# DApp - File Store

Our decentralized application is an online market where users can sell and buy files. The webstore is implemented in Python Flask, the chaincode in Node.js, and the wrapper to access the chaincode by exposing a REST interface also in Node.js. For decentralized and immutable records we use the blockchain technology Hyperledger Fabric.
The complete source code is available at https://github.com/Alexx882/hlf-dapp.

Running the `./startFilestore.sh` script first spins up the Fabric network, creates the channel *mychannel,* and installs and instantiates the FabFile chaincode. Finally it builds and starts the Docker containers for our webstore and chaincode wrapper.

Once started, the webshop is accessible on port 5000.
First a user has to register with username, email, and password (currently just stored inside the container). We chose to not differentiate users between buyer and seller as they should be able to do both transparently. A users gets 1,000 credit initially.
Once logged in, the user sees all offered files. If a new file is uploaded, it is available for purchase automatically. Other users can now buy the file. Therefore, their balance is checked against the file purchase price in the blockchain and the credit is transferred to the file owner. Additionally this sales record is stored, including filename, file hash, file price, and buyername. On success, the new owner is transferred to the corresponding download page. Clicking on the file again allows the download again.
A file can be uploaded by clicking on the '+' next to the Offers headline. The user is then able to enter the desired price and the file to offer. Later, the file can be set unavailable by clicking on the stop symbol. Then it is not listed anymore.

## Group Members

Manuel Herold, 01461450
Alexander Lercher, 01560095
Michael Marolt, 11813960

## Webstore

The Webstore is in the folder `/flask_backend` and consists of a web application written in Python Flask.

Users can register themselves in the webstore which then propagates the data to the blockchain. The balance of the user is retrieved from the blockchain each time a user signs in. Files can be offered in the store. The resulting offer contains the filename, the seller, the price, and the hash of the file.

When a user buys the file, the backend stores the user's possession of the file and notifies the chaincode about the buy-action and the new balance of the user, which also stores it in the blockchain. The user can now download the file through the store.

## Hyperledger Fabric

The Hyperledger Fabric was cloned from https://github.com/hyperledger/fabric-samples. We use an adapted version of the basic-network example, where the CA was already preconfigured. This source code is in the folder `/blockchain/basic-network` and consists of a docker-compose and various settings for Orderer, Peer, CLI, and CA.

## Hyperledger Fabric Chaincode

All chaincode for the blockchain is contained in `/chaincodes`. The Node.js module in `/chaincodes/filescc/` contains three contracts.

*FabUser* is used to register new users to the network. Additionally existing users can be queried and updated.
*FabFile* contains the logic for registering files to sell as well as updating their availability and price. This contract also contains the method to buy a file. Therefore the filename and buyername has to be given as parameter. Then the contract checks if the file is available and if the buyer has enough credit. If everything is okay, the buyer successfully bought the file from the seller.
*FabSale* stores the successfully processed sales, which just adds sales records to the blockchain for easier querying.

Examples of how to use the methods and their parameters can be found in `/chaincodes/commands.sh`.

## Hyperledger Fabric Chaincode Wrapper

The HLF REST Wrapper invokes and queries the chaincode and provides an API for the frontend. It provides routes for registering users, updating the user credit, updating the user trading type and querying users. Furthermore you can register files, update the availability and the price for these files. Finally the files can be bought and the complete sales history is queryable.
The wrapper uses the Hyperledger Fabric SDK for Node.js to forward the REST requests to the corresponding chaincode. The authorized user for interacting with the blockchain is called *filecc-user* and is stored in the Docker image for less manual setup time.
We used environment variables to connect to the correct endpoints in the Hyperledger Fabric, namely for orderer, peer, and certificateAuthority.
The whole source code is contained in `/hlf-rest-wrapper`.

Examples of how to use the routes and their parameters can be found on
https://documenter.getpostman.com/view/9307019/SWT5gzpu?version=latest

## Docker

We dockerized our webstore and wrapper modules for easy deployment with minimal setup time. The Hyperledger Fabric implementation is already dockerized.
We used the example scripts to create an *admin* and *filecc-user* to interact with the blockchain. The wallet which contains these credentials is added to the Docker image so users of the File Store App do not need to create the users themselves.
The `blockchain/basic-network/docker-compose.yml` contains all the information to start the Docker containers and to enable communication between them.

Note: The university uses a proxy, therefore this information was added to the Dockerfiles on the *hlf01.itec.aau.at* machine. Additionally, we found out that no ports are opened to the outside world to access the application.