РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ЖИЗНЕННЫМ ЦИКЛОМ ДИЗАЙН-ТОКЕНОВ

Лазебная Е.А., старший преподаватель,
Присухин Б.Р., студент,
БГТУ им. В.Г. Шухова, г. Белгород, Россия

Аннотация: В статье рассматривается разработка веб-ориентированной информационной системы для управления жизненным циклом дизайн-токенов. Описана клиент-серверная архитектура системы, включающая плагин для Figma на TypeScript, серверную часть на Node.js/Express с REST API и клиентское веб-приложение на React. Представлены алгоритмы синхронизации дизайн-токенов в рамках единой транзакции базы данных с фиксацией истории изменений и мягким удалением, а также алгоритм создания неизменяемых JSON-снапшотов версий коллекций. Описаны основные интерфейсы системы: страница управления токенами с древовидной навигацией и селектором версий, страница истории изменений с фильтрацией по дате и страница авторизации с JWT-аутентификацией.

Ключевые слова: дизайн-токены, Figma, автоматизация, клиент-серверная архитектура, Node.JS, React, версионирование, история изменений.

С ростом сложности цифровых продуктов компании столкнулись с проблемой масштабирования визуальных языков. Первыми системный подход к этой проблеме предложили в Salesforce, внедрив концепцию «дизайн-токенов» как абстрактного слоя между визуальным исполнением и его реализацией в коде. В 2018 году крупнейшие игроки индустрии совместно с сообществом W3C [1] инициировали работу над стандартизацией форматов обмена дизайн-токенами, а в 2022 году Figma внедрила функциональность Variables, позволяющую создавать токены непосредственно внутри макетов. Однако их жизненный цикл — от создания в дизайнерских инструментах до

использования в кодовой базе — часто не автоматизирован, что приводит к ручному, подверженному ошибкам переносу данных, рассогласованию интерфейсов и замедлению разработки.

Целью работы является автоматизация и централизация полного цикла работы с дизайн-токенами для обеспечения единого источника истины, контроля версий и интеграции в процессы разработки.

Информационная система управления жизненным циклом дизайн-токенов состоит из двух основных компонентов: серверной части и клиентского веб-приложения. Дополнительно, в рамках архитектуры учитывается внешний по отношению к системе плагин для Figma, который обеспечивает передачу исходных данных.

Серверная часть реализована на Node.js с использованием фреймворка Express и выполняет приём и обработку данных от плагина Figma через HTTP-запросы, хранение коллекций, переменных и истории изменений в MySQL, предоставление REST API для веб-клиента, управление версиями коллекций, аутентификацию и авторизацию пользователей, а также генерацию файлов целевых форматов (CSS, JSON, SCSS) и сборку NPM-пакета с дизайн-токенами. Клиентское веб-приложение на React обеспечивает управление версиями коллекций, просмотр истории изменений с фильтрацией по дате, экспорт токенов, а также интерфейс для регистрации, авторизации. Внешний плагин для Figma собирает переменные и коллекции через API [2], формирует JSON-манифест и отправляет данные на сервер через HTTP POST запрос. Взаимодействие между компонентами строится следующим образом: веб-клиент обращается к серверу через REST API и получает данные в формате JSON, сервер взаимодействует с MySQL через пул соединений, плагин Figma отправляет данные на сервер также через HTTP POST запросы.

Предложенная архитектура обеспечивает масштабируемость за счёт независимого развития серверной и клиентской частей, гибкость при замене компонентов, лёгкую интеграцию с Figma через внешний плагин и расширяемость для добавления новых форматов экспорта.

Плагин для Figma является ключевым компонентом, обеспечивающим импорт дизайн-токенов в систему (Рис. 1). Он разработан с использованием официального Figma Plugin API на языке программирования TypeScript. Он следует модели разделения ответственности и состоит из двух модулей. Модуль пользовательского интерфейса представляет собой HTML-окно с кнопкой инициации экспорта. Его логика заключается в обработке действия пользователя, приёме структурированных данных от ядра плагина и отправке их на REST API серверной части через fetch-запрос. Модуль бизнес-логики — скрипт, выполняемый в среде Figma. Он по запросу от UI вызывает асинхронные методы Figma API getLocalVariablesAsync() и getLocalVariableCollectionsAsync() для получения полного снапшота переменных текущего файла, преобразует сырые данные в структурированный JSON-объект (сохраняя идентификаторы, имена, типы, значения по режимам и привязку к коллекциям) и отправляет результат обратно в UI-модуль через механизм postMessage.

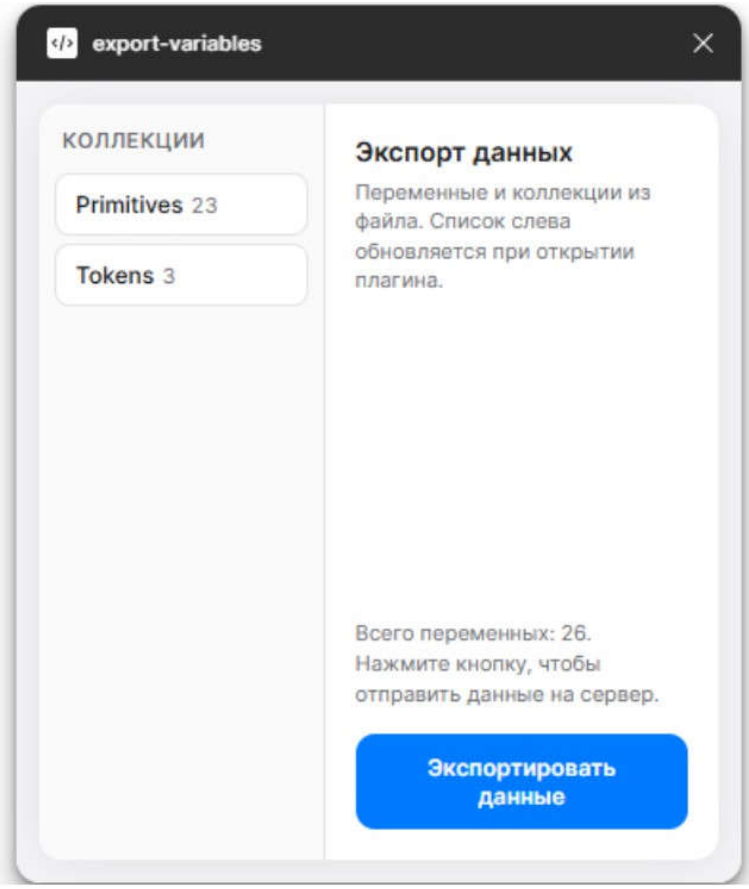

Рис. 1 Интерфейс плагина Figma

При получении данных от плагина Figma сервер выполняет алгоритм сравнения и синхронизации в рамках единой транзакции базы данных. Сначала выполняется синхронизация коллекций: для каждой входящей коллекции выполняется UPSERT в таблицу collections с обновлением названия и списка режимов. Затем для каждой коллекции из таблицы variables загружаются активные (не удалённые) переменные, и на основе сравнения идентификаторов вычисляются три подмножества: новые, изменённые и удалённые. Для новых переменных в таблицу variables добавляются записи, а в таблицу variable_history заносится событие типа created с полным снимком значений. Изменённые переменные обновляются с фиксацией события updated. Удалённые переменные физически не удаляются — для них устанавливается флаг is_deleted = TRUE и время удаления, а в историю добавляется событие deleted. После обработки всех переменных транзакция фиксируется. При любой ошибке выполняется полный откат, возвращая базу данных в исходное состояние.

Алгоритм создания версий коллекций обеспечивает фиксацию неизменяемого снимка состояния всех переменных коллекции. Процесс выполняется в рамках транзакции: из запроса получают идентификатор коллекции collection_id и метаданные версии — имя (version_name), тег (version_tag) и описание (description). Затем формируется снапшот: SQL-запрос выбирает все записи из таблицы variables, принадлежащие указанной коллекции, в их текущем состоянии, а результат сериализуется в JSON-строку. Далее в таблицу collection_versions добавляется запись, содержащая ссылку на коллекцию, метаданные версии, JSON-снапшот в поле snapshot_data и временную метку создания. После этого транзакция фиксируется, и клиенту возвращаются данные о созданной версии. Данные в поле snapshot_data остаются неизменными даже при последующем редактировании или удалении исходных переменных в таблице variables, что обеспечивает точное восстановление и сравнение состояний дизайн-системы в разные моменты времени.

Веб-клиент системы представляет собой одностраничное приложение на React с маршрутизацией через React Router. Приложение имеет два основных маршрута: /tokens для управления токенами и /history для просмотра истории изменений, а также страницу авторизации.

Страница токенов является центральным интерфейсом системы (Рис. 2). Левая панель содержит список коллекций, импортированных из Figma, и дерево групп токенов, построенное на основе разделителя / в именах (например, color/primary/500). Центральная область отображает таблицу токенов выбранной коллекции или группы с визуальным представлением значений (цветовое пятно для COLOR, текст для FLOAT, STRING, BOOLEAN). Селектор версий позволяет переключаться между текущим состоянием коллекции и её сохранёнными версиями. Правая панель предоставляет интерфейс для создания новых версий — авторизованный пользователь может ввести имя версии и создать снимок текущего состояния.

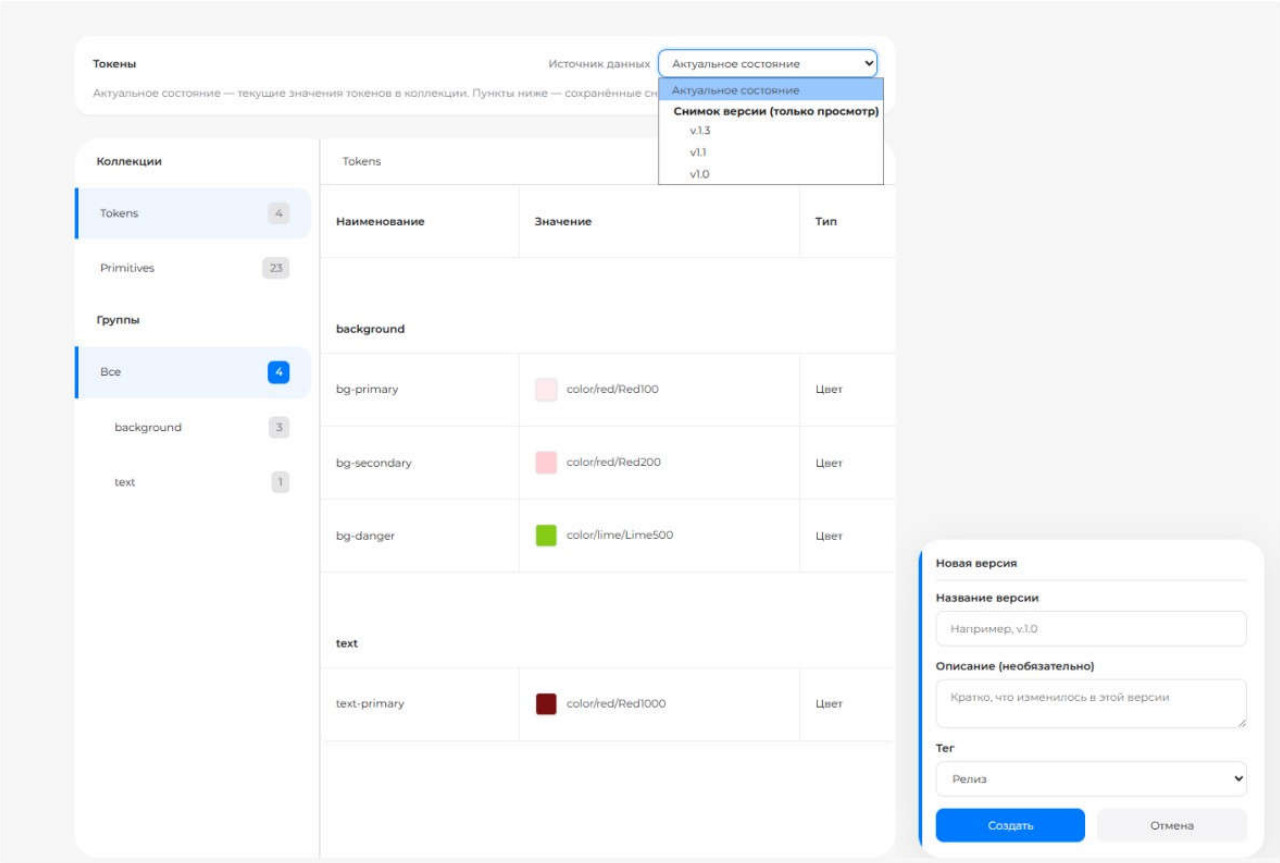

Рис. 2 Главная страница системы

Страница истории позволяет просматривать изменения дизайн-токенов с фильтрацией по временному периоду (Рис. 3).

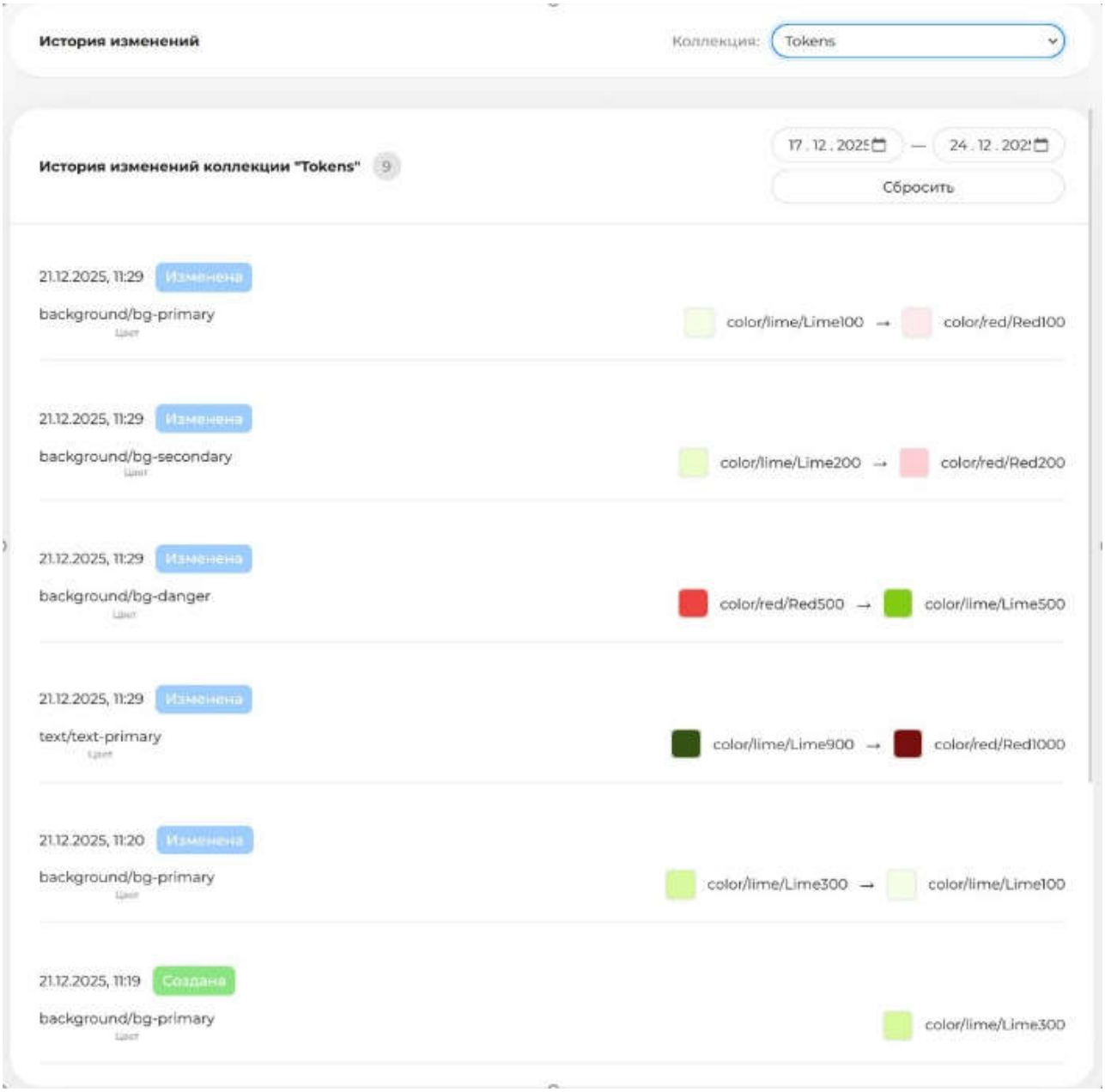

Рис. 3 Страница просмотра истории изменений

Интерфейс включает селектор коллекции, панель фильтрации с календарными селекторами начальной и конечной даты, а также ленту истории. Для каждой записи показываются дата и время изменения, тип изменения (создание, обновление, удаление), имя переменной и визуальное сравнение старого и нового значений.

Страница экспорта. Для передачи дизайн-токенов в кодовую базу разработан интерфейс экспорта, позволяющий выбрать коллекцию и конкретную версию (релизную или черновую). При выборе черновика отображается предупреждение, что версия не предназначена для промышленного использования. Реализован экспорт токенов в форматы CSS, JSON и SCSS с учётом ссылок Figma (VARIABLE_ALIAS): семантические токены, ссылающиеся на примитивы из других коллекций, в экспорте отображаются как ссылки на имена токенов (var(--…) в CSS, $… в SCSS). Для CSS добавлена обёртка :root { … }, для числовых токенов автоматически проставляются единицы измерения px. Добавлена фильтрация по типам переменных, приведённым к типам Figma: color, number, string, boolean. Реализованы предпросмотр токенов перед экспортом и форма выгрузки файла с возможностью ввода имени файла и отмены операции (Рис. 4).

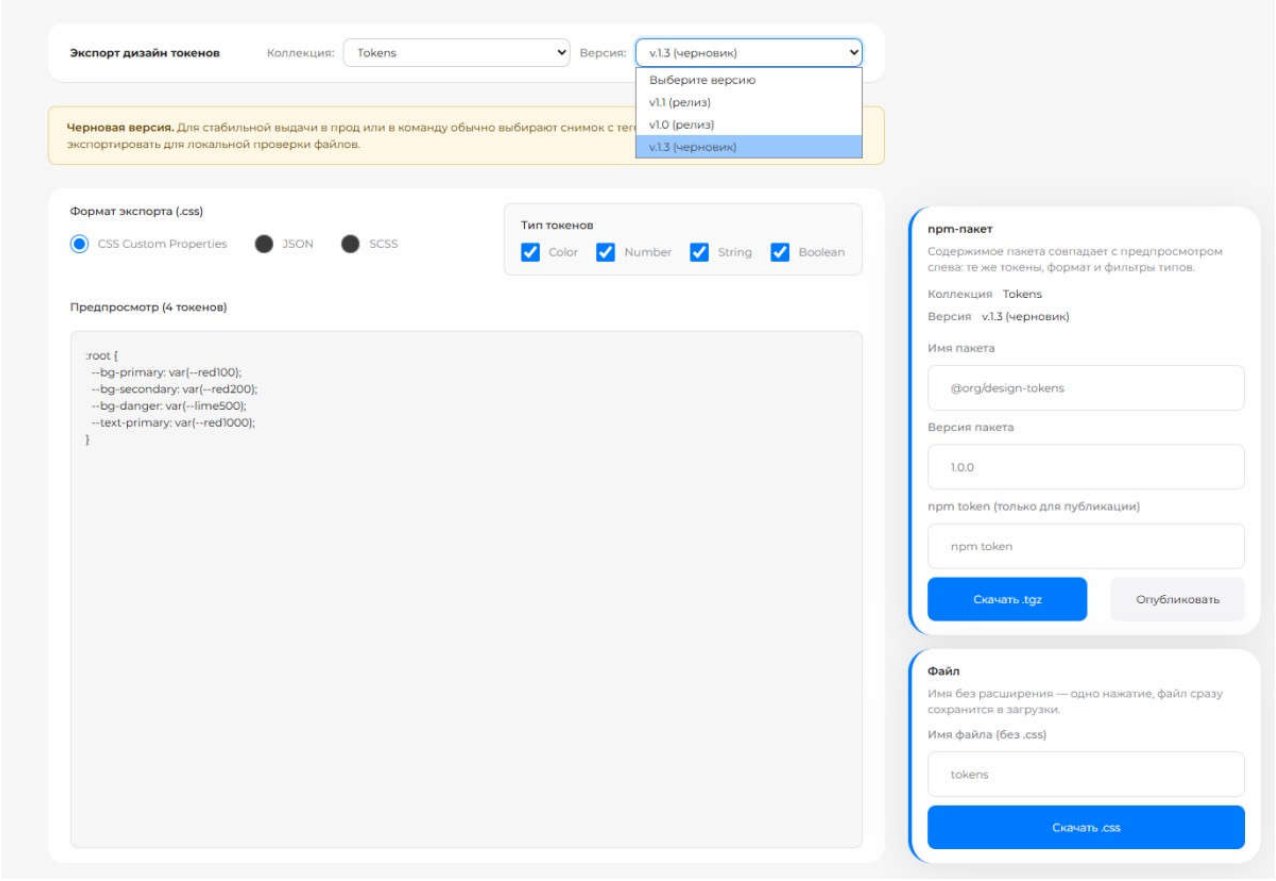

Рис. 4 Страница экспорта с выбором версии и настройками формата

Для удобства интеграции в JavaScript-проекты заложен функционал сборки npm-пакета с токенами. На сервере реализован endpoint, формирующий структуру пакета (package.json, entry-файл с токенами) и выполняющий команду npm pack. В интерфейсе добавлен отдельный блок, где разработчик может указать имя пакета, версию и npm-токен, а затем скачать архив .tgz или опубликовать пакет в реестре. Это позволяет устанавливать актуальные дизайн-токены через стандартный npm install.

Страница авторизации реализует вход в существующий аккаунт и регистрацию нового пользователя через единую форму, динамически меняющую режим (Рис. 5). Выполняется клиентская валидация: обязательность полей, длина пароля и его подтверждение. При успешной отправке POST-запроса на соответствующий эндпоинт сервера полученный JWT-токен сохраняется в localStorage для последующей авторизации запросов, после чего пользователь перенаправляется на главную страницу.

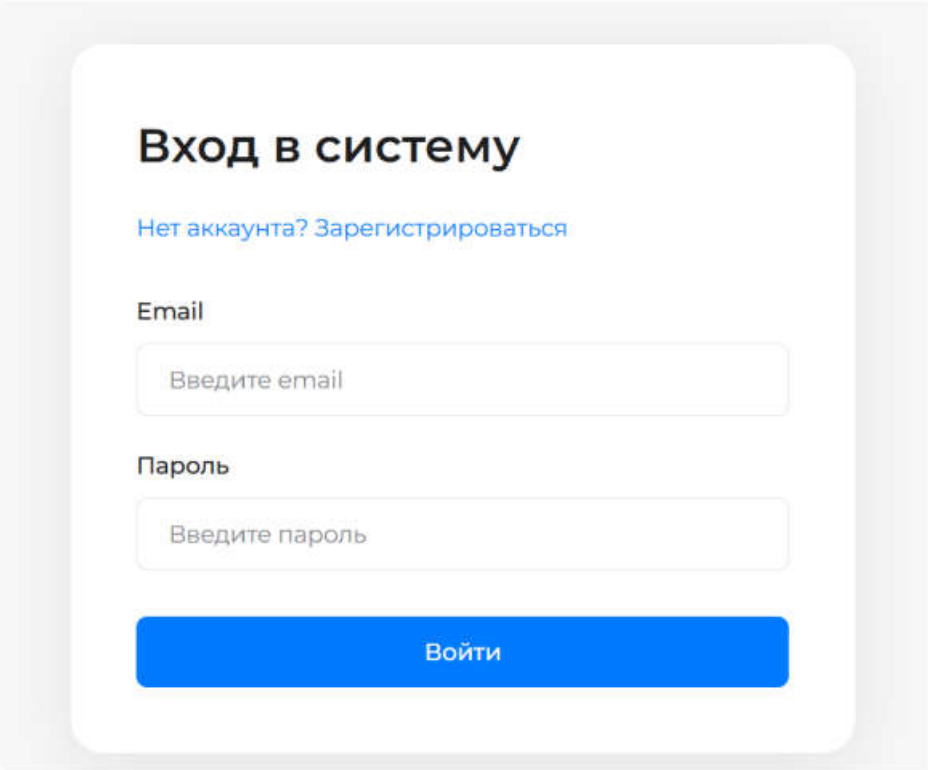

Рис. 5 Страница авторизации

Разработанная веб-ориентированная система управления жизненным циклом дизайн-токенов обеспечивает автоматизированное взаимодействие между дизайнерами и разработчиками. Система включает плагин для Figma для сбора переменных, серверную часть на Node.js с REST API, историей изменений и версионированием коллекций, а также клиентское приложение на React для просмотра токенов, управления версиями и фильтрации истории. Это исключает ручной перенос данных, снижает количество ошибок, обеспечивает единый источник истины для визуального языка продукта и фиксирует полную историю изменений. Разработанная система может быть использована в реальных процессах разработки цифровых продуктов.

Литература

1. Design Tokens Technical Reports 2025.10 [Электронный ресурс] / Design Tokens Community Group. – 2025. – Режим доступа: https://www.designtokens.org/tr/2025.10/ (дата обращения: 06.05.2026).

2. Figma. Plugin API Reference [Электронный ресурс] // Figma Developers. – 2026. – Режим доступа: https://developers.figma.com/docs/plugins/api-reference/ (дата обращения: 06.05.2026).