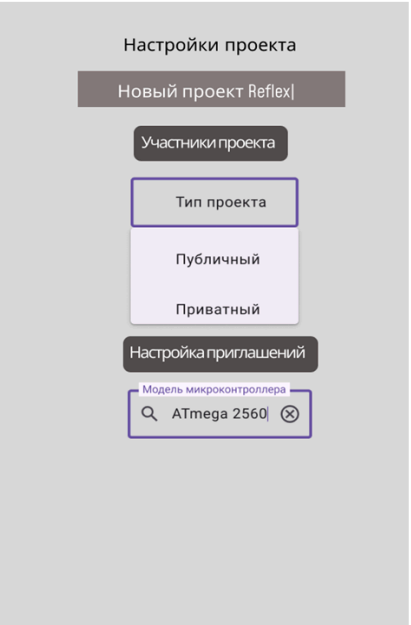


Рисунок 39 – Экран настроек

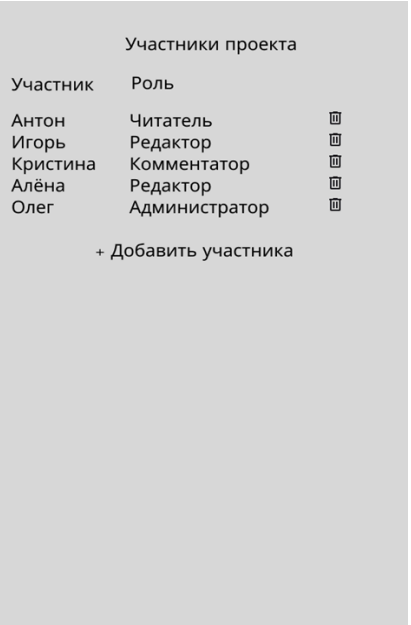
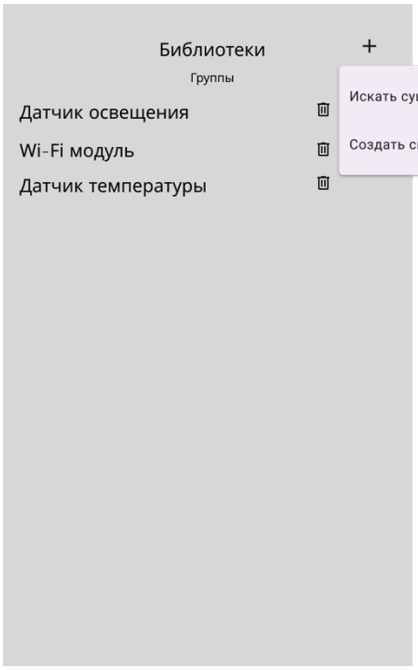
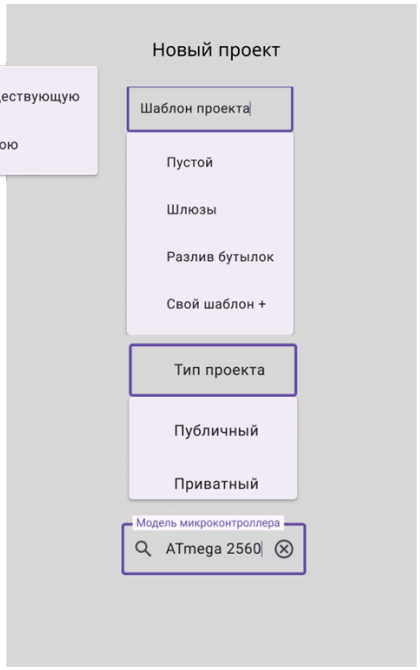


Рисунок 40 – Управление участниками проекта





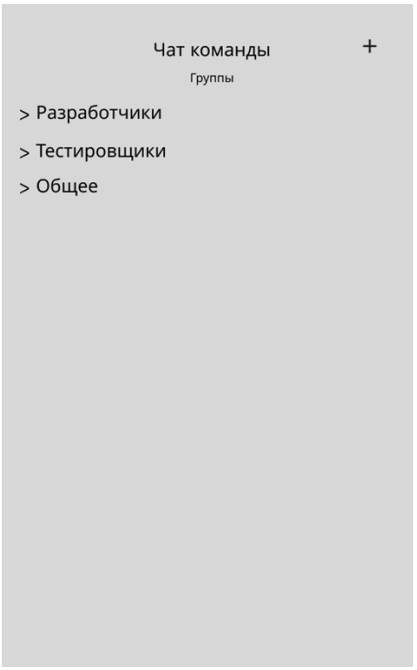


Рисунок 41 – Экран библиотек Рисунок 42 – Создание проекта Рисунок 43 – Экран чата проекта

Кроме этого, в требованиях было определено предоставление возможности идентифицировать подключившегося пользователя, а также отображение положения и действия курсора мыши участника проекта. Выглядеть это должно как на рисунке 44.

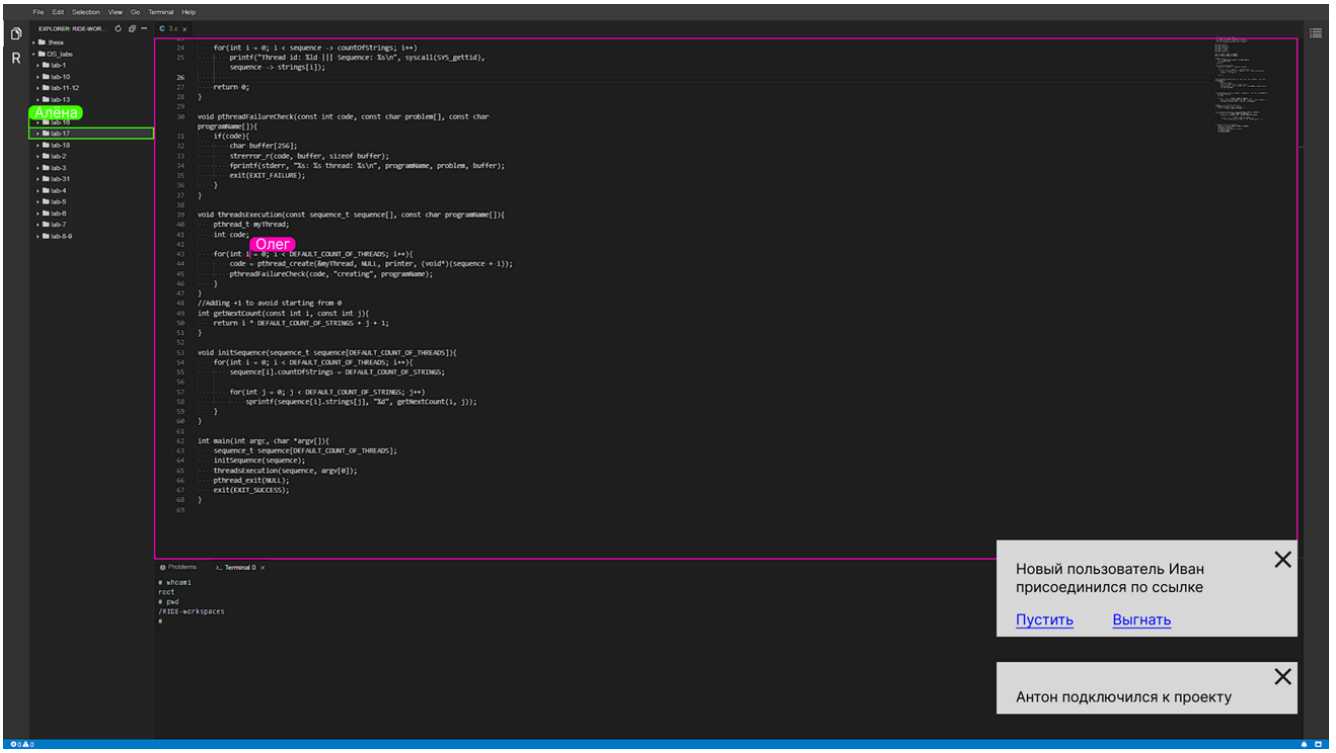


Рисунок 44 – Получение сообщений о присоединившихся пользователях, а также пример визуализации курсоров

4. Реализация

Команда разработки проекта RIDE уже столкнулась с тем, что используемая версия Theia 0.16.0 сильно отстала от текущей (1.48.0 [28]). Было несколько попыток миграции на новую версию Theia, однако ни одна не завершилась. Основная причина заключалась в условии: ранее реализованные модули должны запускаться. Для выполнения такого условия необходимо переписать каждый из модулей. Простое обновление версии невозможно, так как каждая из реализаций завязана на определенных библиотеках, поставляемых вместе с внутренним фреймворком от Theia. И в более новых версиях Theia от библиотеки, которую активно использовал один из модулей, могут отказаться. Таким образом, клиентское приложение использует node 10.24.1 LTS(Long-Term Support), TypeScript, Theia 0.16.0, React, встроенный в зависимость @theia/core, версии 16.0.x и yarn для сборки проекта версии 1.7.0.

Для реализации сервисов сервера был определен следующий список технологий: Kotlin версии 1.9.22 с целевой JVM 21 LTS , Spring Boot 3.2.3, Gradle, описываемый в Kotlin DSL, Nginx и Docker. Крайне важно было выбрать такие версии библиотек, чтобы упростить дальнейшую поддержку. Выбор в пользу Kotlin, а не Java, был сделан, чтобы сэкономить время на написании кода и исключить взаимодействие с Lombok, снизить риски связанные с mutable переменными, упростить проверку на null за счет встроенного механизма Null Safety. Не смотря на растущую популярность проекта Ktor [29], Spring Boot выбран для более простой адаптации студентов, приходящих в проект. Spring Boot предоставляет возможность использования Spring Data JPA, который в текущем проекте применен для взаимодействия с базой данных PostgreSQL. Nginx выполняет роль API шлюза, который упоминался в пункте 2.

4.1. Интеграция с Git

Реализация началась с решения проблемы работы с Git при совместном использовании кода. До Git рассматривался вариант использования CVS, в виду следующих причин:

- Разработчики работают в одной IDE, в одном Docker контейнере над одним репозиторием - централизация, когда Git является распределенной системой контроля версий;
- Необходимость в представлении демонстрации совместного редактирования файла. То есть механизм check-in/check-out;

- И concurrent, что отвечает требованию, про совместную разработку и одновременную модификацию файлов проекта участниками.

Однако, CVS имеет ряд недостатков, что стало причиной развития проекта Git [30]. Кроме этого, интеграцию с Git предусматривают сервисы, которые Поэтому самым простым и удобным способом будет клонировать каждому участнику проекта, имеющему доступ на чтение или изменение. На клонированный репозиторий накладываем ограничение, которое возможность модификации будет предоставлять только владельцу. Это необходимо сделать ровно для того, чтобы избежать проблем к совместному доступу к файлам. Таким образом каждый участник команды может посмотреть, что делает другой участник команды и не помешать его работе над проектом. Таким образом получен bash-скрипт, описанный в Листинге 1.

```
su $USERNAME
cd
git clone $SERVER_URL/$REPO_NAME
chmode -R 744 $REPO_NAME
exit
```

Листинг 1 – Создание локального Git репозитория пользователя с правами модификации для владельца и чтения, для прочих пользователей проекта

4.2. Генерация C для платформы ATmega

Транслятор из языка Reflex в C-код, уже реализован и интегрирован в RIDE. IDE сканирует в проекте .rcs файлы. Если они не содержат ошибок, то для них генерируется C код в месте CMakeFile.txt в директории src-gen.

Однако в Docker контейнере с Ubuntu, в котором запускается Theia, не была установлена утилита cmake. В результате чего, в Dockerfile была добавлена установка cmake. Кнопка старта/запуска, имеющая форму зелёного треугольника, реализует операцию сборки и запуска (листинг 2).

```
cmake ~/REPO_NAME/src-gen/c-code
cmake --build ~/REPO_NAME/src-gen/c-code
```

Листинг 2 – сборка и запуск CMake проекта

Также реализовано формирование CMakeFile.txt исходя из того, какая модель микроконтроллера ATmega выбрана. Предыдущий CMakeFile.txt был дополнен кодом Николаса Гоя, в который в своей статье описал сборку с CMake под AVR микроконтроллеры

[31]. Для того, чтобы обновленный скрипт CMake запускался, в контейнер устанавливаются утилиты avr-gcc и avrdude.

В примере Гоя явно указана модель платы и ее основные параметры конфигурации: частота чипа, скорость передачи данных по умолчанию для UART, используемый программатор и fuse-биты. В нашем случае для упрощения задачи было принято решение взять 3 платы, для которых уже заданы параметры по умолчанию. На текущий момент данные значений по умолчанию хранятся в структуре Map приложения. При выборе из выпадающего списка плат на стороне frontend, происходят изменения в переменных среды системы. Далее, в CMake с помощью $ENV.{} достает нужные параметры для конфигурации.

Справедливо заметить, что текущее решение не рассматривает все устройства ATmega, которые можно запрограммировать с помощью avrdude [32], кроме того оно не подразумевает добавление новых. Поэтому следующие задачи связанные с “C-генератор” будут связаны с написанием сервиса, который будет хранить информацию или собирать информацию о разных платах и их параметрах.

4.3. Документация

Был определён Markdown, как основной формат файлов документации. Для того, чтобы представлять файлы документации, используется open-source библиотека markdown-js [33]. Данная библиотека оказалась совместимой с версией node, используемой в проекте. Документация проекта должна храниться в специальной директории с названием dock и главном файле README.md. Таким образом, пользователь сам определяет структуру документов, сам пишет файлы в удобном для него формате, когда модуль представляет в *.md файлы в финальном виде.

4.4. Менеджер пользователей

Экран “Настроек проекта”, которые доступен только администратору, содержит кнопку “Участники проекта”, ведущую к соответствующему экрану. На этом экране выводится весь список пользователей, имеющих доступ к проекту. Пользователя можно исключить, нажав кнопку, которая находятся напротив каждого справа. Кнопка “+”, для добавления нового пользователя, создает форму с пустыми полями “Участник”, “Роль”.

Поле “Участник” может содержать только латинские буквы, цифры или одиночные дефисы и не может начинаться или заканчиваться дефисом. Не может превышать 39

символов. Это отвечает требованиям к имени пользователя в GitHub. В процессе ввода предлагаются до 5 пользователей: аватар, имя, фамилия и имя GitHub.

Поле “Роль” ограничено вариантами из выпадающего списка “редактор”, “комментатор” и “читатель”.

“.viewignore” и “.editignore” в интерфейсе модуля - ссылки, которые открывают соответствующие файлы. “.viewignore” - набор файлов и директорий недоступных для чтения читателям и комментаторам, “.editignore” - набор файлов недоступных для модификации редакторами. Файлы “.viewignore” и “.editignore” используют паттерны как в .gitignore [34].

Пользователь с ролью “читатель” не может никаким образом изменить структуру проекта: добавить директории, изменить файлы. Ему доступно только чтение файлов и директорий не из списка “.viewignore”.

Пользователь с ролью “комментатор” является наследником роли “читатель”. Комментатору дополнительно позволено оставлять пометки-комментарии к строкам кода.

Пользователь с ролью “редактор” имеет доступ на модификацию файлов и директорий не из списка “.editignore”. На пользователя с ролью “редактор” не накладывается ограничение на чтение файлов и директорий, как на пользователя с ролью “читатель”.

Пользователь с ролью “администратор” имеет root в Linux. Такая роль назначается создателю текущего проекта. Для передачи прав доступа администратору доступна кнопка “Передать права администратора другому пользователю”. После нажатия на нее появляется предупреждение, чтобы пользователь обратил внимание на важность выбора. При подтверждении намерений по передаче права администратора, появляется список участников текущего проекта. Из них администратор выбирает одного. После чего снова появляется предупреждение, которое предлагает отменить передачу прав, в случае которой произойдет возвращение к выбору кандидата. В случае согласия со своим выбором, у пользователя переоткроется окно Theia, запущенное от созданного ранее для него пользователя. А у того, кто получает роль администратора, соответственно приложения Theia будет запущено от root. Вышеописанное для лучшего понимания процесса передачи прав, представлено в виде последовательности действий на рисунках 45-48.

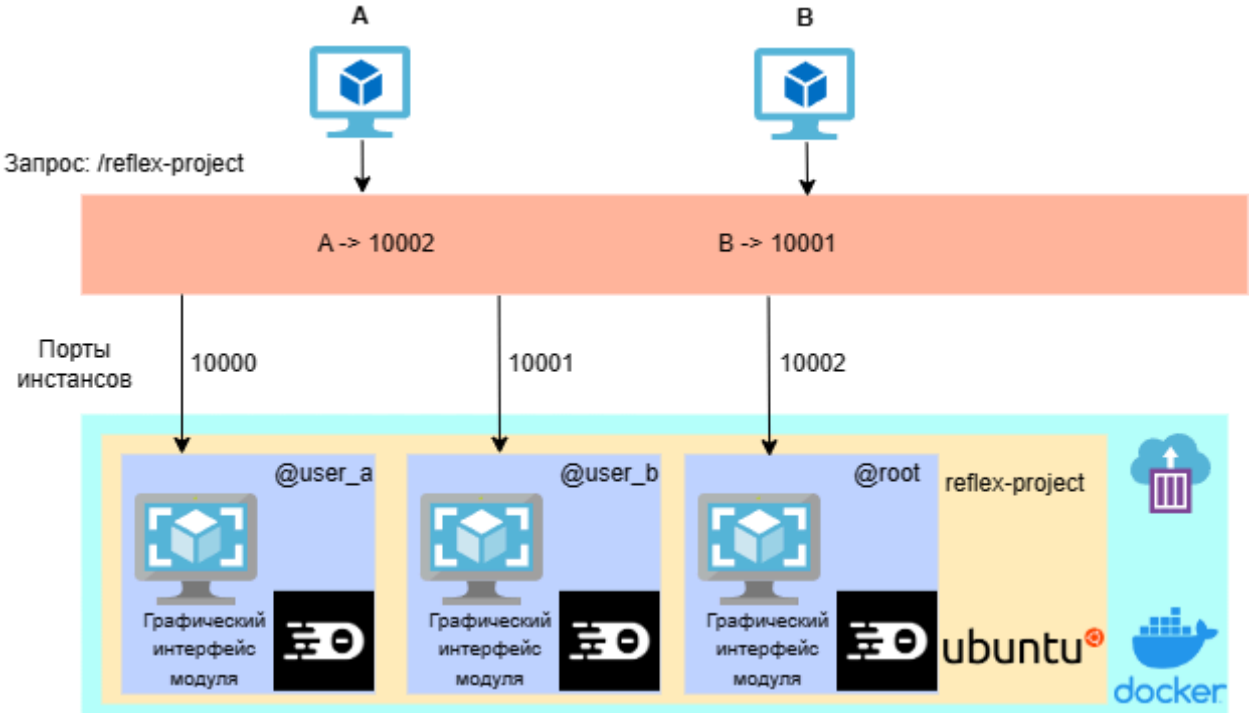


Рисунок 45 – Начальное состояние до применения изменений

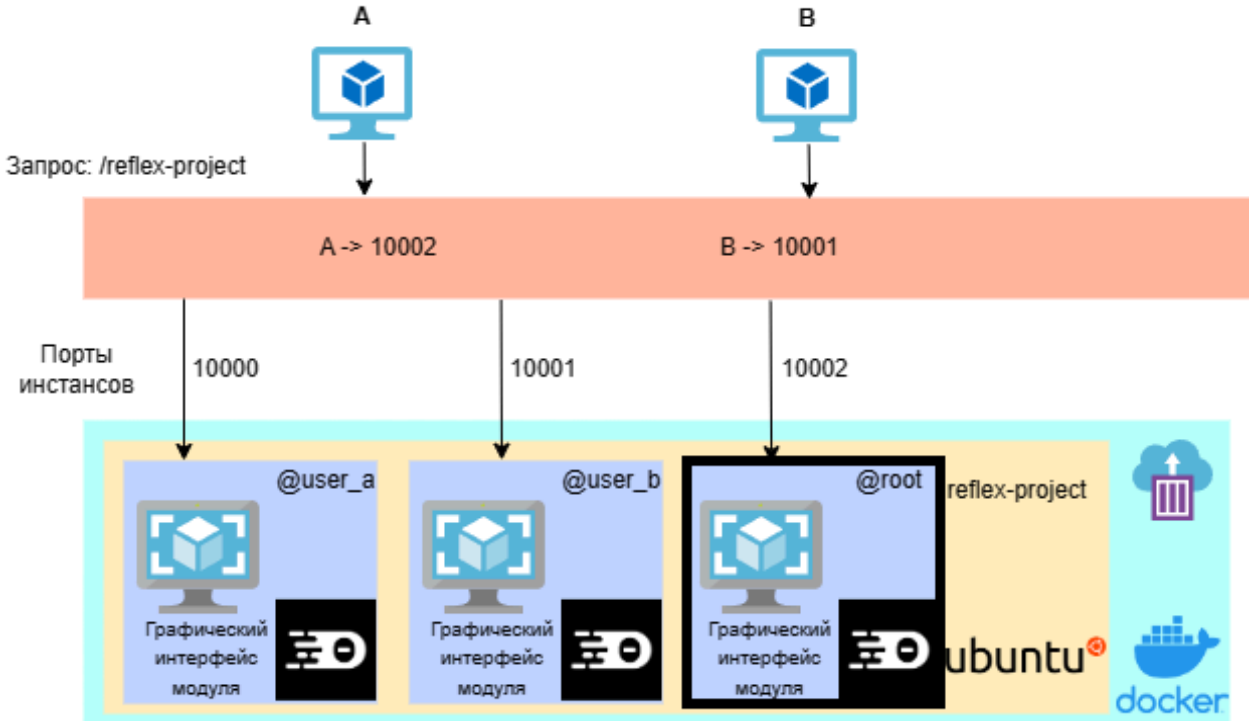


Рисунок 46 – Администратор применяет изменения: теперь администратором становится B

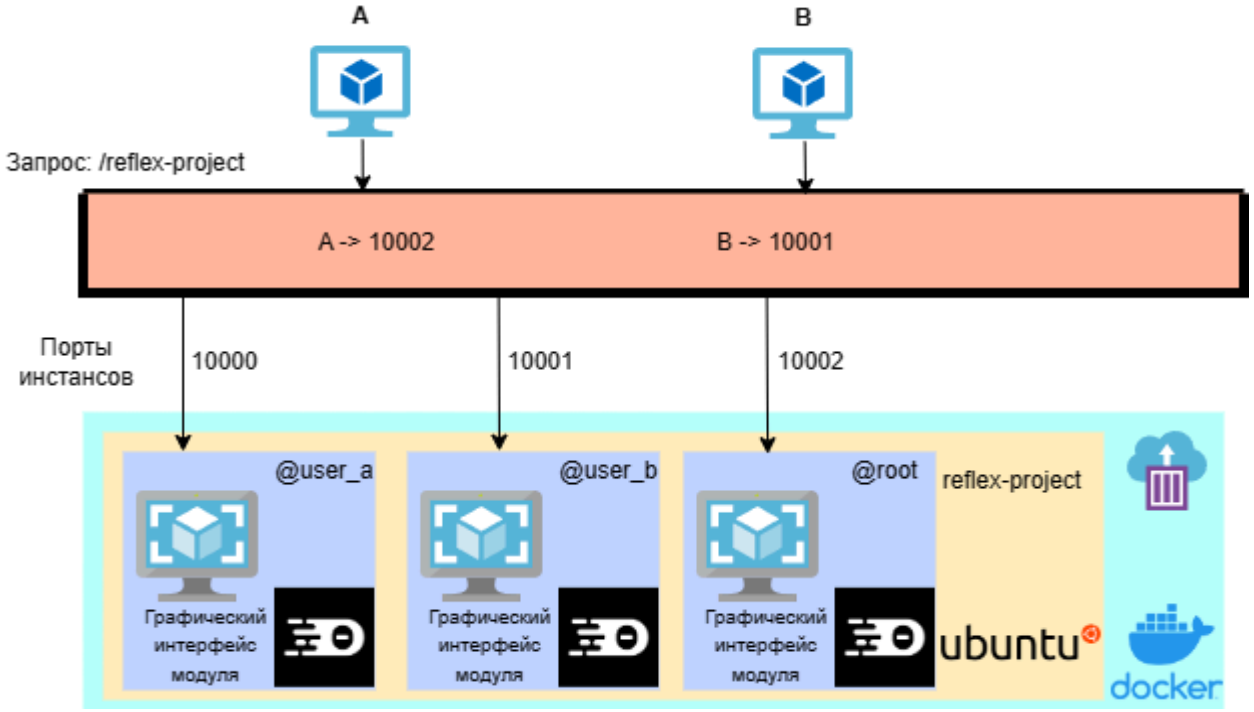


Рисунок 47 – Информация об обновленной конфигурации сообщается сервису авторизации и аутентификации

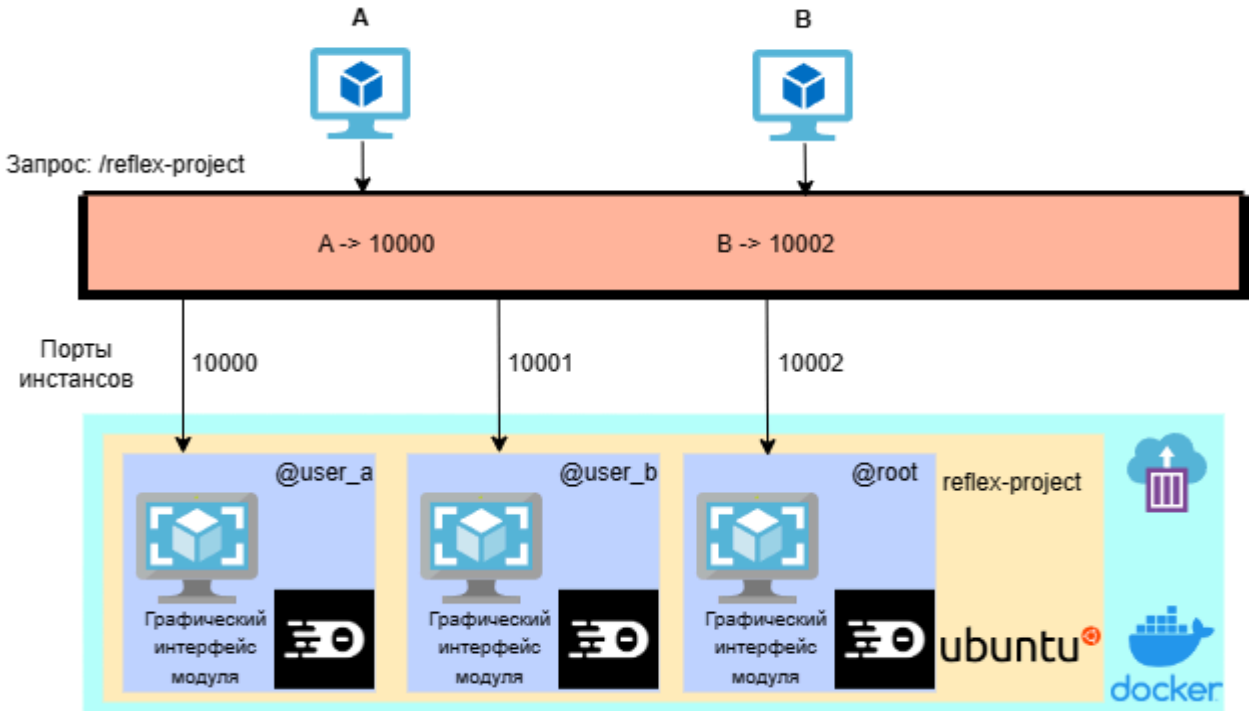


Рисунок 48 – Сервис авторизации и аутентификации обновляет данные в своей таблице соотнесения пользователей к портам для проекта reflex-project

То, что находится между самим контейнером и пользователем, представляет собой совокупность сервиса авторизации и аутентификации пользователей и API-gateway. Получая

запрос от пользователя, сервис получает информацию о проекте, и уже для конкретного пользователя передается экран необходимого инстанса Theia.

Для установления прав чтения и записи написан скрипт на языке bash. Скрипт применяется автоматически при модификации “.viewignore” или “.editignore”.

4.5. Диспетчер области пользователей

Для начала опишем базовые вещи, которые лежат в основе данного приложения. Факт того, что пользователь A наблюдает за B, формируется исходя из того, что пользователем был выбран проект, в домашней директории B. На стороне frontend подключение пользователя к проекту обозначается круглой иконкой случайного цвета, в центре которой первая буква имени пользователя, если пользователь не выбрал аватар (рисунок 49). Если пользователь выбрал аватар (изображение, с которым желает ассоциировать себя пользователь), то иконкой будет изображение аватара, вписанное в окружность. Чтобы узнать имя и фамилию пользователя, можно нажать на иконку и появится информация с именем, фамилией и никнеймом. Для ситуаций, когда пользователей больше 5, предусмотрена возможность просмотра списка пользователя. На рисунке 50, список пользователей разделен на группы “в сети” и “не в сети”.

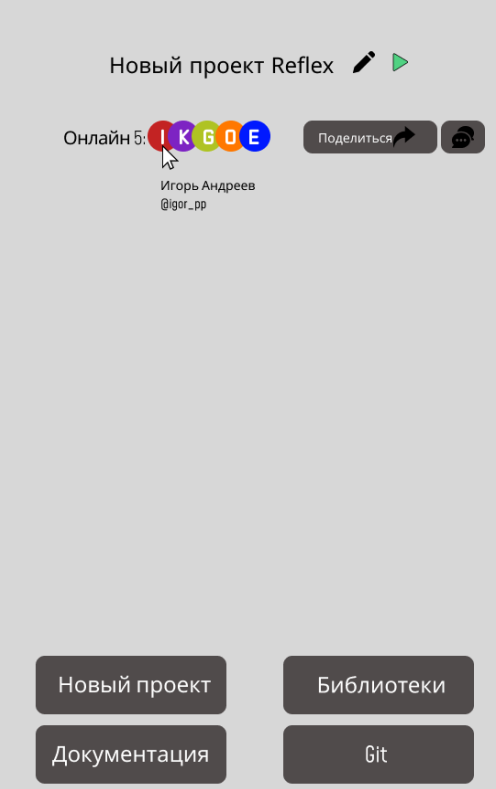


Рисунок 49 – Информация о пользователе, появляющаяся при наведении на него мыши

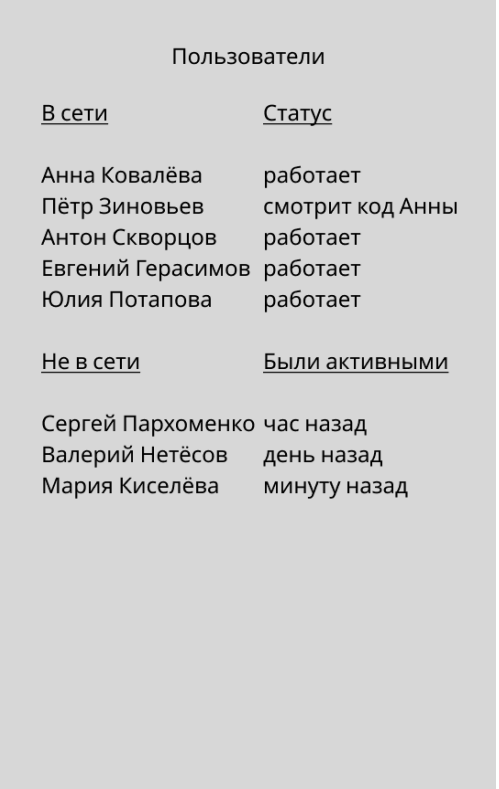


Рисунок 50 – Экран с информацией о всех пользователях проекта

Важным в “Диспетчере области пользователя” является графическое сопровождение перемещения пользователя в проекте. В прототипе это демонстрировался вариант реализации через курсор и выделение границ области просмотра и редактирования. В Theia используется механизм виджетов. Widget - интерфейс, который является базовым для всех элементов графического интерфейса. То, что необходимо для нашей задачи - создание реализации на базе ViewContainer. Однако, ViewContainer уже существует, и задача заключается в том, чтобы внести изменение в него, а не создать новый контейнер. Был изучен внесения модификации с использованием Monaco и существует хороший пример реализации [35]. Но тогда изменения придется вносить и в саму Theia, что повлечет большие изменения в дальнейшем. Таким образом, проблема реализации заключается в том, что была потрачена значительная часть времени на изучение наиболее простого способа реализации, который не привел к требуемому результату. Однако, это позволило хорошо разобраться в проблеме и в конечном счёте спросить мнения у сообщества разработчиков Theia.

Вернувшись к решению проблемы было выяснено следующее: так как в Theia применяется dependency injection фреймворк “Inversify”, то вся логика представляется в виде некоторых сервисов. Поэтому нужным для нас сервисом являлся

@theia/monaco/lib/browser/monaco-editor-service.MonacoEditorService, который через метод addOverlayWidget может принимать виджет типа monaco.editor.IOverlayWidget. Использование именно IOverlayWidget решает задачу отрисовки поверх активного окна текстового редактора [36]. Ниже, в листинге 3, изложен шаблон создания такого виджета.

```
let codeEditor: monaco.editor.ICodeEditor = this.editorService
      .getActiveCodeEditor()!
codeEditor.addOverlayWidget(new class implements monaco.editor.IOverlayWidget {
    getId(): string {
        /* Возвращается Id виджета*/
    }
    getDomNode(): HTMLElement {
        /* Блок отрисовки элемента графического интерфейса */
    }
    getPosition(): monaco.editor.IOverlayWidgetPosition | null {
        /* Возвращается позиция виджета */
    }
})
```

Листинг 3 – пример реализации виджета, который отображается поверх активного редактора кода

4.6. Задание статуса проекта и контроль состояния и статуса пользователей

“Задание статуса проекта” - сервис, который должен был хранить всю информацию о конфигурации проекта, а “Контроль состояния и статуса пользователей” должен был хранить информацию о пользователях. Было принято решение об объединении двух сервисов, так как в конфигурация проекта включает в себя информацию о пользователях (участники проекта, роли). Кроме того, в планах были интеграции первого и второго сервиса с RIDE индивидуально, таким образом единый сервис будет аккумулировать данные, необходимые для RIDE.

Данные сервиса используются не только RIDE, но другими приложениями проекта, которым важно знать о том, какие в данном проекте есть участники (“Интеграция с Git”), или которым нужно поменять данные об участниках (“Менеджер пользователей”). В основе приложения используется база данных, физическая схема приведена на следующей странице. Схема получилась простая, но по некоторым полям следует дать комментарии.

Таблица “project_users” необходима для представления информации о пользователях (таблица “user”) проекта Reflex (таблица “reflex_project”). В “project_users” есть поле “role”, для которого выбран тип char, так как каждую роль можно сократить до одной буквы. ‘a’ - администратор (“administrator”), ‘r’ - читатель (“reader”), ‘w’ - редактор (“writer”), ‘c’ - комментатор (“commentator”). Таблица “reflex_project” имеет поля “name”, “is_public” и “github_api_token” соответственно имя, статус проекта и токен Github. Токен Github используется сервисом “Синхронизация с Github репозиторием”. Подробнее ознакомиться с физической схемой базы данных, использующей PostgreSQL, можно на рисунке 51.

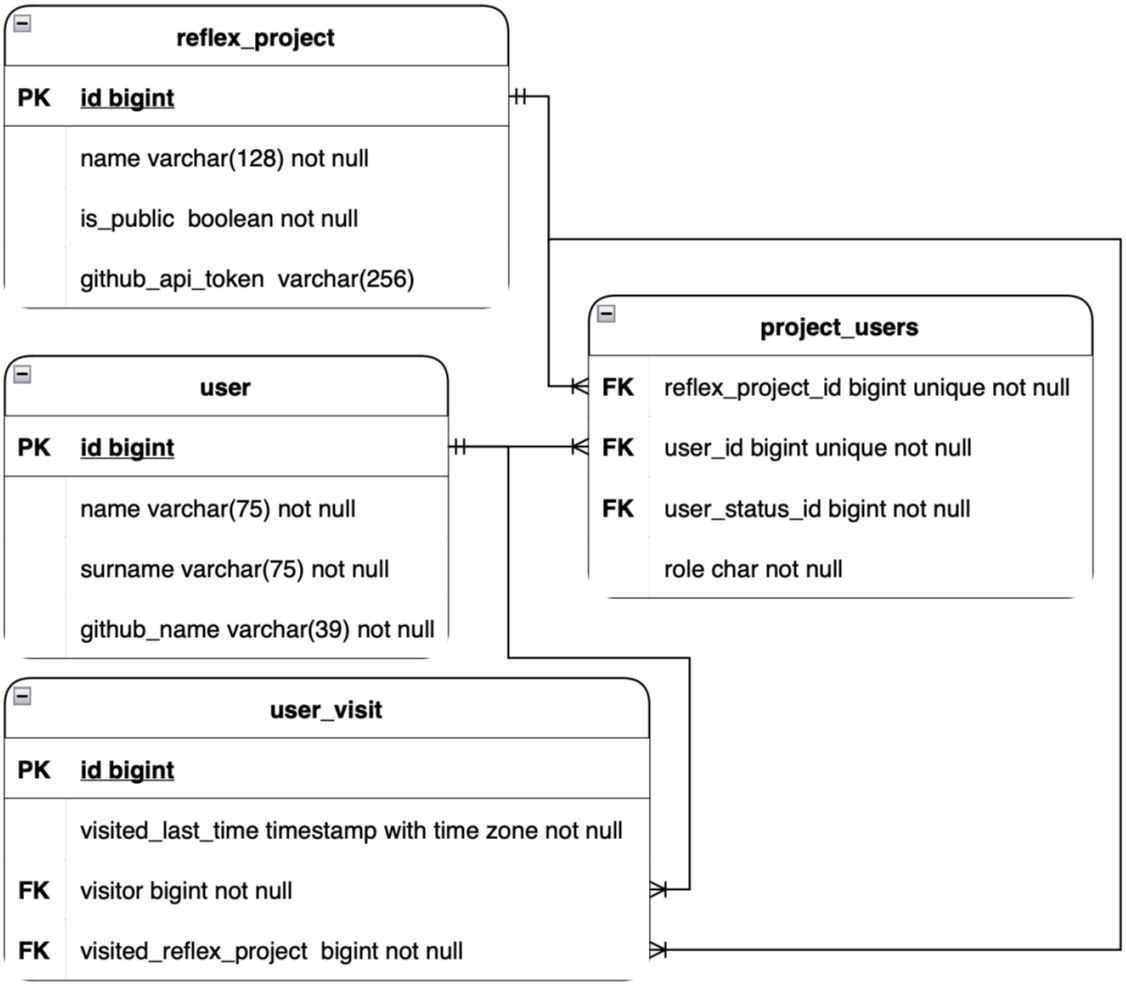


Рисунок 51 – Физическая схема данных приложения “Проект-статус и RIDE-пользователи”

4.7. Создание проекта

Когда на frontend модуля в разделе “Новый проект” в выпадающем списке “Шаблон проекта” пользователь видит возможные варианты, то это будет список с сервера “Создание

проекта”, на который идёт GET запрос на получение существующих шаблонов. Информация о шаблонах приходит в следующем виде JSON массива объектов из листинга 4.

```
[
  {
    “templateName”: “Шлюзы”,
    “templateId”: 1
  }
]
```

Листинг 4 – JSON объект, получаемый при GET запросе на “Генератор”

Пользователю предлагаются шаблоны: “Шлюзы”, “Разлив бутылок”, “Сушилка” и пустой.

После того, как пользователь выбрал одно из этого списка и завершил заполнение в формы, на сервис “Создание проекта” отправляется POST запрос с JSON структурой из листинга 5.

```
{
  “newProjectName”: “Названиe проекта”,
  “templateId”: 1,
  “projectType”: “private” | “public”
}
```

Листинг 5 – JSON объект, отправляемый при POST запросе на “Генератор”

Сервисом “Создание проекта” на Github создается репозиторий с кодом проекта-шаблона. Доступ к репозиторию (public, private) определяется на основе того, какой тип был выбран в форме.

Затем сервис “Интеграция с Git” выполняет свою задачу. Таким образом, у каждого участника проекта появляется сгенерированный шаблон. В текущей реализации доступен выбор только из трех вышеперечисленных вариантов. Реализация не предусматривает добавление или создание нового шаблона.

4.8. Чат-коммуникатор

Был использован Rocket Chat, который представляет собой уже готовый чат-сервер и имеет бесплатную версию, которую можно развернуть в Docker контейнере [37]. Хорошо описанный REST API [38] и популярность Rocket Chat [39] были ключевыми факторами для его выбора. Проект является open-source [40], что позволит в будущем создать ответвление и доработать его под требования команды.

Для того, чтобы развернуть сервис в Docker-контейнере скрипт из листинга 6.

```
# производится инициализация репликасета MongoDB
docker run --name db -d mongo:4.0 --smallfiles --replSet rs0 --oplogSize 128
docker exec -ti db mongo --eval "printjson(rs.initiate())"
# затем запускается сам инстанс Rocket Chat, прилинкованный MongoDB
docker run --name rocketchat -p 9003:3000 --link db --env MONGO_OPLOG_URL=
mongodb://db:27017/local -d rocket.chat
```

Листинг 6 – установка Rocket Chat сервера через Docker

REST API Rocket Chat представляет собою запросы к MongoDB. При возникающих трудностях можно обратиться к GitHub репозиторию jadolg/rocketchat_API, в котором используется Python как пример клиента [41]. Имея готовый API требовалось написать frontend часть “Коммуникатора”. Задача была упрощена: использовались только текстовые сообщения и одна группа.

4.9. Синхронизация с Github репозиторием

Была настроена интеграция с Github, который выполняет роль основного хранилища кода. Команда разработки RIDE сошлась на том, что это является оптимальным решением: пользователи хранят на удобном для них сервисе данные, а с разработчиков снимается ответственность на хранение кода в отдельном хранилище. Также предусматривался вариант использования Gitlab для тех случаев, когда команде важно иметь локальное хранилище, а не глобальное как в случае с Github. Аргументировать выбор Gitlab можно следующими пунктами:

- Удобство использования в силу развитого интерфейса;
- Гибкость Gitlab в настройке;
- Хорошо написанная документация API, что важно для интеграции;
- Бесплатная community edition версия;
- Простота установки с помощью Docker образа Gitlab.

Однако, в силу того, что в требованиях на главной была задача интеграции с Github, то первая реализация появилась для него. Причем в процессе учитывался момент открытости реализации для возможной интеграции с Gitlab.

В текущей реализации, при создании проекта Reflex, автоматически создается его удаленная копия на сервере, выстраиваются настройки, соответствующие настройкам проекта: доступ к репозиторию, права связанные с доступом и модификацией. Было реализовано взаимодействие “Модуль-Github”, поэтому изменение конфигурации в модуле

влияет на конфигурацию в Github. Если администратор или другой пользователь, имеющий доступ к Github, производит изменение на Git сервере, то данные на модуле обновятся. На модуле производится обновление данных раз в минуту, чтобы поддерживать взаимодействие “Github-Модуль”. Важно отметить следующее: данное взаимодействие не могло бы работать, если бы пользователь не сгенерировал токен доступа Github и не предоставил его для RIDE. Данная задача решалась разработчиками, занимающимися RIDE, и именно они решают задачу хранения ключей и получения их от пользователя.

4.10. Reflex библиотеки

Проект Reflex, если код не содержит в файлах ошибок, транслируется в C код, который расположен в директории src-gen. Кроме кода на C, в директории для сборки располагается CMakeFile - файл сборки проекта. CMake на текущий момент является де-факто кроссплатформенным стандартом для разработки на C/C++ приложений. Данная технология позволяет также упростить работу с внешними библиотечными зависимостями на этапах загрузки необходимых ресурсов и сборки в целом. Внешние зависимости хранятся на серверах, называющихся пакетными менеджерами. Самые известные из них vcpckg и Conan.io. Был выбран conan.io исходя из следующих факторов:

- Conan.io на разных платформах и в разных средах разработки использовать проще, чем vcpkg, в силу того, что vcpkg продукт Microsoft, лучше всего работает на Windows и с Visual Studio.
- Возможность организации своего Conan сервера-репозитория, где могут быть готовые бинарные сборки. Это способствует значительной экономии времени.
- Более простой удобный API для интеграции у Conan, что было необходимо для сервиса библиотек.

Далее в контейнер с Theia была установлен conan, через команду “pip install conan”. Для репозитория библиотек запущен контейнер с JFrog Artifactory Community Edition для C/C++, так как в отличие от Conan Server реализация JFrog позволяет добавлять библиотеки не только мейнтейнерам - определенному кругу лиц с доступом к главному репозиторию. Frontend модуля управления проектом на странице “Библиотеки”, выводит список библиотек, которые загружены на сервер. Каждая из библиотек представляет собой гиперссылкой на инструкцию по загрузке с сервера Conan, где есть conanfile.txt пакета, команда для локальной загрузки и информация для подключения добавления в CMakeFile.txt.

Кроме этого в интерфейсе была создана кнопка для выгрузки библиотеки пользователя на сервер Conan. Кнопка при нажатии выполняет bash-скрипт из листинга 7.

```
# Производится один раз. Необходимо для добавления сервера библиотек.
conan remote add reflex_lib_server $REFLEX_LIB_SERVER_URL
conan search -r=reflex_lib_server
# Команда, которая исполняется при нажатии кнопки выгрузки библиотеки
conan upload $REFLEX_PROJECT_NAME --all -r=reflex_lib_server
```

Листинг 7 – Выгрузка библиотеки на сервер Conan

5. Эксперимент

Микросервисы, написанные на Kotlin, в процессе разработки покрывались тестами. Для Kotlin существует библиотека MockK для создания моков (mocks) и стабов (stubs), JUnit и Spring Boot Test. Для того, чтобы сделать тестирование с моками правильным, потребовалось для сервисов Spring, то есть классов, которые помечаются аннотацией @Service, создавать сперва интерфейсы, а затем сами имплементации (от англ. implementation - реализация). Для тестирования логики, использующей взаимодействие с базой данных PostgreSQL, применялись контейнерные тесты.

Так как разработка производилась в среде IntelliJ IDEA, то была возможность воспользоваться плагином “Test coverage” [42]. Инструмент показывал для сервисов, которые писались раньше, 100% в силу того, что было достаточно времени, чтобы улучшить значения по Class, Method, Line и Branch. Однако тот факт, что было необходимо ускорить темп разработки и перенести фокус на frontend, стал причиной того, что для многих сервисов тесты отсутствуют.

Говоря про утилиты, которые в основном использовали bash скрипты, было решено воспользоваться фреймворком bats. Помог анализ Никиты Соболева [43], в котором присутвуют assert.sh, shunit2 и roundup.

Тестирование Theia приложения производилось по рекомендациям из документации Theia для версии 0.16.1 [44]. В разделе “Test directory structure” подробно описано, какая должна папочная структура проекта с тестами. В примере используется библиотека MochaJS. Она поставляется вместе с пакетами Theia, поэтому дополнительной установки не требовалось.

Большое влияние в организации тестирования оказала книга Владимира Хорикова “Принципы юнит-тестирования” [45]. В книге приведены примеры на языке C#, но это никак не влияет на понимание материала и применение его на практике. Обсуждаемые паттерны (AAA, “Простой объект”, “CanExecute/Execute” и др.) и принципы (“черный ящик”, “белый ящик”, “Fail Fast”, “Low coupling, high cohesion ” и др.) помогли сэкономить время как на придумывании тестов, так и улучшили качество: простота и понятность, скорость прохождение тестов, эффективность, которая помогает уменьшить количество тестов, их объем и покроет многие случаи.

На арендованном тестовом сервере с конфигурацией ЦП Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz, 29 Гбайт постоянной памяти и 965 Мбайт была попытка запуска части приложений (рисунок 52). Сообщение об ошибке: “Error response from daemon: mkdir

/var/lib/docker/overlay2/…/merged: no space left on device. Error: failed to start containers: <название контейнера>”.

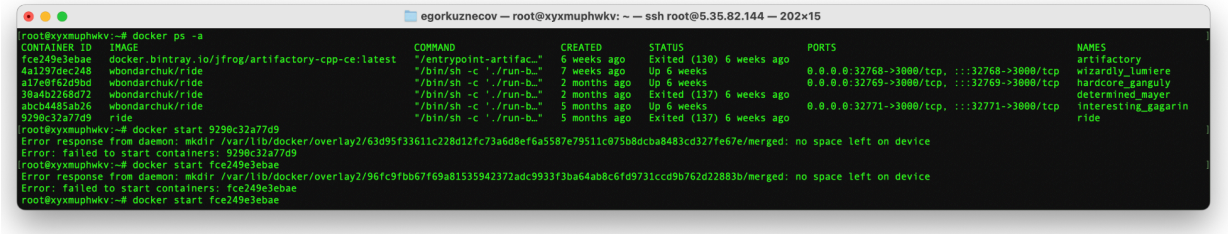


Рисунок 52 – Ошибка отсутствия свободного места в разделе диска

Решение такой проблемы одно: освобождение дискового пространства. Попытка очистки не завершилась успехом, так как ненужных данных было не так много. Кроме того, пользователь RIDE, создающий проект, создает новый контейнер с Theia. Таким образом возможность создания новых проектов Reflex ограничена небольшим объемом постоянной памяти на текущем сервере. Кроме того, стоило разместить сервер модуля управления проектами Reflex и RIDE по разным физическим машинам. Основная цель выделения в архитектуре сервера модуля, формирование его из микросервисов – уменьшить объем контейнеров Theia клиентов и разместить вычисления и работу тяжелых сервисов на более производительной машине, чем клиент Theia.

Таким образом, необходимо два сервера, на одном из которых был бы RIDE, а на другом – сервер модуля управления проектами Reflex. Для RIDE, как можно было видеть на ранее приведенном изображении, для трех сервисов достаточно такой недорогой конфигурации. Но для набора сервисов модуля, где запущены Kotlin приложения, использующие JVM требовательную к достаточным объемам оперативной памяти, Rocket Chat и центр библиотечных зависимостей Conan.

ЗАКЛЮЧЕНИЕ

В рамках данной работы был сделан анализ существующих облачных IDE решений и реализации в них многопользовательского режима. Анализ изложен в таблице из приложения А. В таблице были отображены общие компоненты приложений, наличие или отсутствие которых отмечали в ячейках. В последнем столбце делался вывод по компоненту. На основе этого сформированы требования.

Требования определили сервисы, которые будут реализовывать основную логику модуля. Перемещение большей части приложений на сервер модуля позволит уменьшить размер контейнера. Требования, которые проще реализовывать, используя возможности ОС контейнера, и которые сильно зависят от конкретного проекта, было решено реализовывать в виде набора утилит.

Спроектированы макеты для реализации графического пользовательского интерфейса. Были описаны основные экраны и поля, а также показано, как должно выглядеть многопользовательское взаимодействие в реализуемом приложении.

Реализация приложения была непростой в силу накопившихся проблем с устаревшими версиями библиотек Theia и самой Theia. Также проблемой стало то, что объем разработки и количество сервисов, интеграцию с которыми необходимо было реализовать, был весьма велик. Как результат – некачественное тестовое покрытие, низкая скорость работы приложения и его нестабильность работы. Однако факт существование первой реализации положительно сказался на мотивации для развития модуля. Привлечение дополнительных разработчиков и накопленные новые знания в разработке приложений для Theia помогут улучшить работу приложения и завершить разработку первой стабильной версии.

Большой ошибкой стало то, что не была заранее проведена правильная оценка ресурсов сервера, на котором будут запускаться все сервисы, отвечающие за работу модуля управления проектом Reflex. С другой стороны, это не стало большой проблемой, так как на данный момент разработка еще продолжается: пишутся тесты и исправляются серьезные баги. Приложения запускаются на компьютере разработчика и не представляют проблемы для локального тестирования.

Часть работы про исследование механизмов управления правами доступа в облачном IDE языка Reflex представлена в 2023 году на Международной научной конференции “Математическое и компьютерное моделирование” факультета цифровых технологий и кибербезопасности ОмГУ в городе Омск [46]. В 2024 была опубликована статья в соавторстве

с Зюбиным Владимиром Евгеньевичем в сборнике международной конференции SmartIndustryCon, размещенном в цифровой библиотеке IEEE Xplore [47], тезис и выступление на XI-й конференции “Математическое и компьютерное моделирование” [48], тезис и выступление на Международной научной студенческой конференции в секции “Инструментальные и прикладные программные системы”.

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлен с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

Кузнецов Егор Владимирович _____

ФИО студента *Подпись студента*

« ____ »_____2024г.

(заполняется от руки)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Fylaktopoulos G. et al. An overview of platforms for cloud based development //SpringerPlus. – 2016. – Т. 5. – С. 1-13.
2. Sartoni M. AWS Services for Cloud Robotics Applications : дис. – Politecnico di Torino, 2022.
3. Zheng P. et al. Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives //Frontiers of Mechanical Engineering. – 2018. – Т. 13. – С. 137-150.
4. Red Hat Developer. Red Hat Openshift Dev Spaces // Red Hat, Inc. 2024 [Электронный ресурс] – Режим доступа: https://developers.redhat.com/products/openshift-dev-spaces/overview (дата обращения 19.05.2024).
5. Red Hat & Eclipse Che | Red Hat Developer [Электронный ресурс] – Режим доступа: https://developers.redhat.com/blog/2016/06/27/red-hat-eclipse-che-2 (дата обращения 19.05.2024).
6. Visual studio code documentation [Электронный ресурс] – Режим доступа: https://code.visualstudio.com/docs (дата обращения 19.05.2024).
7. Install IntelliJ IDEA | IntelliJ IDEA Documentation [Электронный ресурс] – Режим доступа: https://www.jetbrains.com/help/idea/installation-guide.html (дата обращения 19.05.2024).
8. Top IDE index [Электронный ресурс] – Режим доступа: https://pypl.github.io/IDE.html (дата обращения 19.05.2024).
9. CodeCollab: Compile, Collaborate, Create [Электронный ресурс] – https://codecollab.io/ (дата обращения 19.05.2024).
10. Replit, “Terms of Service ("Terms")”, last updated: 11/29/2023 [Электронный ресурс] – Режим доступа: https://replit.com/site/terms (дата обращения 19.05.2024).
11. Home | Eclipse Che [Электронный ресурс] – Режим доступа: https://eclipse.dev/che/ (дата обращения 19.05.2024).
12. IntelliJ IDEA – the Leading Java and Kotlin IDE [Электронный ресурс] – Режим доступа: https://www.jetbrains.com/idea/ (дата обращения 19.05.2024).
13. Remix Etherium IDE Documentation 2024 [Электронный ресурс] – Режим доступа: https://remix-ide.readthedocs.io/en/latest/ (дата обращения 19.05.2024).

14. Arduino Cloud Web Editor [Электронный ресурс] – Режим доступа: https://support.arduino.cc/hc/en-us/categories/360002234259-Arduino-Cloud (дата обращения 19.05.2024).
15. Chedup S. et al. Performance comparison of arduino ide and runlinc ide for promotion of iot stem ai in education process //Machine Learning, Deep Learning and Computational Intelligence for Wireless Communication: Proceedings of MDCWC 2020. – Springer Singapore, 2021. – С. 237-254.
16. Martikkala A. et al. Trends for low-cost and open-source IoT solutions development for industry 4.0 //Procedia Manufacturing. – 2021. – Т. 55. – С. 298-305.
17. Eclipse Theia [Электронный ресурс] – Режим доступа: https://projects.eclipse.org/projects/ecd.theia (дата обращения 19.05.2024).
18. Eclipse Theia | Authoring VS Code Extensions [Электронный ресурс] – Режим доступа: https://theia-ide.org/docs/authoring_vscode_extensions/ (дата обращения 19.05.2024).
19. Theia - Cloud and Desktop IDE Platform [Электронный ресурс] – Режим доступа: https://theia-ide.org/ (дата обращения 19.05.2024).
20. Tinkercad - Circuits [Электронный ресурс] – Режим доступа: https://www.tinkercad.com/circuits (дата обращения 19.05.2024).
21. Tinkercad - Classrooms [Электронный ресурс] – Режим доступа: https://www.tinkercad.com/classrooms-resources (дата обращения 19.05.2024).
22. Introduction to Eclipse Che :: Eclipse Che Documentation [Электронный ресурс] – Режим доступа: https://eclipse.dev/che/docs/stable/overview/introduction-to-eclipse-che/ (дата обращения 19.05.2024).
23. Try Red Hat OpenShift :: Developer sandbox [Электронный ресурс] – Режим доступа: https://www.redhat.com/en/technologies/cloud-computing/openshift/try-it (дата обращения 19.05.2024).
24. Gornev I. A., Bondarchuk V. V. Towards Collaborative Coding in RIDE Web IDE //2023 IEEE XVI International Scientific and Technical Conference Actual Problems of Electronic Instrument Engineering (APEIE). – IEEE, 2023. – С. 970-974 .
25. AWS named as a Leader in 2023 Gartner Magic Quadrant for Strategic Cloud Platform Services for thirteenth year in a row | AWS News Blog [Электронный ресурс] – Режим доступа: https://aws.amazon.com/ru/blogs/aws/read-the-2023-gartner-magic-quadrant-for-strategic-cloud-platform-services/ (дата обращения 19.05.2024).

26. Cloud IDE - AWS Cloud9 - AWS [Электронный ресурс] – Режим доступа: https://aws.amazon.com/cloud9/ (дата обращения 19.05.2024).
27. About Open VSX Registry [Электронный ресурс] – Режим доступа: https://open-vsx.org/about (дата обращения 19.05.2024).
28. Releases - eclipse-theia/theia (github.com) [Электронный ресурс] – Режим доступа: https://github.com/eclipse-theia/theia/releases (дата обращения 19.05.2024).
29. Ktor: Build Asynchronous Servers and Clients in Kotlin | Ktor Framework [Электронный ресурс] – Режим доступа: https://ktor.io/ (дата обращения 19.05.2024).
30. Loeliger J., McCullough M. Version Control with Git: Powerful tools and techniques for collaborative software development. – "O'Reilly Media, Inc.", 2012. – С. 4-5.
31. The Missing Bit | Using CMake to deploy to AVR microcontrollers [Электронный ресурс] – Режим доступа: https://www.kuon.ch/post/2018-07-11-avr-cmake/ (дата обращения 19.05.2024).
32. Hans Eirik Bull, Brian S. Dean, Stefan Rüger und Jörg Wunsch. AVRDUDE. A program for downloading/uploading AVR microcontroller flash, EEPROM and more for AVRDUDE, Version 7.3-20240207 - Free Software Foundation, 2024. – С. 66-68.
33. evilstreak/markdown-js: A Markdown parser for javascript (github.com) [Электронный ресурс] – Режим доступа: https://github.com/evilstreak/markdown-js (дата обращения 19.05.2024).
34. Git - gitignore Documentation (git-scm.com) [Электронный ресурс] – Режим доступа: https://git-scm.com/docs/gitignore (дата обращения 19.05.2024).
35. How to create a collaborative code editor with Monaco, Yjs, Next.js, and Liveblocks | Liveblocks Documentation [Электронный ресурс] – Режим доступа: https://liveblocks.io/docs/guides/how-to-create-a-collaborative-code-editor-with-monaco-yjs-nextjs-and-liveblocks (дата обращения 19.05.2024).
36. IOverlayWidget | Monaco Editor API (microsoft.github.io) [Электронный ресурс] – Режим доступа: https://microsoft.github.io/monaco-editor/typedoc/interfaces/editor.IOverlayWidget.html (дата обращения 19.05.2024).
37. rocket.chat - Official Image | Docker Hub [Электронный ресурс] – Режим доступа: https://hub.docker.com/_/rocket.chat (дата обращения 19.05.2024).
38. REST API | Rocket.Chat Developer [Электронный ресурс] – Режим доступа: https://developer.rocket.chat/reference/api/rest-api (дата обращения 19.05.2024).

39. Pumble | 15 Best Team Chat Apps in 2024 [Электронный ресурс] – Режим доступа: https://pumble.com/learn/pumble/best-team-chat-apps (дата обращения 19.05.2024).
40. RocketChat/Rocket.Chat: The communications platform that puts data protection first. (github.com) [Электронный ресурс] – Режим доступа: https://github.com/RocketChat/Rocket.Chat (дата обращения 19.05.2024).
41. rocketchat_API/rocketchat_API/APISections at master · jadolg/rocketchat_API (github.com) [Электронный ресурс] – Режим доступа: https://github.com/jadolg/rocketchat_API/tree/master/rocketchat_API/APISections (дата обращения 19.05.2024).
42. Code coverage | IntelliJ IDEA Documantation [Электронный ресурс] – Режим доступа: https://www.jetbrains.com/help/idea/code-coverage.html (дата обращения 19.05.2024).
43. Тестирование Bash приложений, Никита Соболев [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/278937/ (дата обращения 19.05.2024).
44. docs/Testing.md | Github, eclipse-theia/theia branch 0.16.1 [Электронный ресурс] – Режим доступа: https://github.com/eclipse-theia/theia/blob/0.16.1/doc/Testing.md (дата обращения 19.05.2024).
45. Khorikov V. Unit Testing Principles, Practices, and Patterns. – Simon and Schuster, 2020 – 304 с.
46. Кузнецов, Е. В. Исследование механизмов управления правами доступа в облачном IDE языка Reflex // Математическое и компьютерное моделирование. Сборник материалов X Международной научной конференции. – Омск, 2023. – С. 253-254.
47. Kuznetsov Y., Zyubin V. Development of Project Management Module for Reflex Cloud IDE //2024 International Russian Smart Industry Conference (SmartIndustryCon). – IEEE, 2024. – С. 957-961.
48. Кузнецов, Е. В. Разработка модуля управления проектами для облачного IDE языка Reflex // Математическое и компьютерное моделирование. Сборник материалов XI Международной научной конференции. – Омск, 2024. – С. 164-166.

ПРИЛОЖЕНИЕ А

Сравнительная таблица анализа существующих решений

Список IDE, участвующих в анализе:

1) codecollab.io;
2) replit.com;
3) VS Code;
4) Intellij IDEA;
5) Remix Etherium IDE;
6) Arduino Cloud Web Editor;
7) RIDE;
8) Tinkercad;
9) Eclipse Che;
10) AWS Cloud9.

Значение цветов ячеек столбца “Вывод по пункту”: зеленый – уже присутствует в RIDE, желтый – отсутствует в RIDE, необходимо реализовать.

Таблица А.1 - Таблица сравнительного анализа существующих IDE решений

Номер IDE / Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Описание страницы приветствия											В большинстве сред разработки используется вариант выбора проекта из набора проектов перед началом работы.
Сайт-презентация	+	-	-	-	-	+	+	-	-	-	
Начало работы, выбор проекта	-	+	+	+	+	-	-	+	+	+	

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Наличие терминала	+	+	+	+	+	-	+	-	+	+	Терминал есть, но можно сделать решение без его использования. С другой стороны, терминал - это возможность установки различных консольных приложений, а также возможность для тонкой настройки нашего рабочего пространства/контейнера.
Многооконный режим. Открытие нескольких файлов на редактирование, настройка интерфейса.	+	+	+	+	+	-	+	+	+	+	Присутствует.

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Система контроля версий Git											Для Eclipse Theia существует магазин приложений, в котором есть инструменты для взаимодействия с git.
Наличие git	+	+	+	+	+	-	-	-	+	+	
Графическое взаимодействие с git	-	+	+	+	+	-	-	-	+	+	
Кнопка “поделиться” - предоставление возможности подключения к проекту для других пользователей	+	+	+	+	+	+	-	+	+	+	Стоит добавить кнопку “поделиться” и “присоединиться”.
Групповая разработка. Взаимодействие с людьми в команде.	+	+	+	+	-	-	-	+	+	+	Необходимо реализовать текстовый чат, а затем голосовой.
Модерация. Роли и управление правами доступа											Сделать read-only, editor роли. Предоставить возможность
Либо полный доступ, либо только на чтение	+	+	+	-	-	-	-	+	-	-	

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Только на чтение, на изменение, полный доступ, настраиваемый доступ, группы пользователей	-	-	-	+	-	-	-	-	+	+	назначения ролей участникам проекта. Автоматически создавать нового пользователя с ОС контейнера, если говорим про взаимодействие с терминалом.
Разработка с мобильных устройств	+	+	+	-	-	+	-	+	-	-	Предлагать пользователю перейти в desktop версию и включать read-only режим, если нет возможности писать.
Документация и ресурсы к среде разработки											Примеры хорошего сайта с документацией: VS Code и IDEA. Пример простого доступа к документации: home-вкладка
Внутри IDE	+	+	-	-	-	+	-	-	-	-	

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
На отдельном сайте	-	-	+	+	+	+	-	+	+	+	remix.etherium, т.к. он на первом плане знакомит со своими языками, что мы планируем делать. По VS Code и IDEA достаточно видео с разборами различных задач на youtube: настройка установка и пр., что соответственно должно быть у RIDE.
Доступ к коду в GitHub	-	-	+	+	+	+	-	-	+	-	
Обратная связь с пользователями											В силу небольшого размера проекта, лучше всего подойдёт Почта, вопрос коммьюнити, баг репорты.
Почта	+	+	+	+	+	+	-	+	+	+	
Форум сообщества	-	+	+	-	+	+	-	+	+	+	
Форма для сообщения о уязвимостях	-	+	+	+	+	-	+	-	+	+	

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Возможности для разработчиков дополнений											Сделать магазин дополнений, руководство по созданию дополнения, пример дополнения с открытым кодом. Создать возможность для монетизации разработчиков дополнений.
Инструкция по созданию дополнения	-	-	+	+	-	-	+	-	+	+	
Магазин дополнений	-	+	+	+	+	-	+	-	+	+	
Добавление платного контента	-	-	+	+	-	+	-	-	+	+	
Безопасность											Разработать privacy policy и/или terms. Спрашивать пользователя при использовании не гарантирующих безопасную работу файлов.
Регистрация по почте	+	+	+	+	+	+	-	+	+	+	
Регистрация через GitHub	+	+	+	+	+	+	-	-	+	+	

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Добавление банковской карты при регистрации	-	-	-	-	-	-	-	-	-	+	
Логика compile/start/run											Виртуальный контроллер, развертывание, Средства отладки. Создать данные кнопки. Продумать, как пользователь будет настраивать конфигурацию для них.
Привязана к языку программирования в созданном проекте	+	+	-	-	+	+	-	+	-	-	
Возможность изменения строки по умолчанию для компиляции	+	-	+	+	-	-	-	-	+	+	
Механизмы финансирования											Больше подходит стратегия от replit.com.
Спонсоры в лице институтов/университетов	+	-	-	+	-	+	-	-	-	-	
Пользовательская подписка	-	+	-	+	-	+	-	-	+	+	

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Подписки для организаций	-	+	-	+	-	+	-	-	+	+	
Пользовательские пожертвования	+	+	-	-	+	-	-	-	-	-	
Выгрузка проекта из git репозитория											Добавить взаимодействие с Github.
В GitHub	+	+	+	+	+	-	-	-	+	+	
Напрямую к себе	+	+	+	+	+	+	+	+	+	+	Возможность сохранения проекта из RIDE уже осуществлена.
Возможность создания шаблонов проекта	-	-	+	+	+	+	-	+	+	+	Добавить шаблоны.
Подсветка синтаксиса популярных языков программирования	+	+	+	+	+	+	+	-	+	+	Добавить подсветку синтаксиса для XML и C, т.к. poST/Reflex программы создают файлы с такими расширениями.

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Возможность просмотра .md файлов	-	+	+	+	+	-	-	-	+	+	Встроить изначально или позже написать дополнение.
Встроенный построитель UML-диаграмм	-	-	-	+	-	-	-	-	-	-	Можно обойтись без данного инструмента.
Создание нового проекта по шаблону											Для начала написать простой шаблон проекта Reflex. Затем расширить возможности с помощью генераторов.
Пользователю предлагается список языков программирования. Для каждого создается “Hello world” файл на этом языке.	+	+	+	+	+	+	-	+	+	+	
New Project, Empty Project - кнопки для создания соответственно нового и пустого проекта	-	+	-	+	-	-	-	+	+	+	

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Создание с помощью плагинов-генераторов	-	-	+	+	-	-	-	-	+	+	
New Project, Empty Project - кнопки для создания соответственно нового и пустого проекта	-	+	-	+	-	-	-	+	+	+	
Создание с помощью плагинов-генераторов	-	-	+	+	-	-	-	-	+	+	
Способы настройки существующего проекта											Добавить возможность изменения названия проекта. Заложить в разработке возможность изменения версии компилятора, когда появится другая версия Reflex. Создание новых модулей необходимо
Изменение названия	+	+	+	+	+	+	-	+	+	+	
Изменение версии компилятора	-	-	+	+	-	+	-	-	+	+	
Добавление новых модулей в проекте	-	-	-	+	-	-	-	-	+	+	

Продолжение таблицы А.1

Номер IDE Параметры	1	2	3	4	5	6	7	8	9	10	Вывод по пункту
Настройка используемых библиотек	-	-	+	+	-	+	-	-	+	+	для разбития логики на небольшие блоки.
Отслеживание действий пользователей при групповой разработке											Добавить отслеживание пользователей. Для этого можно использовать аватары или подписывать именем элементы интерфейса, отвечающие за ориентацию пользователя (курсор мыши, курсор, текстовый курсор и т.д.).
Аватары участников в блоке “Участники”	+	+	+	-	-	-	-	+	+	+	
Курсоры разных цветов для разных участников с их именем	+	+	+	-	-	-	-	+	+	+	

ПРИЛОЖЕНИЕ Б

МОДУЛЬ УПРАВЛЕНИЯ ПРОЕКТАМИ ОБЛАЧНОГО IDE ЯЗЫКА REFLEX

ОПИСАНИЕ ПРОГРАММЫ

Листов 12

Новосибирск, 2024

СОДЕРЖАНИЕ

АННОТАЦИЯ

В данном программном документе приведено описание реализации модуля управления проектами в облачном IDE языка Reflex, который может быть использован в уже существующих системах.

Основной функционал программной системы – это выявление и обработка намерений пользователя, представленных программой, на основе переданных на вход названий интентов, а также их параметров.

Оформление программного документа “Описание программы” произведено по требованиям ЕСПД (ГОСТ 19.402-78, ГОСТ 19.105-78).

1. Общие сведения

1.1. Обозначение и наименование программы

Полное наименование программы – модуль управления проектами для облачного IDE языка Reflex.

1.2. Программное обеспечение, необходимое для функционирования программы

Программа работает на любой операционной системе, которая имеет Web браузер. Серверная часть программы написанная при помощи фреймворка Spring Boot – это Kotlin-фреймворк, используемый для программирования автономных приложений на основе Spring. Docker и docker-compose для запуска сервера.

Клиентская часть программы написана при помощи внутреннего фреймворка Eclipse Theia для разработки плагинов на TypeScript.

1.3. Языки программирования

Исходным языком программирования является Kotlin версии 1.9.22 и TypeScript 8.10.1. Для скриптов используется Linux bash.

2. Функциональное назначение

2.1. Назначение программы

Модуль предназначен для создания и разработки Reflex проектов в рамках конкретной предметной области. В основе работы модуля используется взаимодействие со внешними сервисами (conan, Rocket Chat и Github), а также Git и файловой системой Linux.

3. Описание логической структуры

3.1. Алгоритм, структура программы и используемые методы

Главным классом каждого подмодуля сервера является класс запуска модуля (Application), отвечающий за конфигурацию и за начало работы модуля.

Программа сервера состоит из нескольких подмодулей, каждый из которых имеет свою собственную ответственность.

Программа клиента использует модуль, который автоматически подключается к Theia за счёт биндинга (от англ. bind “связывать”).

Классы запуска модулей представлены в таблице Б.1.

Таблица Б.1 – Класс запуска модуля

Класс	Назначение класса
Application	Запуск подмодуля сервера
ContainerModule	Связывание интерфейса модуля с интерфейсом самого IDE, его интеграция и запуск с главным приложением.

3.1.1. Подмодуль сервера для задания статуса проекта

Данный подмодуль отвечает за предоставление статуса проекта и его обновление. Имеет контроллер ProjectStatusController и соответствующий сервис ProjectStatusService (таблица Б.2).

Таблица Б.2 – Методы класса ProjectStatusService

Имя	Тип результата	Назначение
getProjectStatus	ProjectStatusDto	Получить Id проекта, найти проект по Id в базе. Если найден проект – вернуть его статус. Иначе – ошибка.
updateProjectStatus	void	Получить Id проекта и обновленный статус, найти проект по Id в базе. Заменить в найденном проекте поле статуса на актуальное. Сохранить изменения.

3.1.2. Подмодуль сервера для создания проекта

Данный подмодуль отвечает за создание нового проекта Relex в зависимости от выбранного проекта по умолчанию. Имеет контроллер ProjectGeneratorController и сервис ProjectGeneratorService (таблица Б.3).

Таблица Б.3 – Методы класса ProjectGeneratorService

Имя	Тип результата	Назначение
getProjectTemplates	List<ProjectTemplateDto>	Получить список доступных шаблонов.
createNewProject	NewProjectDto	Получить имя проекта, его область видимости и генератор для него. Сохранить полученные данные в базе. Вернуть структуру с созданным проектом и его сгенерированным Id.

3.1.3. Подмодуль сервера для контроля состояния статуса пользователей

Данный подмодуль отвечает за состояния статуса пользователя. Состояния статуса пользователя включают в себя роль и статус активности. Имеет контроллер UserStatusController и сервис UserStatusService (таблица Б.4).

Таблица Б.4 – Методы класса UserStatusService

Имя	Тип результата	Назначение
getUsers	List<UserDto>	Получить Id проекта. Найти в базе пользователей проекта. Вернуть пользователей проекта.
addUser	void	Получить Id проекта и Id пользователя. Добавить в базу данных связь “Проект-пользователь”.
updateUser	void	Получить Id проекта и обновленного пользователя. Найти пользователя и изменить его данные в проекте на актуальные.

3.1.4. Утилита интеграции с Git

Данная утилита отвечает за интеграцию с Git. Запускает bash-скрипт, который по полученным данным о пользователях создает локальные репозитории и определяет такой режим доступа в файловой системе для проекта, который был передан скрипту (таблица Б.5).

Таблица Б.5 – Функции скрипта git_integrate.sh

Имя	Назначение
createUser	Получить имя пользователя. Создать для этого пользователя по имени Linux пользователя.
createLocalGitRepo	Получить имя пользователя. Стать нужным пользователем по полученному имени. От такого пользователя клонировать в домашнюю директорию Git проект с заданной ссылкой. Установить права на модификацию и для прочих только на чтение. Вернуться к начальному пользователю.

3.1.5. Утилита генерации C для платформы ATmega

Данная утилита отвечает за генерацию C кода для платформы ATmega. Осуществляется вызов bash-скрипта при нажатии зеленой кнопки главного меню интерфейса модуля.

3.1.6. Модуль Theia клиента

Данный модуль осуществляет отображение информации в интерфейсе.

4. Используемые технические средства

Программа сервера может быть запущена на любом вычислительном устройстве, где установлена JDK версии 21 и новее, а также Docker и docker-compose. Программа клиента запускается на любом вычислительном устройстве, на котором имеется веб-браузер, отвечающий следующим требованиям: Edge 121.0.2277 и новее, Opera 108.0.5067 и новее, YaBrowser 24.1.3.809 и новее, Mozilla Firefox 123.0.1 и новее, Safari 16.5.2 и новее или Google Chrome 122.0.6261 и новее.

5. Вызов и загрузка

Прежде всего необходимо убедиться, что Github, Rocket Chat и conan серверы запущены. Затем запуск каждого серверного подмодуля осуществляется следующим образом: с помощью среды IntelliJ IDEA открыть проект ReflexProjectManagmentModule и запустить все классы с названием Application, нажатием на зелёную кнопку в данной среде.

Запуск клиентской части приложения осуществляется открытием пользователем проекта в RIDE в веб-браузере по адресу https://ride.poporg.com, в левой вертикальной панели которого будет располагаться кнопка с иконкой Reflex.

6. Входные данные

В таблице Б.6 представлены входные данные подмодулей.

Таблица Б.6 – Входные данные подмодулей

Подмодуль	Входные данные
Подмодуль сервера для задания статуса проекта	Id проекта и статус проекта
Подмодуль сервера для создания проекта	Название проекта, шаблон для генератора и область видимости проекта, Id проекта
Подмодуль сервера для контроля состояния пользователей	Имя пользователя, структура UserDto пользователя, Id проекта
Утилита интеграции с Git	Ссылка на удаленный Git репозиторий, список пользователей проекта с их именами и правами доступа
Утилита интеграции С для платформы ATmega	Модуль микроконтроллера ATmega, путь до проекта
Модуль Theia клиента	Нажатия манипулятором мышь в области редактора Eclipse Theia, входные данные с форм ввода, нажатия на кнопки

7. Выходные данные

В таблице Б.7 представлены выходные данные для сервисов серверного модуля системы.

Таблица Б.7 – Выходные данные сервисов

Сервис	Выходные данные
Подмодуль сервера для задания статуса проекта	Статус проекта
Подмодуль сервера для создания проекта	Шаблоны генератора проекта, структура нового проекта
Подмодуль сервера для контроля состояния пользователей	Список пользователей, структура нового пользователя
Утилита интеграции с Git	Файловая структура проекта Reflex для совместной разработки и использования Git
Утилита интеграции С для платформы ATmega	CMakeFile.txt для сборки под нужную версию ATmega, файлы С кода, *.hex файл для отправки на микроконтроллер.
Модуль Theia клиента	Окно Theia дополнения (виджета), маркеры области активности пользователя в текстовом редакторе (цветные прямоугольники и курсоры с именем пользователя), нотификация о подключившихся новых пользователей по ссылке

8. Лист регистрации изменений

Таблица Б.8 – Лист регистрации изменений в программном документе “Описание программы”

Номера листов (страниц)					Всего листов (страниц) в документе	№ домента	Входящий № сопроводительного документа	Подпись	Дата
Номер изм.	измененных	замененных	новых	аннулированных					

ПРИЛОЖЕНИЕ В

МОДУЛЬ УПРАВЛЕНИЯ ПРОЕКТАМИ ОБЛАЧНОГО IDE ЯЗЫКА REFLEX

РУКОВОДСТВО ОПЕРАТОРА

Листов 8

Новосибирск, 2024

СОДЕРЖАНИЕ

АННОТАЦИЯ

В данном документе приведено руководство программиста для модуля управления проектами в облачном IDE языка Reflex.

Функцией модуля является управление проектом в облачном IDE языка Reflex: создание проекта и его разработка в режиме команды.

Исходными языками программирования являются Kotlin, TypeScript и Linux Bash.

Оформление программного документа “Руководство оператора” произведено по требованиям ГОСТ 19.505-79 “ЕСПД. Руководство оператора” и ГОСТ 19.105-78 “Единая система программной документации (ЕСПД). Общие требования к программным документам (с Изменением N 1)”.

1. Назначение инструмента

Программное средство предназначено для создания Reflex проектов в облачном IDE и их дальнейшей разработки в многопользовательском режиме для команд разработчиков.

Модуль управления проектами в облачном языка Reflex решает следующие задачи:

- создание проекта Reflex по шаблону;
- интеграция с git репозиторием;
- определение видимости проекта;
- определение условия использования проекта;
- документирование проекта;
- назначение ролей участникам проекта (владелец, администратор, редактор, комментатор, читатель);
- идентификация подключившихся пользователей;
- отображение текущего положения курсора онлайн-пользователей;
- установка области отображения на текущее положение курсора онлайн-пользователя;
- коммуникация между участниками проекта;
- использование сторонних библиотек;
- создание библиотек;
- сохранение копии проекта на локальный или облачный диск пользователя;
- восстановление проекта из сохраненной копии;
- запуск генератора Си файла для платформы ATmega.

2. Условия выполнения программы

2.1. Минимальный состав аппаратных средств

Минимальные требования клиента зависят от минимальных требований веб-браузера, который он использует. Ознакомиться с возможными для использования браузерами можно в пункте 2.2.

Для работы программы сервера необходимо устройство, удовлетворяющее минимальным требованиям из таблицы В.1.

Таблица В.1 – Требования к техническим средствам

Параметр	Минимальные требования
Объем оперативной памяти	От 4 Гб оперативной памяти для обеспечения работы нескольких JVM приложений сервисов conan и Rocket Chat
Дисковая память	100 ГБ и более для хранения Reflex библиотек и данных в БД Postgres и MongoDB, которая относится к сервису Rocket Chat

2.2. Минимальный состав программных средств

Docker и docker-compose для сборки и запуска приложений сервера. JDK 21 версии и выше для работы Kotlin приложений.

Программа клиента запускается на любом вычислительном устройстве, на котором имеется веб-браузер, отвечающий следующим требованиям: Edge 121.0.2277 и новее, Opera 108.0.5067 и новее, YaBrowser 24.1.3.809 и новее, Mozilla Firefox 123.0.1 и новее, Safari 16.5.2 и новее или Google Chrome 122.0.6261 и новее.

2.3. Требования к оператору

Конечный оператор программы должен обладать практическими навыками работы с интерфейсами командной строки, Docker. Оператор должен иметь представление о работе с облачными IDE и знать язык Reflex.

3. Выполнение программы

3.1. Загрузка и запуск программы

Запуск и загрузка программы осуществляется в следующей последовательности. Производится клонирование репозитория сервера модуля и поднятие контейнеров (Листинг В.1).

```
git clone https://github.com/egorkuzn/ReflexProjectManagmentServer.git
cd ReflexProjectManagmentServer
docker-compose up
```

Листинг В.1 – Клонирование репозитория сервера модуля

Затем пользователем создается проект в RIDE по адресу https://ride.poporg.com. В данный момент автоматически создается контейнер с Theia, в котором вызывается команда из листинга В.2.

```
yarn run start --hostname *hostname* --port *port*
```

Листинг В.2 – Запуск Theia приложения, в процессе которого происходит связывание виджета с IDE

После того, как новый проект запущен, оператор получает возможность использования модуля.

3.2. Выполнение программы

В левой вертикальной панели окна Theia располагается модуль управления проектами Reflex. Нажатие на иконку виджета открывает главное меню и таким образом предоставляется возможность использования модуля через графический интерфейс IDE.

Главное меню предоставляет информацию об активных участниках проекта, названии проекта. В верхней части окна главного экрана располагаются кнопки запуска генератора С программы, кнопка редактирования параметров проекта. В нижней части окна располагаются следующие кнопки: “Новый проект”, “Библиотеки”, “Документация” и “Git”. Эти кнопки открывают соответствующие экраны настроек, из которых можно вернуться на главный экран с помощью кнопки “Назад”.

3.3. Завершение работы программы

Завершение работы программы сервера сопровождается остановкой работы контейнеров сервера. Завершение работы программы клиента происходит вместе с завершением работы окна IDE – закрытие вкладки браузера.

4. Сообщения оператору

Оператор получает сообщения в графическом окне модуля. Через это окно производится основное информирование оператора. Кроме этого сообщения появляются в виде всплывающих окон нотификации в правом нижнем окне среды разработки, а также сообщение активных областей участников команды сопровождается выделением прямоугольниками и указателями с именем активного участника.

5. Лист регистрации изменений

Таблица В.2 – Лист регистрации изменений руководства программиста

Номера листов (страниц)					Всего листов (страниц) в документе	№ домента	Входящий № сопроводительного документа	Подпись	Дата
Номер изм.	измененных	замененных	новых	аннулированных					