# VlaDO – веб-приложение для управления документами с контролем версий

## Введение

Ежедневно люди создают и обмениваются множеством электронных документов: текстовых файлов, изображений, таблиц и т.д. Часто возникает проблема **контроля версий**: при пересылке файлов по email или через мессенджеры трудно отследить, у кого самая актуальная копия документа. Совместная работа осложняется отсутствием централизованного хранилища и системы прав доступа. Существующие облачные решения (Google Drive, Dropbox и др.) частично решают проблему хранения, но либо не дают полноценного отслеживания изменений, либо навязывают ограничения в бесплатных версиях. Возникает потребность в открытом бесплатном инструменте, который позволит пользователям самостоятельно хранить и структурировать документы, отслеживать историю изменений и безопасно делиться файлами с коллегами. **Проект VlaDO** нацелен на удовлетворение этой потребности – это веб-приложение для управления документами, поддерживающее версионирование и совместный доступ к файлам.

## Цель и задачи разработки

**Цель проекта** – разработать полнофункциональное веб-приложение для хранения и совместного использования документов, которое обеспечит удобный **контроль версий** файлов, разграничение доступа и уведомление пользователей об изменениях. Приложение должно быть бесплатным и развертываемым на собственном сервере, что особенно важно для небольших команд и учебных проектов без бюджета на коммерческие системы.

Для достижения цели были решены следующие основные **задачи разработки**:

* **Регистрация и аутентификация пользователей:** реализован удобный способ создания аккаунта и входа в систему. Пароли хранятся в безопасном виде (с хэшированием), а после входа выдается **JWT-токен** для последующих запросов.
* **Управление контактами:** пользователи могут находить друг друга по имени или email и добавлять в список контактов для упрощения взаимодействия. Это подготовительный шаг для приглашения в совместные проекты.
* **Объединение в группы (комнаты):** введено понятие «комнаты» – отдельных пространств для коллективной работы. Пользователь может создать комнату под определённый проект, пригласить туда других участников и задать им роли (права доступа).
* **Хранение документов с версионированием:** реализована загрузка файлов на сервер и структурирование их по комнатам. Если в комнате загружается файл с именем, которое уже существует, система предложит сохранить его как новую версию, сохраняя при этом предыдущую копию.
* **Отслеживание истории версий:** каждая обновлённая версия документа сохраняется, и пользователи могут просматривать список предыдущих версий или восстановить нужную. Для обеспечения целостности цепочки версий при загрузке новой версии проверяется хеш предыдущей – это предотвращает потерю данных при параллельной работе.
* **Система токенов доступа:** по запросу генерируется уникальный токен (специальная ссылка), по которому сторонний пользователь может получить доступ к конкретному документу. При создании токена задаются права (только просмотр или редактирование) и срок действия, после истечения которого доступ автоматически закрывается. Владелец документа может в любой момент отозвать токен досрочно, сделав ссылку неактивной.
* **Архивация и удаление документов:** добавлена возможность “мягкого” удаления. Неактуальные файлы пользователь может переместить в архив – они скрываются из основного списка, но сохраняются на сервере с возможностью восстановления или окончательного удаления. Аналогично можно архивировать целую комнату вместе со всеми документами, если проект завершён.
* **Уведомления и лента активности:** разработана система уведомлений о всех важных событиях. Добавление нового документа, обновление версии, приглашение в комнату и другие события фиксируются как объекты активности. Пользователь видит на главной странице список последних действий, а также количество непрочитанных уведомлений; после просмотра уведомления помечаются прочитанными. Это обеспечивает прозрачность совместной работы – никто не пропустит обновление важных файлов.
* **Безопасность и разграничение доступа:** реализовано строгое разграничение прав. Только авторизованные пользователи могут вызывать защищённые методы API, и **каждая** операция проверяет права пользователя на данное действие. Например, просматривать или обновлять документ могут лишь участники соответствующей комнаты с достаточным уровнем доступа, либо пользователи с действующим токеном.
* **REST API и документация:** все функции сервера доступны через REST API, а для интеграции и тестирования предоставлена интерактивная документация Swagger (OpenAPI). Клиентская часть взаимодействует с сервером посредством обмена JSON данными по этому API.

## Основные функции и возможности системы

С точки зрения пользователя, приложение VlaDO предоставляет следующие ключевые возможности для повседневной работы:

* **Профиль пользователя:** быстрая регистрация аккаунта и вход. Пользователь может обновить свои данные профиля – имя, email, пароль. Предусмотрена проверка уникальности имени пользователя/e-mail при регистрации, а также восстановление пароля через почту (при реализации этого функционала).
* **Создание “комнат” (проектных рабочих пространств):** пользователь может создавать комнаты для разных проектов или групп документов. **Владелец комнаты** даёт ей название, устанавливает **уровень доступа по умолчанию** для новых участников (например, только чтение) и приглашает коллег.
* **Приглашение участников и роли:** в созданную комнату можно добавить участников, указав их email или выбрав из контактов. Новому участнику присваивается определённый уровень доступа – **только просмотр (Read)**, **редактирование (Edit)** или **полный доступ (Full)**, аналогичный владельцу. Права доступа можно менять или отозвать в любой момент. По умолчанию никто, кроме вас, не имеет доступа к вашим данным, пока вы его явно не предоставите.
* **Загрузка документов и управление версиями:** внутри каждой комнаты доступна загрузка файлов. Пользователь нажимает кнопку «Добавить документ», выбирает файл, и он сохраняется на сервере. Если документ с таким именем уже есть, новая загрузка автоматически сохранит **новую версию** файла, а старая версия останется в системе. Можно просматривать список документов в комнате, а для каждого документа – всю историю его версий (с указанием авторов и времени изменений). Сам файл хранится на сервере, откуда его можно скачать в любой момент.
* **Массовые операции с файлами:** для удобства реализована возможность загрузки нескольких файлов сразу (пакетная загрузка) и скачивания нескольких выбранных документов одним архивом (ZIP). Это особенно ускоряет работу, если нужно перенести сразу целый набор документов.
* **Генерация общих ссылок (токенов):** владелец документа при необходимости может быстро поделиться файлом с человеком, не зарегистрированным в системе. Для этого генерируется специальная ссылка (токен доступа) с указанием срока действия и прав (просмотр или редактирование). Получивший ссылку сможет открыть документ через браузер без отдельного аккаунта. Владелец при желании может отменить эту ссылку, чтобы закрыть доступ раньше срока.
* **Архивирование данных:** ненужные более документы пользователь может поместить в архив. Такие файлы не отображаются вместе с остальными, чтобы не мешать, но остаются доступными в разделе «Архив» на случай, если понадобятся снова. Аналогично можно архивировать целую комнату проектов, когда работа по ней завершена (при этом документы не теряются, а убираются из основного списка). При окончательном удалении система запрашивает подтверждение, чтобы избежать случайной потери данных.
* **Поиск и навигация:** реализован поиск по списку комнат – можно быстро найти нужную комнату по названию или ID, а также отфильтровать по дате создания/изменения. Это удобно, когда проектов становится много. Также есть поиск пользователей по имени или email для облегчения добавления контактов и приглашений. В перспективе можно реализовать и поиск по названиям документов или содержимому.
* **Уведомления о событиях:** приложение оповещает пользователя обо всех важных событиях. Например, если вас пригласили в новую комнату – появится уведомление, которое нужно подтвердить или отклонить. Если в вашей комнате добавлен новый файл или загружена новая версия документа – вы тоже получаете уведомление. Уведомления собираются в **ленту активности**: есть вкладка для непрочитанных (например, «5 новых событий») и вкладка для всей истории изменений. Просмотренные уведомления помечаются прочитанными. Благодаря этому каждый участник своевременно узнаёт о новых файлах, изменениях и приглашениях и не упустит важную информацию.

В совокупности эти возможности делают систему полезным инструментом для совместной работы с документами. Приложение сочетает функциональность, близкую к корпоративным системам документооборота, с простотой и доступностью, необходимыми индивидуальным пользователям и маленьким командам.

## Архитектура и техническая реализация

Проект VlaDO построен по архитектуре «клиент–сервер» с четким разделением фронтенда и бекенда. **Клиентская часть** – это одностраничное веб-приложение (**SPA**) на React. Вся логика интерфейса, навигация по страницам (например, «Мои документы», «Активность», «Профиль») и обмен данными реализованы на стороне браузера с помощью JavaScript/TypeScript. При навигации страницы не перезагружаются полностью – обновляются только нужные компоненты, что обеспечивает высокую отзывчивость интерфейса. После входа в систему SPA-клиент сохраняет выданный сервером JWT-токен и добавляет его в заголовок Authorization каждого запроса, чтобы сервер мог идентифицировать пользователя. Обмен данными между фронтом и бекендом происходит по HTTP в формате JSON через RESTful API.

**Серверная часть** реализована на платформе ASP.NET Core (язык C#) и предоставляет REST API для всех функций приложения. В сервер встроена многослойная архитектура:

* **Контроллеры (Web API)** – принимают входящие HTTP-запросы от клиента. В приложении реализованы различные контроллеры по сферам ответственности: AuthController (регистрация и логин), RoomController (операции с комнатами), DocumentController (загрузка и управление документами), ActivityController (уведомления и лента активности), ContactsController (работа с контактами) и т.д. Каждый контроллер определяет маршруты URL и типы запросов (GET, POST, PUT/PATCH, DELETE) для соответствующих операций, и обрабатывает их минимальной логикой – в основном, вызывает соответствующие методы сервисного слоя и формирует ответ. Также контроллеры проверяют наличие валидного JWT-токена в запросе, чтобы убедиться, что пользователь авторизован.
* **Слой сервисов (бизнес-логика)** – основной код приложения, где реализованы правила и последовательность операций. Для каждой крупной функциональности есть соответствующий сервис: AuthService (регистрация/вход), RoomService, DocumentService, ActivityService, ContactsService и пр. Сервисы инкапсулируют логику: например, метод Register в AuthService проверяет корректность данных, хэширует пароль и создает нового пользователя; метод AddDocument в DocumentService сохраняет файл и создаёт запись первой версии; метод AddVersion проверяет хеш предыдущей версии и сохраняет новую версию документа, фиксируя событие активности. Сервисы не зависят от деталей веб-слоя – они могут вызываться из любого интерфейса (что упрощает тестирование).
* **Слой доступа к данным (репозитории)** – обеспечивает сохранение и чтение данных из базы данных. Используется ORM **Entity Framework Core** для работы с реляционной SQL-базой (в тестовой среде – встроенная SQLite). Определены репозитории для каждой сущности: IUserRepository, IRoomRepository, IDocumentRepository, IActivityRepository и т.д. Репозитории инкапсулируют запросы к базе (LINQ-запросы к контексту EF) и возвращают данные сервисам.

Благодаря такому разделению, каждый компонент системы имеет свою область ответственности (**Single Responsibility Principle** из SOLID), и модули слабо связаны друг с другом. Связь между контроллерами, сервисами и репозиториями управляется через механизм **внедрения зависимостей (DI)**, предоставляемый .NET Core. Это означает, что при старте приложения все сервисы и репозитории регистрируются в специальном контейнере, и затем нужные экземпляры автоматически передаются в конструкторы классов. DI реализует принцип инверсии управления и облегчает замену реализаций (например, можно подставить тестовую реализацию репозитория для модульного тестирования).

Диаграмма архитектуры проекта иллюстрирует взаимодействие этих компонентов. Контроллеры обмениваются данными с клиентом через DTO (объекты передачи данных), вызывают методы сервисов, которые обращаются к репозиториям для работы с базой. Для аутентификации используется JWT: при логине AuthController вызывает AuthService, который проверяет учетные данные и генерирует токен с цифровой подписью; этот токен затем используется клиентом для доступа к защищённым маршрутам. В систему встроены и вспомогательные компоненты, например, **Swagger UI** – сгенерированная документация API, позволяющая удобно просматривать и тестировать все доступные методы. Также реализована базовая отправка почтовых уведомлений (например, при сбросе пароля) через внешний SMTP-сервис.

**Технологический стек:** таким образом, на бекенде используются C# (ASP.NET Core) + Entity Framework Core, JSON Web Token для авторизации, Swagger для документации, xUnit для модульного тестирования. На фронтенде – React + Vite (современный сборщик) + библиотека компонентов Ant Design (предположительно, судя по интерфейсу) для быстрой разработки UI. Весь код построен с учетом современных подходов и принципов (REST, SPA, SOLID), что сделало проект хорошей практикой в освоении полного цикла веб-разработки.

## Демонстрация работы приложения

Теперь продемонстрируем работу приложения VlaDO и его основных модулей интерфейса. После успешного входа пользователь попадает на главную страницу – **дашборд**, где собрана ключевая информация.

Интерфейс главной страницы приложения (дашборд). Слева – список «Мои комнаты» пользователя, по центру – лента последних активностей, справа – список недавно изменённых документов.

На скриншоте выше показан дашборд приложения. В левой колонке отображаются комнаты пользователя (рабочие пространства по проектам). Здесь, например, у пользователя есть комнаты под названиями «312», «Владос» и «Тест». Над списком находится кнопка **«+ Комната»** для создания нового пространства. По центру страницы видна панель **«Последние активности»** – здесь перечислены свежие события: добавление документов, запросы на добавление в контакты, удаление файлов и пр., с указанием даты【7†】. Справа расположен список **«Последние документы»**, где показаны документы, недавно изменённые во всех комнатах пользователя (для удобства быстрого доступа к ним). В шапке интерфейса доступна навигация: справа вверху отображается имя вошедшего пользователя («Test»), иконки профиля и контактов, значок **колокольчика для уведомлений** (где появляется счётчик, если есть новые события) и кнопка «Выйти» для завершения сеанса. Таким образом, дашборд позволяет сразу увидеть общую картину: какие проекты у вас есть, что нового произошло и с какими файлами вы недавно работали.

Перейдем к управлению документами в одной из комнат. Пользователь может открыть раздел «Комнаты» и выбрать нужную – например, комнату **«312»**. На скриншоте ниже отображается интерфейс комнаты с документами и участниками.

Рабочее пространство комнаты «312». На фоне – таблица документов в комнате, на переднем плане – окно управления участниками комнаты с указанием прав доступа.

Перед нами **менеджер комнаты**. В данной комнате хранится список документов (в таблице видны, например, файл «Дайджест сентября 2024.pdf» версии v1, создан 27.06.2025 пользователем Test, и файл descriptor.json v1). Справа (не на скриншоте) есть кнопка **«+ Документ»** – через нее участники могут загружать новые файлы в эту комнату. Если загрузить файл с именем, которое уже присутствует, система не заменит старый файл, а создаст для документа новую версию. Таким образом, всегда сохраняется история изменений.

На переднем плане показано окно **«Пользователи комнаты “312”»**, где владелец управляет участниками проекта. Указан **список участников**: в данном случае это два пользователя – [*v.borodin07@list.ru*](mailto:v.borodin07@list.ru) и Test. Здесь же задается **уровень доступа по умолчанию** (сейчас установлен как «Только чтение»). Это значит, что новые приглашённые будут иметь минимальные права, пока владелец не повысит их при необходимости. Приглашение нового участника происходит просто: достаточно ввести email пользователя (должен быть зарегистрирован в системе) – ему придет уведомление о приглашении в комнату, которое он может принять. После принятия он появится в этом списке. Владелец или администратор комнаты может менять роль участника (например, дать права редактирования или полный доступ) либо удалить участника из комнаты. Таким образом, **совместная работа** организована гибко: можно в любой момент включить нового коллегу в проект или ограничить доступ текущему участнику.

Рассмотрим теперь, как система осуществляет **контроль версий** документов. Допустим, в комнату уже загружен некоторый файл – например, исходный код ProjectSchema.cs версии 1. Если другой пользователь параллельно загрузит обновлённый файл с таким же именем, то у документа появится версия 2. При этом VlaDO не “затирает” предыдущий вариант, а сохраняет обе ветки изменений. В результате формируется **дерево версий**, которое можно просмотреть в специальном окне истории.

Окно «Дерево версий» для документа, иллюстрирующее разветвление историй при параллельных обновлениях. Каждая версия помечена своим номером и содержит информацию о файле и времени изменения.

На скриншоте выше представлен фрагмент окна **«Дерево версий»** для одного из документов. Видно, что помимо последовательных версий v1, v2, существуют ответвления (например, версия **v2-1**, **v3-1** и их дочерние версии v3-1.1, v3-1.2 и т.д.). Такое ветвление возникает, когда несколько изменений вносятся параллельно разными пользователями или в разной последовательности. **Система сохраняет все разветвленные версии**, чтобы ни одно изменение не было утеряно: каждый ответвленный вариант файла доступен и пронумерован особым образом. Пользователь может изучить, какая версия была создана от какой (стрелки и вложенность наглядно показывают происхождение версий), просмотреть содержимое любого старого файла и при необходимости восстановить предыдущую версию как основную. Благодаря такому подходу реализован полноценный механизм версионирования, аналогичный системам контроля версий, но в более удобной для рядового пользователя форме.

Помимо управления версиями, в приложении реализованы и другие полезные функции, которые можно увидеть в работе. Например, система отправляет уведомления при событиях: если ваш коллега загрузил новый документ или обновил файл в общей комнате, вы сразу увидите это в ленте активности на главной странице и в списке уведомлений (значок колокольчика). Приглашения в комнаты тоже появляются как уведомления, требующие действия. Также доступен **поиск** – вы можете быстро найти нужную комнату по названию через строку поиска. А чтобы поделиться документом с внешним пользователем, достаточно сгенерировать токен-ссылку через интерфейс (например, через меню документа) – система покажет вам готовый URL для отправки коллегам. В демонстрации мы ограничились основными модулями: созданием комнат, загрузкой документов, просмотром истории версий и уведомлениями, – этого достаточно, чтобы увидеть, как приложение облегчает совместную работу с файлами. Каждый шаг, от добавления пользователя до обновления документа, прозрачно фиксируется и контролируется системой.

## Итоги и перспективы развития

**Итоги проекта:** в результате разработки получено полноценное веб-приложение **VlaDO** для управления документами, решающее обозначенную проблему контроля версий и совместного доступа. Система предоставляет централизованное хранилище документов с разделением по проектным комнатам и гибким разграничением доступа между пользователями. Реализован полноценный контроль версий: все изменения файлов фиксируются, предыдущие версии сохраняются и доступны для просмотра или восстановления. Пользователи могут безопасно редактировать документы, не опасаясь потери данных – при ошибке всегда можно откатиться на нужную итерацию. Внедрен механизм уведомлений, повышающий прозрачность работы в команде: участники своевременно получают информацию о новых файлах, обновлениях версий, приглашениях в проекты и других событиях, что позволяет координировать работу без постоянных переписок. Большое внимание уделено безопасности: авторизация построена на JWT, все данные передаются по защищенному протоколу, пароли хранятся в хэшированном виде. На всех уровнях проверяются права доступа – никто посторонний не прочитает и не изменит документ без соответствующих прав. Архитектура приложения получилась гибкой и расширяемой: соблюдение принципов SOLID и многослойности упростило отладку и дальнейшее сопровождение кода. Проект использует только бесплатные технологии с открытой лицензией (ASP.NET Core, React, SQLite и др.), исходный код при необходимости может быть выложен в открытый репозиторий – это подтверждает достигнутую цель создать **бесплатное и открытое** решение, которым можно свободно пользоваться и при желании модифицировать.

Все запланированные функции успешно реализованы, и приложение полностью соответствует поставленным требованиям. VlaDO уже сейчас позволяет небольшим рабочим группам эффективно организовать коллективную работу с документами: документы всегда в порядке, всегда доступна последняя версия, есть история изменений, и можно безопасно делиться файлами с нужными людьми. Проект продемонстрировал целый стек современных технологий (frontend, backend, API, база данных, безопасность) и принципы разработки в действии, а автор получил ценную практику полного цикла создания веб-приложения.

**Перспективы развития:** хотя основная цель достигнута, у системы есть потенциал для дальнейшего улучшения и расширения функциональности. В будущем можно реализовать:

* **Более тонкие настройки доступа:** сейчас предусмотрены три роли (чтение, редактирование, полный доступ). Можно добавить промежуточные уровни или особые права – например, ограничить видимость конкретных документов для отдельных участников, ввести роль наблюдателя (просмотр без скачивания) или временный доступ, который автоматически отзывается по истечении заданной даты.
* **Улучшение интерфейса и UX:** расширить возможности фронтенда для удобства пользователей. Например, сделать предварительный просмотр документов (PDF, изображения) прямо в браузере без скачивания, реализовать наглядное сравнение версий текстовых файлов (подсветку изменений между двумя версиями). Полезно добавить визуальные индикаторы новых изменений и комментариев. Также стоит внедрить отображение прогресса при загрузке больших файлов, возможность отмены загрузки, поддержку перетаскивания (drag & drop) папок целиком вместе со структурой (что упростит перенос больших архивов данных).
* **Система комментариев и обсуждений:** для повышения интерактивности можно позволить оставлять комментарии к документам или конкретным версиям (например, обсуждать внесенные правки). Появится новая сущность «Комментарий» и соответствующий интерфейс, что сделает работу над документами ближе к привычным офисным пакетам.
* **Интеграция с внешними сервисами:** сейчас уведомления о событиях доступны только внутри приложения. В перспективе можно добавить отправку email-уведомлений на почту пользователей (например, о приглашениях или важных обновлениях). Для этого интегрируется почтовый SMTP-сервер или API вроде SendGrid. Также можно реализовать экспорт документов в сторонние облака (напр. в Google Drive) или синхронизацию с локальным диском.
* **Мобильный клиент:** создание мобильного приложения для смартфонов на базе React Native или Flutter расширит аудиторию VlaDO. Благодаря уже существующему REST API мобильный фронтенд сможет использовать те же серверные методы без переработки бекенда. Пользователи получат доступ к документам и уведомлениям в любом месте со своих телефонов.
* **Глубокое тестирование и безопасность:** по мере развития проекта стоит увеличить покрытие кода автотестами, провести аудит безопасности (защититься от SQL-инъекций, XSS-атак, внедрить Content Security Policy для SPA, добавить капчу или задержки на форму входа против перебора паролей). Эти меры повысят надежность продукта для промышленного применения.

Перечисленные направления показывают, что у VlaDO есть хорошие возможности для эволюции. Проект уже сейчас решает поставленные задачи и может быть полезен своей целевой аудитории. В дальнейшем при доработке он способен вырасти в полноценный продукт, конкурентоспособный в нише систем документооборота для небольших команд. Для учебного же проекта данный результат демонстрирует успешное применение современных технологий и подходов. **Проект завершён, цели достигнуты, и VlaDO готов приносить пользу пользователям, нуждающимся в простом и надёжном инструменте управления документами с контролем версий.**

Спасибо за внимание.