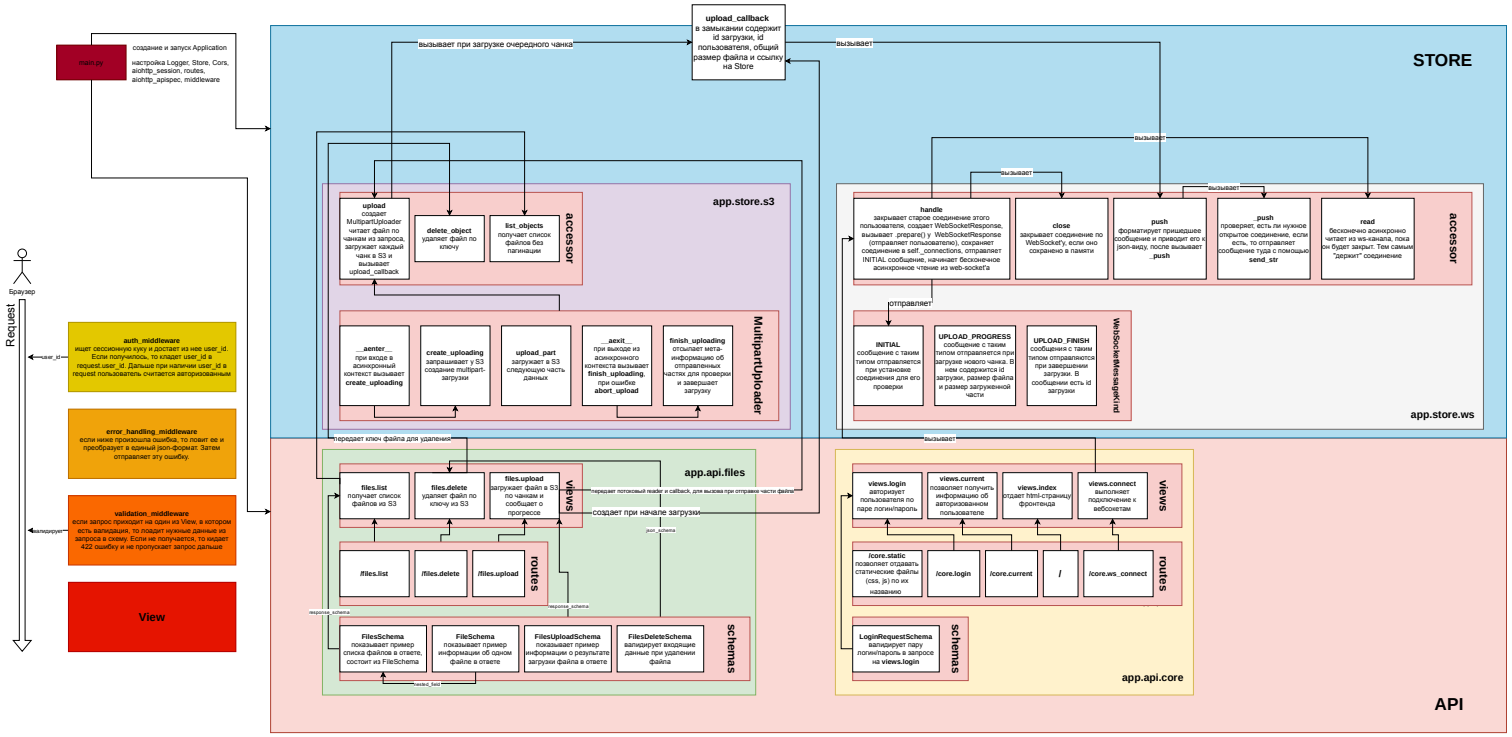
main.py
создание и запуск Application
настройка Logger, Store, Cors, aiohttp_session, routes, aiohttp_apispec, middleware
Браузер
Request
user_id
auth_middleware
ищет сессионную куку и достает из нее user_id. Если получилось, то кладет user_id в request.user_id. Дальше при наличии user_id в request пользователь считается авторизованным
error_handling_middleware
если ниже произошла ошибка, то ловит ее и преобразует в единый json-формат. Затем отправляет эту ошибку
валидирует
validation_middleware
если запрос приходит на один из View, в котором есть валидация, то подает нужные данные из запроса в схему. Если не получается, то кидает 422 ошибку и не пропускает запрос дальше
View
upload_callback
в замыкании содержит id загрузки, id пользователя, общий размер файла и ссылку на Store
вызывает при загрузке очередного чанка
вызывает
STORE
app.store.s3
upload
создает MultipartUploader читает файл по чанкам из запроса, загружает каждый чанк в S3 и вызывает upload_callback
delete_object
удаляет файл по ключу
list_objects
получает список файлов без пагинации
accessor
__aenter__
при входе в асинхронный контекст вызывает create_uploading
create_uploading
запрашивает у S3 создание multipart-загрузки
upload_part
загружает в S3 следующую часть данных
__aexit__
при выходе из асинхронного контекста вызывает finish_uploading, при ошибке abort_upload
finish_uploading
отсылает мета-информацию об отправленных частях для проверки и завершает загрузку
MultipartUploader
app.store.ws
handle
закрывает старое соединение этого пользователя, создает WebSocketResponse, вызывает _prepare() у WebSocketResponse (отправляет пользователю), сохраняет соединение в self._connections, отправляет INITIAL сообщение, начинает бесконечное асинхронное чтение из web-socket'а
close
закрывает соединение по WebSocket'у, если оно сохранено в памяти
push
форматирует пришедшее сообщение и приводит его к json-виду, после вызывает _push
_push
проверяет, есть ли нужное открытое соединение, если есть, то отправляет сообщение type с помощью send_str
read
бесконечно асинхронно читает из ws-канала, пока он будет закрыт. Тем самым "держит" соединение
accessor
отправляет
INITIAL
сообщение с таким типом отправляется при установке соединения для его проверки
UPLOAD_PROGRESS
сообщение с таким типом отправляется при загрузке нового чанка. В нем содержится id загрузки, размер файла и размер загруженной части
UPLOAD_FINISH
сообщения с такми типом отправляются при завершении загрузки. В сообщении есть id загрузки
WebSocketResponseJsonDict
передает ключ файла для удаления
передает потоковый reader и callback, для вызова при отправке части файла
создает при начале загрузки
app.api.files
files.list
получает список файлов из S3
files.delete
удаляет файл по ключу из S3
files.upload
загружает файл в S3 по чанкам и сообщает о прогрессе
views
/files.list
/files.delete
/files.upload
routes
FilesSchema
показывает пример списка файлов в ответе, состоит из FileSchema
FileSchema
показывает пример информации об одном файле в ответе
FilesUploadSchema
показывает пример информации о результате загрузки файла в ответе
FilesDeleteSchema
валидирует входящие данные при удалении файла
schemas
json_schema
response_schema
request_schema
app.api.core
views.login
авторизует пользователя по паре логин/пароль
views.current
позволяет получить информацию об авторизованном пользователе
views.index
отдает html-страницу фронтенда
views.connect
выполняет подключение к вебсокетам
views
/core.static
позволяет отдавать статические файлы (css, js) по их названию
/core.login
/core.current
/
/core.ws_connect
routes
LoginRequestSchema
валидирует пару логин/пароль в запросе на views.login
schemas
API