ВИДЕОМЕССЕНДЖЕР ДЛЯ СОВМЕСТНОЙ РАБОТОЙ НАД КОДОМ

Н. Р. Спиричева, А. В. Андронов,
А. С. Коновалов, И. П. Мишурин, М. В. Цветков
Институт радиоэлектроники и информационных технологий — РтФ
Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина,
Екатеринбург, Россия,
nr.spirichevf@urfu.ru

Аннотация. Современная разработка программного обеспечения все больше осуществляется удаленно. Это создает необходимость в эффективных инструментах коммуникации и совместной работы. Видеомессенджер для совместного редактирования кода является актуальным решением для устранения преград, связанных с удаленной работой и обучением, обеспечением эффективной коммуникации и сотрудничества в режиме реального времени.

***Ключевые слова*:** видео мессенджер; редактирование кода; совместная разработка.

Введение

Современный мир требует новых средств и технологий коммуникации. Удаленная работа, удаленное и гибридное обучение является нормой современной жизни и требует подходящих инструментов. Разработчикам программного обеспечения необходимы мессенджеры, предоставляющие возможность совместной работы над кодом. У популярных на сегодняшний день систем такая возможность отсутствует. Видеомессенджер, обеспечивающий возможность коммуникации и взаимодействия в режиме реального времени, станет неотъемлемым компонентом рабочего и учебного процесса.

Проблемы удаленной коммуникации

В современном мире цифровизация и автоматизация охватывают все больше сфер повседневной человеческой жизни. Можно долго спорить о том, какие плюсы и минусы у такой стремительной цифровизации, однако спорить о том, что жизнь обычных людей становится проще с появлением новых программных продуктов, невозможно.

Это значит, что от того, насколько компетентными будут разработчики и от того, насколько хорошие продукты они будут выпускать, напрямую зависит качество жизни человека.

Развитие любого специалиста невозможно представить без тесной коммуникации с коллегами. Успех процесса обучения программированию начинающих разработчиков напрямую зависит от качества коммуникации с преподавателем.

В настоящее время процесс взаимодействия между двумя разработчиками, если они находятся удаленно, представляет собой следующее: один человек демонстрирует экран с кодом и объясняет его другому. При этом второй разработчик видит только то, что ему показывает первый, он не может самостоятельно открывать файлы и смотреть, как устроен код. Он вынужден словами объяснять, какой фрагмент кода он может посмотреть. Также если второй разработчик захочет внести изменения в код первого, то ему нужно будет словами объяснять, что необходимо изменить первому разработчику. Подобная сложность во взаимодействии усложняет процесс поиска. Или другой пример, когда преподаватель объясняет студентам какой-либо алгоритм или шаблон проектирования, студенту сложнее понять происходящее на экране при подобном взаимодействии. Эту проблему решает наш проект.

Продемонстрируем тенденции в работе разработчиков программного обеспечения [1]. Данные, предоставленные на рис. 1, свидетельствуют о значительной доле разработчиков, предпочитающих удаленную работу или гибридный формат.

Архитектура системы

Приложение состоит из четырех основных модулей, схема архитектуры представлена на рис. 2:

- клиентская часть, представляющей собой SPA- и Desktop-приложения;
- серверное приложения, доставляющего статическое содержание страницы (Web Application);
- серверное приложения представляющего собой API, на который приходят все запросы для изменения состояния приложения (создание/удаления пользователя, встречи, канала и т. д.);
- серверное приложение для модуля совместного редактирования кода.

Рис. 1. Распределение сотрудников по формату работы за 2023 год

Для серверного приложения была выбрана трехслойная архитектура [2]. Приложение состоит из трех слоев: представления, бизнес-логики и слоя доступа к базе данных. В этом проекте слой представления отвечает за получение и обработку запросов, передачу их на слой бизнес логики. Слой бизнес-логики отвечает за манипуляцию над данными, передачу или запрос данных у слоя доступа к данным, а также обращение к сторонним ресурсам. При этом верхние уровни зависят от нижних, и уровень доступа к данным не зависит ни от чего.

В качестве языка программирования для создания серверной части приложения был выбран C#, в качестве веб-фреймворка использован веб-фреймворк ASP.NET [3].

Для создания базы данных была использована СУБД PostgreSQL, для работы с ней использован Entity Framework — ORM для.NET [4].

Для написания клиентской части был выбран язык программирования TypeScrip t [5].

В качестве веб-сервера для доставки статического содержания сайта (HTML, JavaScript, CSS) использован Node.js [6, 7], библиотека для клиент-серверной коммуникации — Signal R.

Функциональные требования

Для успешной разработки приложения необходимо определить функциональные требования, которые будут удовлетворять потребности пользователей и обеспечивать высокую производительность и удобство использования. В случае разработки видеомессенджера с возможностью совместного редактирования кода основные требования должны соответствовать следующим критериям:

1) аутентификация и авторизация (регистрация новых пользователей и управление учетными записями; вход в систему с использованием логина и пароля или других методов аутентификации);

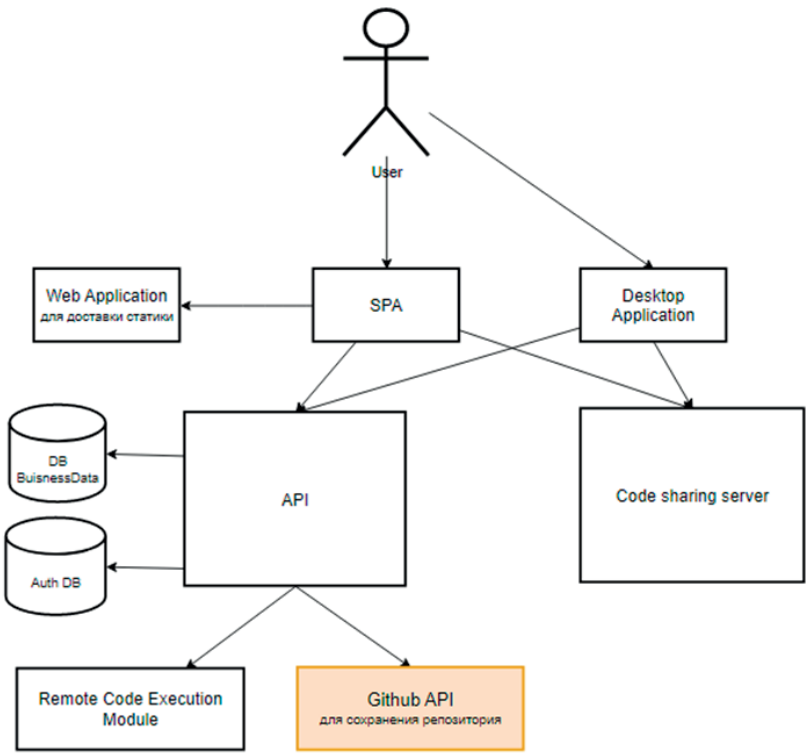

Рис. 2. Архитектура системы

2) создание и управление комнатами (поиск и присоединение к существующим комнатам; возможность создания новых комнат для совместной работы над кодом; управление доступом к комнатам, определение прав доступа пользователей);

3) обмен сообщениями и видеочат (возможность проведения видеочата между участниками комнаты; поддержка функций микрофона и камеры для общения по видеочату; отправка и получение текстовых сообщений в режиме реального времени между участниками комнаты);

4) совместное редактирование кода (реализация функциональности совместного редактирования кода в режиме реального времени; отображение изменений кода других участников комнаты синхронно и автоматическое обновление интерфейса; поддержка подсветки синтаксиса, автодополнения кода и других функций, облегчающих работу с кодом);

5) управление файлами и проектами (проектов между участниками комнаты; организация файловой структуры проектов и удобный доступ к файлам и папкам; поддержка операций копирования, перемещения и удаления файлов);

6) интеграция с серверной частью (установка соединения с сервером для передачи данных, обновлений и событий в режиме реального времени; обмен данными с сервером через API с использованием технологии Axios; обработка и отображение ошибок и статусов ответов сервер);

7) поддержка различных типов кода (поддержка различных языков программирования и их синтаксиса при совместном редактировании кода);

8) надежность и безопасность (обеспечение защиты данных и конфиденциальности пользователей; обработка ошибок и исключений для предотвращения сбоев в работе приложения);

9) интерфейс и пользовательский опыт (создание интуитивно понятного и привлекательного пользовательского интерфейса; организация навигации и удобство использования функций приложения; поддержка адаптивного дизайна для работы на различных устройствах и экранах).

В результате определения функциональных требований были четко сформулированы основные возможности и функции, которые должны быть реализованы в приложении. Это позволило иметь ясное понимание того, какие задачи необходимо решать и какие инструменты и технологии использовать для достижения поставленных целей.

Реализация мессенжера

Экран с каналами (рис. 3). На этом экране пользователю предоставляется список доступных каналов, в которые он может перейти. Каждый канал представлен в виде карточки с названием и изображением. Пользователь выбирает канал, в который он хочет перейти, нажав на соответствующую карточку. После выбора канала пользователь будет перенаправлен на экран комнаты, где он сможет общаться с другими участниками, а также работать над совместным редактированием кода.

Экран аккаунта (рис. 4). На экране аккаунта пользователь может просмотреть и изменить информацию о своем профиле и настройки своего аккаунта. Здесь пользователь может загрузить аватар, обновить контактные данные, изменить пароль, а также удалить свой аккаунт.

Экран канала (рис. 5). Содержит список участников, список комнат для видеоконференций,

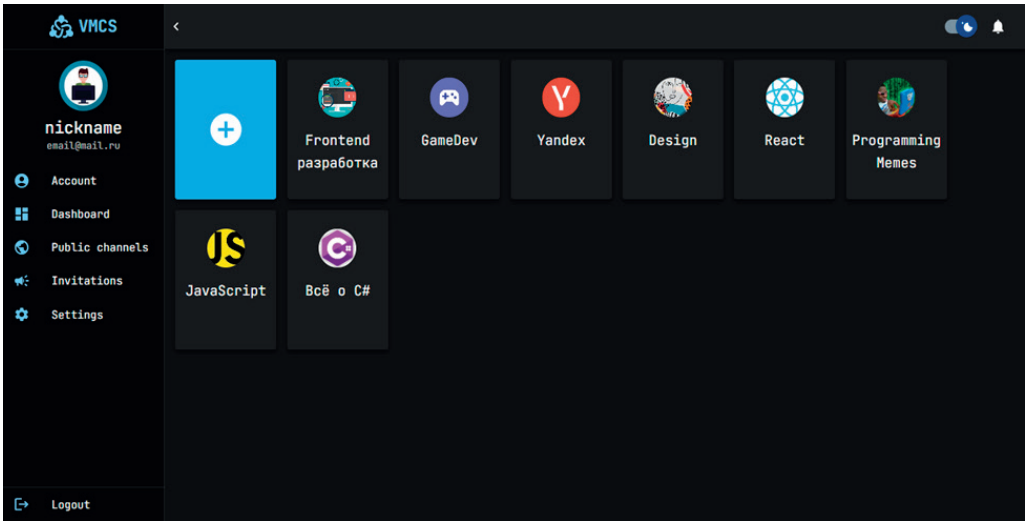

Рис. 3. Экран с каналами

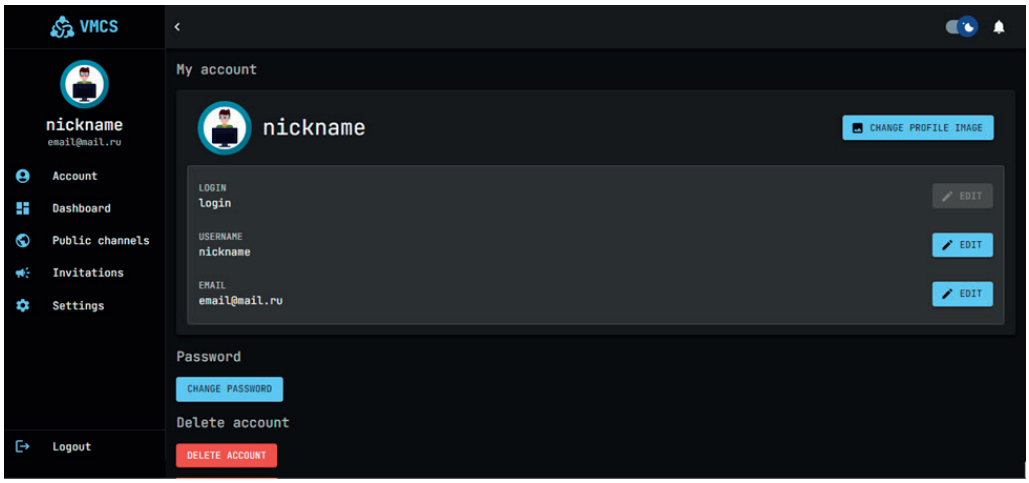

Рис. 4. Экран аккаунта

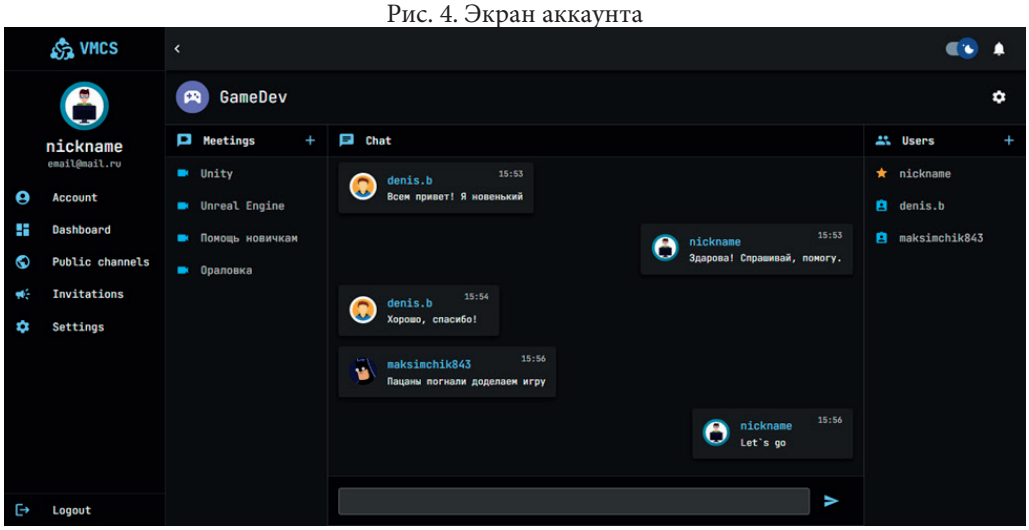

Рис. 5. Экран канала

а также чат для общения между участниками. На данном экране есть возможность присоединиться к комнате, где можно общаться по видеосвязи с другими участниками и совместно редактировать код.

Экран комнаты — это центральное место, где пользователи могут общаться по видеосвязи, обмениваться сообщениями и работать вместе над проектом. На экране присутствует кнопка для переключения между режимами видео и редактирования кода, которые позволяют пользователю выбрать режим работы.

В режиме видеоконференции (рис. 6) отображается видеопоток пользователей, участвующих в конференции.

В режиме редактирования кода (рис. 7) пользователи могут работать совместно с другими пользователями над редактированием кода. Экран в режиме редактирования кода предоставляет инструменты для отображения дерева файлов и папок, редактор кода в режиме реального времени и кнопки для сохранения, скачивания или публикации изменений в репозитории. Редактор кода имеет такие функциональности, как автоматическое форматирование, подсветка синтаксиса, функции поиска и замены, а также функции авто дополнения.

Результатом разработки пользовательского интерфейса для приложения стало создание наглядного, интуитивно понятного и функционального интерфейса, обеспечивающего удобную навигацию и взаимодействие пользователей с приложением. Каждый экран был разработан с учетом конкретных потребностей и задач пользователей, предоставляя им необходимые функции [8, 9].

Развертывание системы

Процесс развертывания приложения включает несколько шагов, связанных со сборкой, контейнеризацией и развертыванием каждого компонента.

Для контейнеризации данного проекта используется технология Docker [10]. Каждый компонент

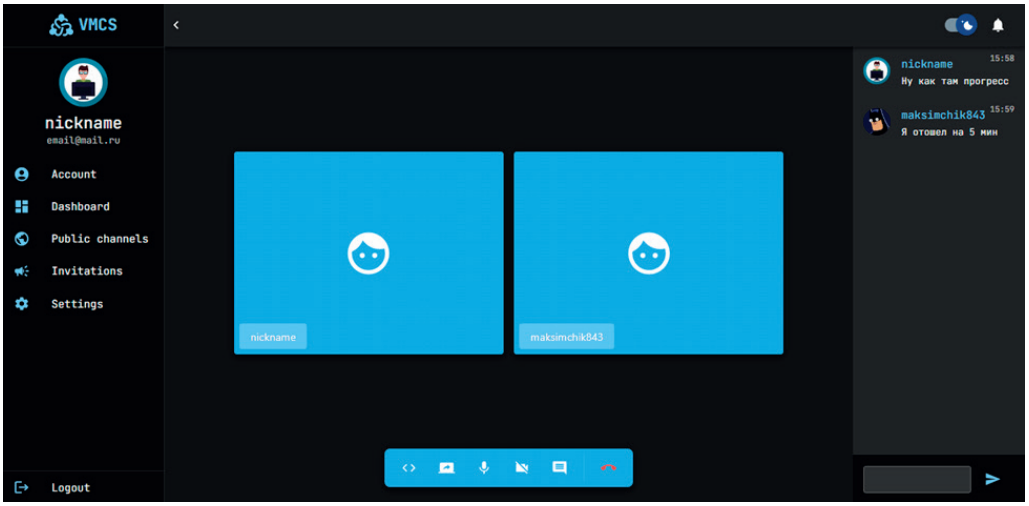

Рис. 6. Экран комнаты в режиме видеоконференции

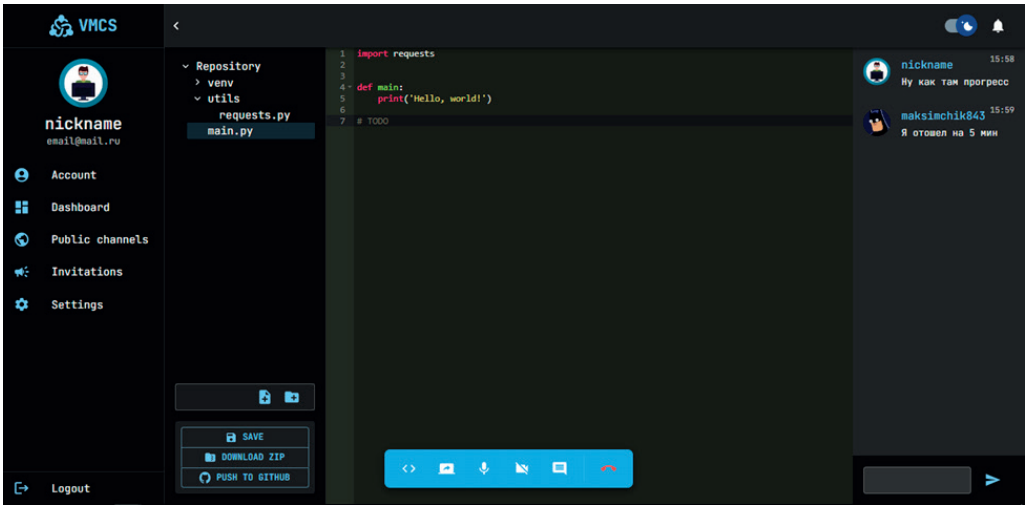

Рис. 7. Экран редактирования кода

приложения упаковывается в отдельный Docker-контейнер, который содержит все необходимые зависимости, библиотеки, переменные окружения и исполняемый код.

Создание Docker-контейнера начинается с создания Docker-образа, на основе которого будет создаваться контейнер. Для создания подобного образа необходимо иметь Docker-файл, который определяется для каждого компонента отдельно. После создания Docker-файла запускается процесс сборки образа, технология Docker будет читать Docker-файл и выполнять инструкции, чтобы создать образ (рис. 8).

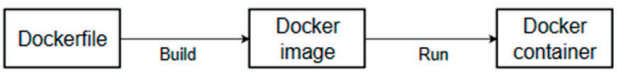

Рис. 8. Контейнеризация

В проекте используется методология CI/CD для автоматизации процесса сборки, тестирования и развертывания приложения. В данном случае используется такая технология, как Github Actions [11]. Она позволяет автоматически собирать Docker-контейнеры и разворачивать их на виртуальной машине (рис. 9).

Разрабатываемое приложение имеет возможность не только совместно работать над кодом, но и запускать данный код с помощью стороннего сервиса Judge0 [12]. Данный сервис является открытым API, на который можно отправлять код в виде base64 строки и получать результат выполнения.

Приложение было протестировано при выполнении проектного практикума и выпускной квалификационной работы бакалавра.