Отчёт по проекту Shared IDE

Исхаков Тимур

22 июня 2016 г.

1 Описание проекта

IDE для различных целей (в основном для языков программирования) с возможностью коллективного редактирования.

Текущая функциональность:

- Файловое хранилище:
  - Авторизация, регистраци;
  - Создание, удаление, экспорт файлов;
  - Автоматически обновляемый список файлов;
  - Права доступа к файлам: режимы просмотра и редактирования.
- Коллективное редактирование документов
  - Синхронизация изменений;
  - Сохранение изменений на сервере;
  - Отображение нескольких курсоров;
  - Экспорт в случае разрыва соединения;
  - Удобный редактор, поддерживающий несколько языков программирования.

Стек технологий:

- Back-end: синхронный Django, асинхронный Channels;
- Front-end: React, асинхронный WebSocket.

2 Скриншоты интерфейса

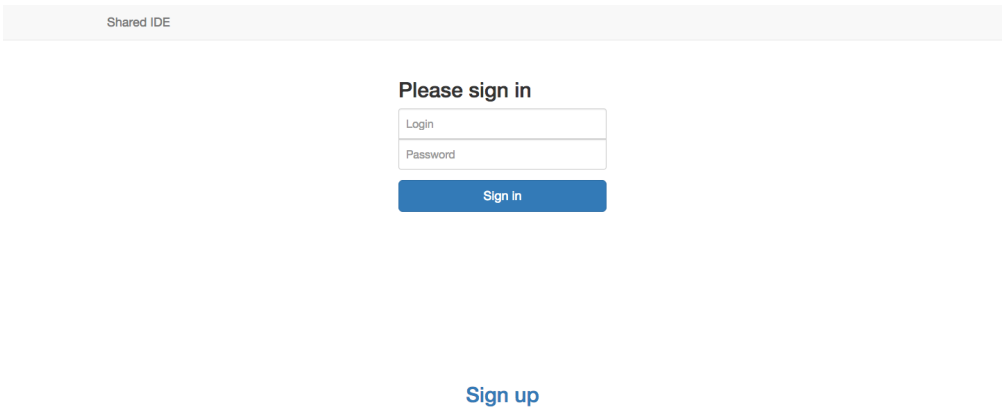

Рис. 1: Авторизация

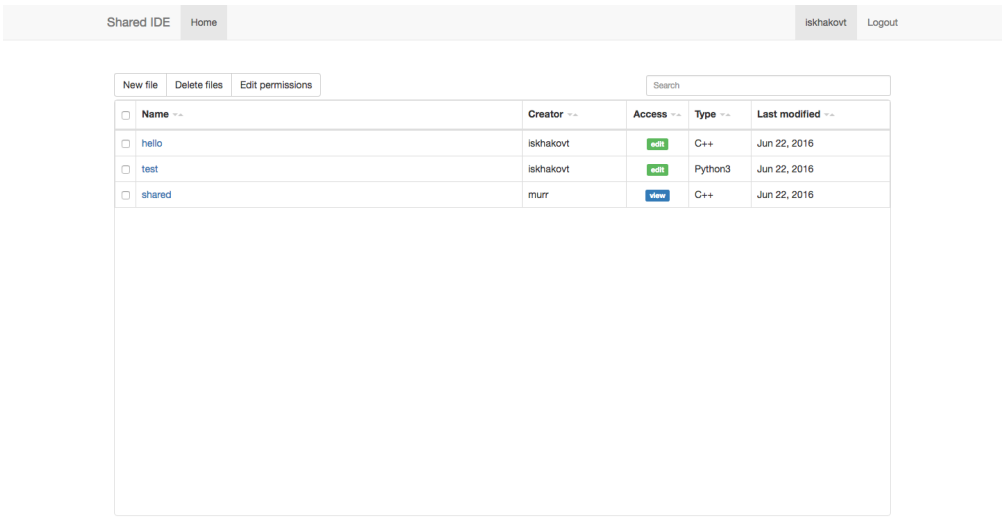

Рис. 2: Список файлов

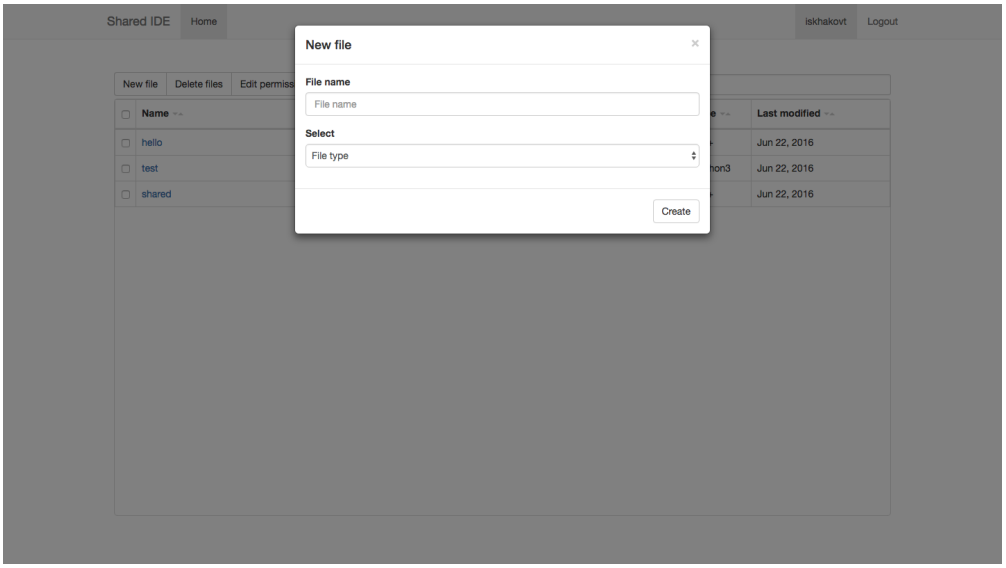

Рис. 3: Создание нового файла

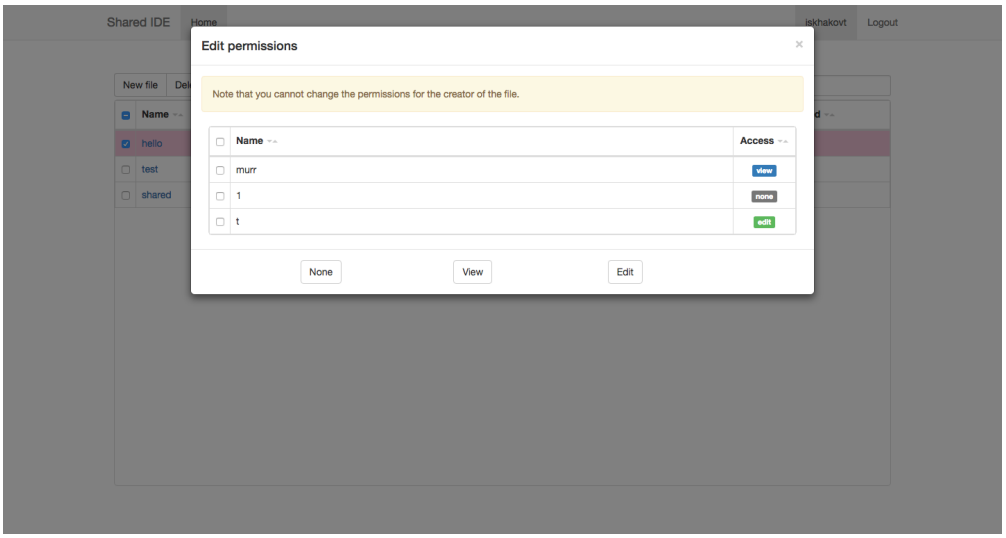

Рис. 4: Редактирование прав файлов

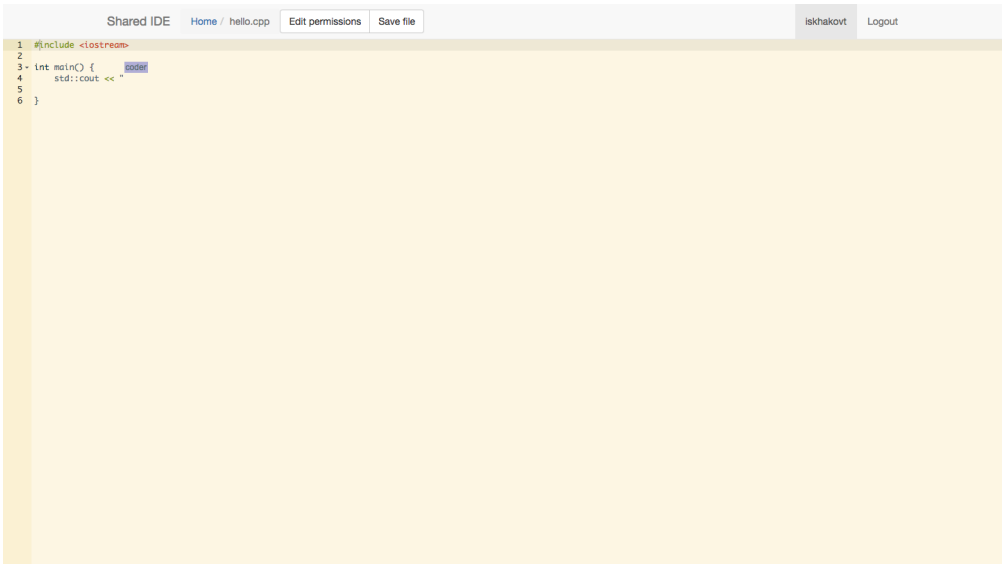

Рис. 5: Редактор

3 Текущая архитектура

Для синхронизации изменений работы вместо GET и POST запросов используется протокол WebSocket.

Django — синхронный фреймворк, поэтому асинхронный WebSocket нельзя использовать с Django нативно. Для осуществления работы WebSocket используется Redis сервер, который подклчается в Django, подменяя её wsgi-приложение.

Для создания асинхронных задач из Django (обработчика сообщений на стороне сервера, который будет получать изменения клиентов через WebSocket) изначально использовался Celery, затем был заменён на Channels как более узкоспециализированный вариант.

Изменения отправляются broadcast-сообщение подключившимся пользователям с правами редактирования, также получаются сервером. Сервер дополнительно пересылает сообщение в канал для подключившихся пользователей с правами просмотра. Также посылаются специальные сообщения в случае подключения и отключения пользователя.

Сторона клиента реализована с помощью фреймворка React, применение изменений (добавление, удаление) реализовано самостоятельно. Положение курсора также меняется в зависимости от изменений.

4 Сборка JS, JSX

В проекте используются JS по стандарту ECMAScript 6 на стороне клиента. В файлах имеются зависимости: «import», «export».

К сожалению, ECMAScript 6 не поддерживается современными браузерами, поэтому файлы, используемы на стороне клиента, собираются в момент сборки в монолитные JS-файлы:

1. babel: раскрытие ECMAScript 6;
2. browserify: разрешение зависимостей.

Собранные файлы из `./src/js/` оказываются в `./static/`.

Конфигурация сборки находится в файле `./.babelrc`, необходимые пакеты — в файле `./package.json`. Python-зависимости находятся в файле `requirements.txt`.

Сборка занимает порядка минуты, поэтому она кешируется: хеши собранных файлов хранятся в файле `./js_build_hash.json`.

5 Установка

1. Установка пакетов:

```
pip3 install -r requirements.txt
npm install
```

2. Настойка Django, сборка базы данных:

```
./manage.py makemigrations
./manage.py migrate
./manage.py createsuperuser
```

6 Запуск

1. Запуск Redis сервера:

```
redis-server
```

2. Запуск Django сервера (вместе с запуском Channels):

```
./manage.py runserver
```