

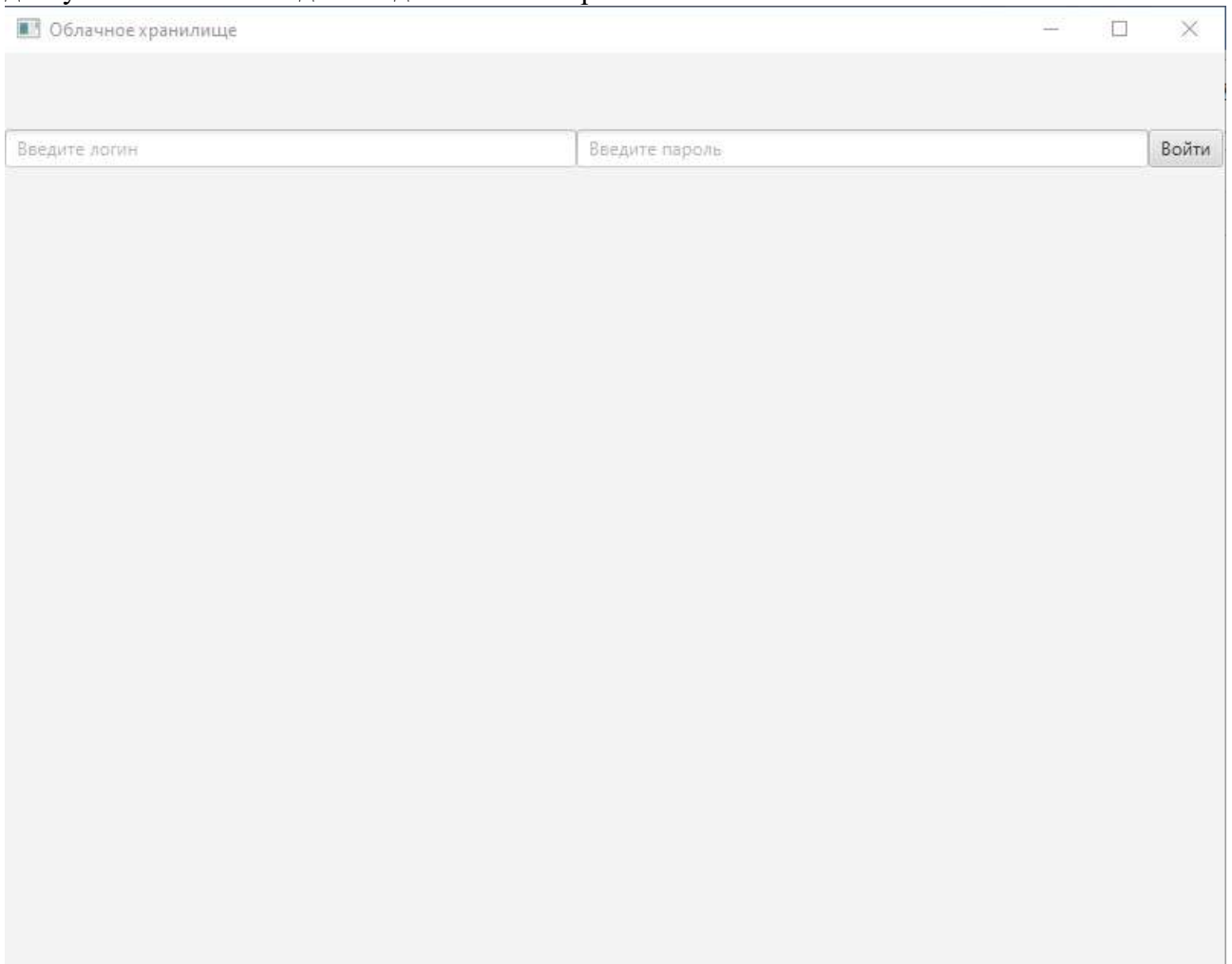
Автор проекта: Пугачева Дарья
Название проекта: Облачное хранилище

Функционал приложения:

После авторизации клиента ему становятся доступны выгрузка файлов из своей файловой системы в облако и загрузка файлов из облака.

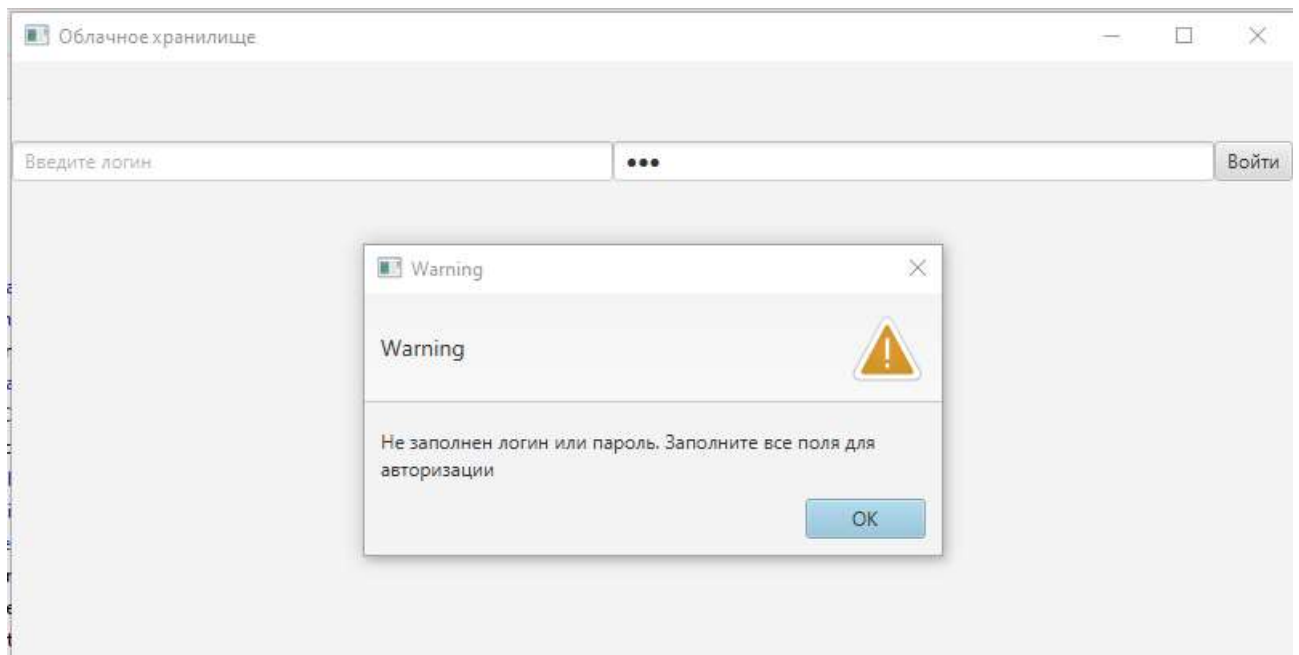
Порядок работы приложения (внешнее описание):

1. Авторизация. При запуске приложения до момента авторизации пользователю доступны только поля для ввода логина и пароля и кнопка «Войти».

The image shows a screenshot of a web application window titled "Облачное хранилище". The window has a light gray header bar with the title and standard window controls (minimize, maximize, close). Below the header, there is a login form. The form consists of two input fields: "Введите логин" (Enter login) and "Введите пароль" (Enter password). To the right of the password field is a button labeled "Войти" (Login). The background of the window is a solid light gray color.

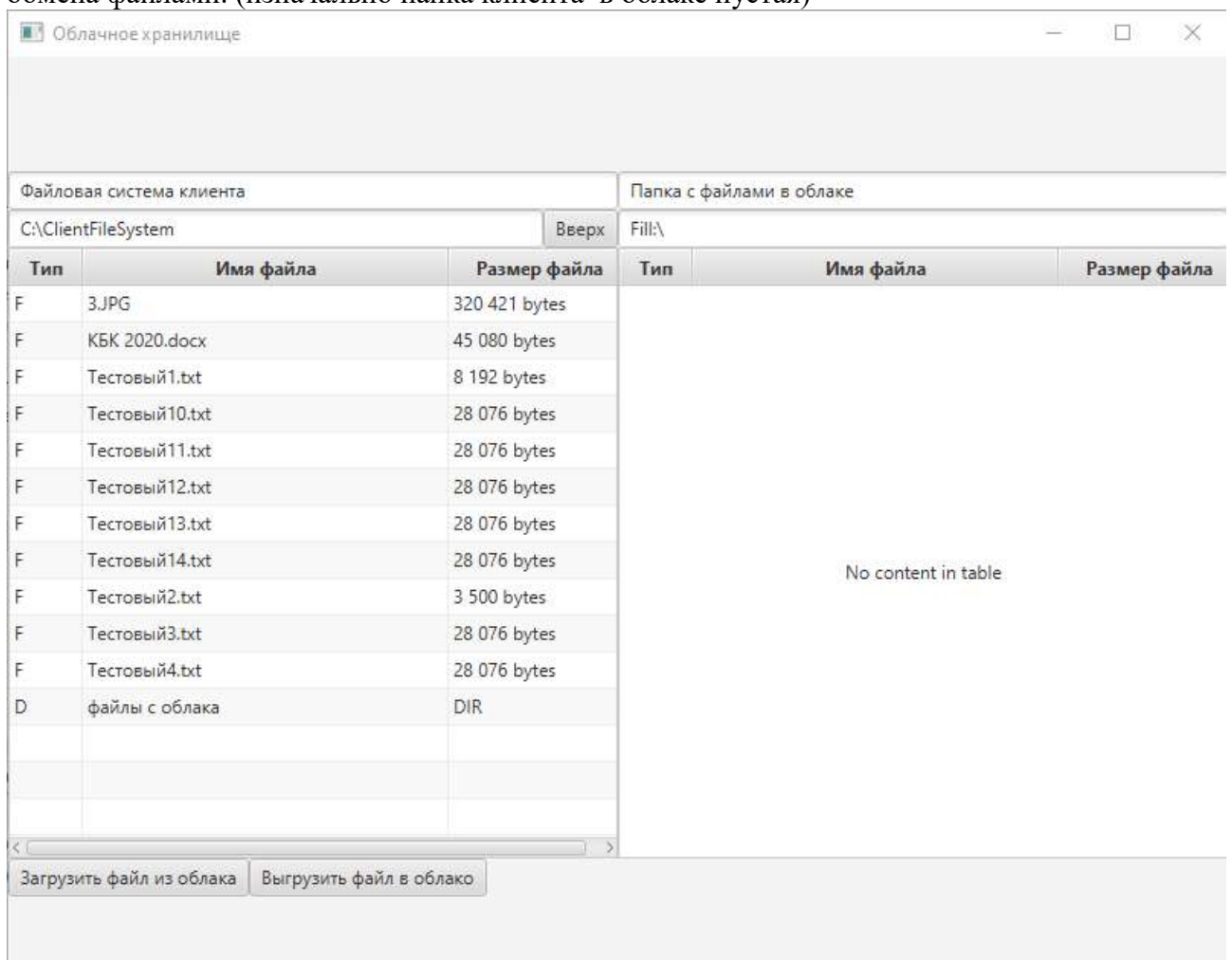
Сервер подключается к базе данных при старте и отключается при закрытии.

При авторизации осуществляется проверка, все ли поля для авторизации заполнил клиент. Если какое-то из полей не заполнено, то выдается соответствующее предупреждение:



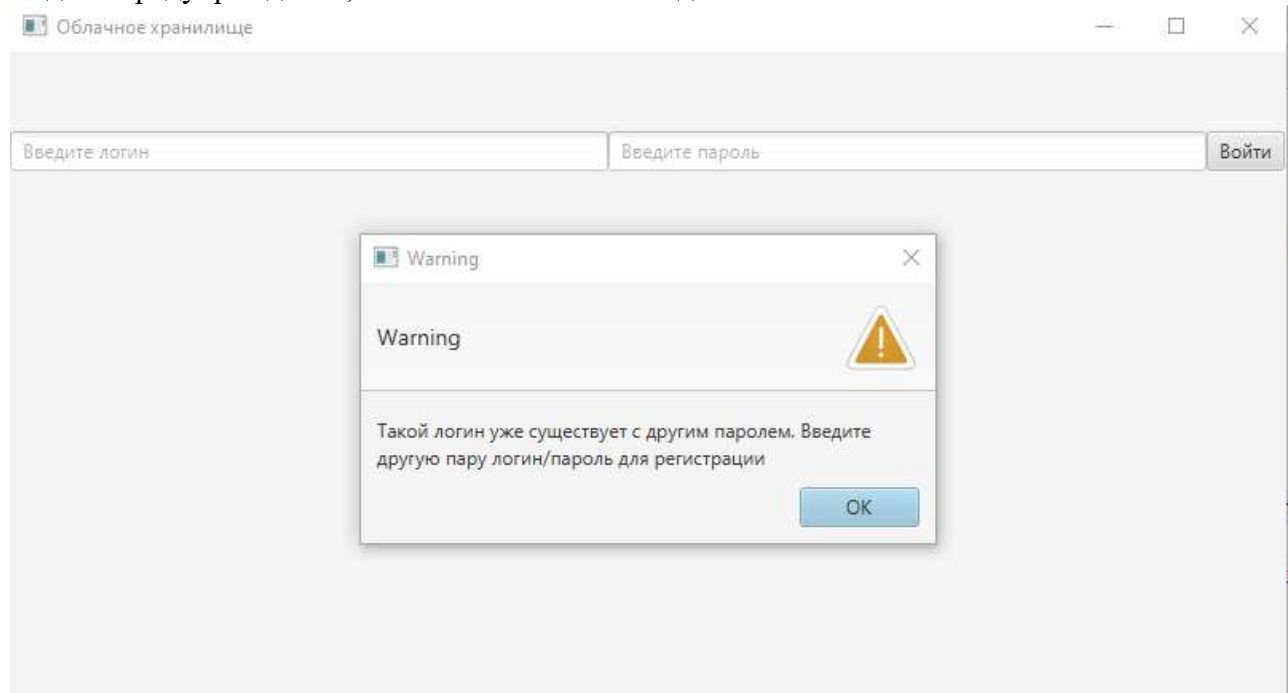
На этапе авторизации возможны три варианта развития событий:

- клиента нет в базе данных, а также нет и одноименного логина в БД. В этом случае логин и пароль записываются в БД, а на сервере создается папка с названием, соответствующим логину клиента. И именно к этой папке у клиента будет доступ для обмена файлами. (изначально папка клиента в облаке пустая)

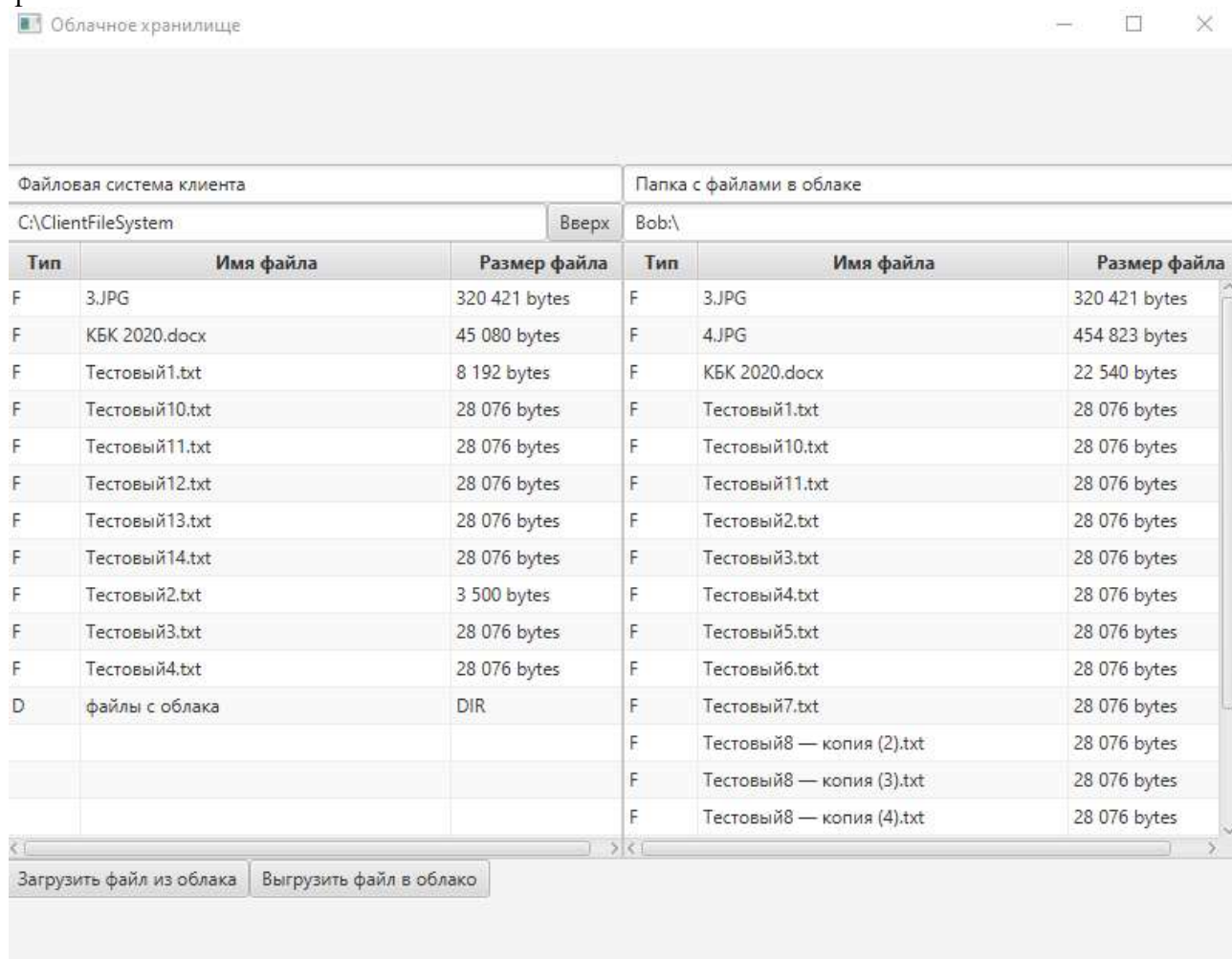


- в базе данных есть клиент с таким логином, но пароль другой. В этом случае будет

выдано предупреждение, чтобы клиент изменил данные:



- В базе данных уже есть клиент с таким логином и паролем. (т.е. его папка в облаке тоже уже существует на момент авторизации). В этом случае открывается доступ к файлам клиента:



2. В своей файловой системе клиент может передвигаться свободно между

директориями (двойным кликом мыши попадать в директории и при помощи кнопки «Вверх» подниматься в родительскую директорию.

Файловая система клиента			Папка с файлами в облаке		
C:\ClientFileSystem\файлы с облака			Вверх	Bob:\	
Тип	Имя файла	Размер файла	Тип	Имя файла	Размер файла
F	Тестовый5.txt	28 076 bytes	F	3.JPG	320 421 bytes
F	Тестовый6.txt	28 076 bytes	F	4.JPG	454 823 bytes
F	Тестовый8 — копия (2).txt	28 076 bytes	F	КБК 2020.docx	22 540 bytes
F	Тестовый8 — копия (3).txt	28 076 bytes	F	Тестовый1.txt	28 076 bytes
F	Тестовый8 — копия (4).txt	28 076 bytes	F	Тестовый10.txt	28 076 bytes
F	Тестовый8 — копия (5).txt	28 076 bytes	F	Тестовый11.txt	28 076 bytes
F	Тестовый8 — копия (6).txt	28 076 bytes	F	Тестовый2.txt	28 076 bytes
			F	Тестовый3.txt	28 076 bytes
			F	Тестовый4.txt	28 076 bytes
			F	Тестовый5.txt	28 076 bytes
			F	Тестовый6.txt	28 076 bytes
			F	Тестовый7.txt	28 076 bytes
			F	Тестовый8 — копия (2).txt	28 076 bytes
			F	Тестовый8 — копия (3).txt	28 076 bytes
			F	Тестовый8 — копия (4).txt	28 076 bytes

На стороне сервера движение вверх не предусмотрено для того, чтобы клиент имел доступ исключительно к своей папке в облаке, а также потому, что исхожу из условного ограничения, что в облаке клиент не создает папки, а хранит лишь файлы, т.е. никакое сквозное движение по директориям не требуется.

3. При выгрузке файлов из системы клиента в облако и из облака к клиенту происходит обновление списка файлов в облаке и на клиенте соответственно.

Выгрузка / загрузка реализована через нажатие кнопок «Выгрузить файл в облако» и «Загрузить файл из облака». При этом должен быть выбран файл на соответствующей стороне. Если клиент перепутал кнопки (файл выбран не там), то высвечивается предупреждение о необходимости выбрать файл в директории-источнике. Кнопки выгрузки и загрузки становятся нерабочими на период, пока файл выгружается/загружается, а по окончании загрузки файла блокировка кнопок снимается.

Облачное хранилище

Файловая система клиента

Папка с файлами в облаке

C:\ClientFileSystem

Вверх

Bob:\

Тип	Имя файла	Размер файла
F	3.JPG	320 421 bytes
F	КБК 2020.docx	454 823 bytes
F	Тестовый1.txt	22 540 bytes
F	Тестовый10.txt	28 076 bytes
F	Тестовый11.txt	28 076 bytes
F	Тестовый12.txt	28 076 bytes
F	Тестовый13.txt	28 076 bytes
F	Тестовый14.txt	28 076 bytes
F	Тестовый2.txt	3 500 bytes
F	Тестовый3.txt	28 076 bytes
F	Тестовый4.txt	28 076 bytes
D	файлы с облака	DIR
F	Тестовый2.txt	28 076 bytes
F	Тестовый3.txt	28 076 bytes
F	Тестовый4.txt	28 076 bytes
F	Тестовый5.txt	28 076 bytes
F	Тестовый6.txt	28 076 bytes
F	Тестовый7.txt	28 076 bytes
F	Тестовый8 — копия (2).txt	28 076 bytes
F	Тестовый8 — копия (3).txt	28 076 bytes
F	Тестовый8 — копия (4).txt	28 076 bytes

Загрузить файл из облака

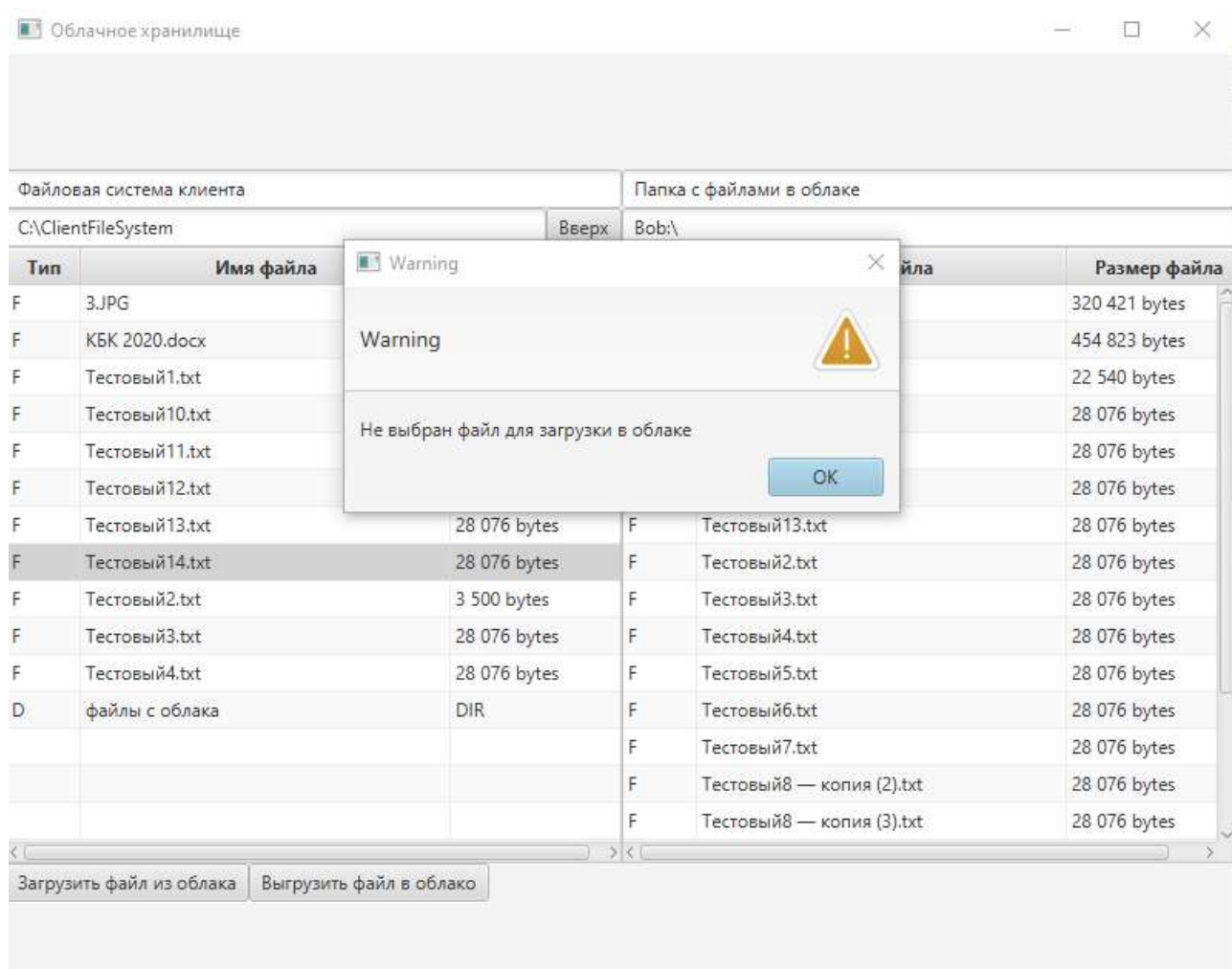
Выгрузить файл в облако

Warning

Warning

Не выбран файл для выгрузки в облако

OK



Основные принципы и решения при реализации работы приложения:

1. Для реализации внешней формы используется функционал Java FX
2. Для обмена данными между сервером и клиентом с обеих сторон используется библиотека Netty.
3. Обмен между клиентом и сервером реализован за счет отправления команд-сообщений (исчерпывающий перечень названий команд перечислен в специальном вспомогательном классе `enum`) с необходимыми аргументами, в результате которых принимающая сторона осуществляет соответствующие действия:
 - отправляется ответная команда, в том числе с запрошенными данными,
 - меняется содержание внешней формы (обновление списка файлов)
 - выводятся сообщения об ошибке в случае некорректных запросов
 - для приема файлов выстраивается в `pipeline` подходящая для этого функционала цепочка хэндлеров;
 - по окончании приема файлов цепочка хэндлеров в `pipeline` выстраивается в изначальный (базовый) вариант, подходящий для приема команд с аргументами.
4. Для работы с БД используется SQLite.
5. Необходимая информация про БД, порт, путь к файловой системе клиента и папке клиента в облаке сохранена в файлах `.properties` в модуле клиента и сервера.

Описание функционала классов:

1. Модуль `client`.

Controller является связкой между Интерфейсом клиента и функционалом приложения.

ClientInboundCommandHandler предназначен для интерпретации поступающих клиенту команд от сервера. В зависимости от поступившей команды производит вызов исполнения команды предназначенным для этого классом, отправку ответной команды на сервер, обращается к классу-переключателю цепочки хэндлеров в pipeline.

FilesInboundClientHandler предназначен для корректного получения от сервера файлов. Реализовано получение файла, направление ответной команды о результате получения файла, обращение к классу-переключателю цепочки хэндлеров в pipeline.

ClientPipelineCheckoutService предназначен для выстраивания хэндлеров в pipeline в требуемой последовательности.

NettyNetworkService предназначен для установления сетевого соединения. Также в нем реализованы методы отправки команд и файлов.

Factory предназначен для инициализации объектов классов.

AcceptedLoginCommand, CloudFilesListCommand, FailedLoginCommand, UploadFileCommand реализуют связку между хэндлером и контроллером для исполнения соответствующих команд.

ClientCommandDictionaryServiceImpl – класс-аккумулятор команд (словарь команд), через который удобно обращаться к любой из вышеперечисленных команд.

ClientPropertiesReciever – класс, реализующий удобное получения значений из properties-файлов.

MainClientApp – точка входа для клиента.

2. Модуль common (в нем находятся классы, общие для модулей client и server)

Command – для обмена командами создаются объекты этого класса (с подходящими для функционала приложения полями)

CommandType – класс-перечисление – справочник названий команд для единообразия и исключения возможных ошибок в случае опечаток.

FileInfo – в виде объектов этого класса передается информация о файлах.

3. Модуль server

CommandInboundHandler предназначен для интерпретации поступающих на сервер команд от клиента. В зависимости от поступившей команды производит вызов исполнения команды предназначенным для этого классом, отправку ответной команды клиенту, обращается к классу-переключателю цепочки хэндлеров в pipeline.

FilesInboundHandler предназначен для корректного получения от клиента файлов. Реализовано получение файла, направление ответной команды о результате получения файла, обращение к классу-переключателю цепочки хэндлеров в pipeline.

ServerPipelineCheckoutService предназначен для выстраивания хэндлеров в pipeline в требуемой последовательности.

NettyServerService предназначен для установления сетевого соединения. При установлении соединения также реализовано подключение к базе данных (обращение к классу DatabaseConnectionService)

Factory предназначен для инициализации объектов классов.

AuthenticationCommand и **ListCreatingCommand** реализуют исполнение команд (аутентификации клиента и создания листа файлов в директории клиента в облачном хранилище), которые CommandInboundHandler запускает к исполнению через CommandDictionaryService. Команды реализуются за счет обращения к соответствующим им сервисам (AuthenticationServiceImpl и ListOfClientFilesInCloudCreatingService)

DatabaseConnectionServiceImpl реализует логику подключения к базе данных.

ServerPropertiesReciever – класс, реализующий удобное получения значений из properties-файлов.

MainServerApp – точка входа для сервера.