Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Digitálne meny a blockchain

Zadanie 3 – Hyperledger Fabric smart systém

Mena: Lukáš Michalík, Lucia Murzová, Andrii Rybak

Cvičiaci: Ing. Kristián Košťál, PhD.
Ing. Lukáš Mastiľak

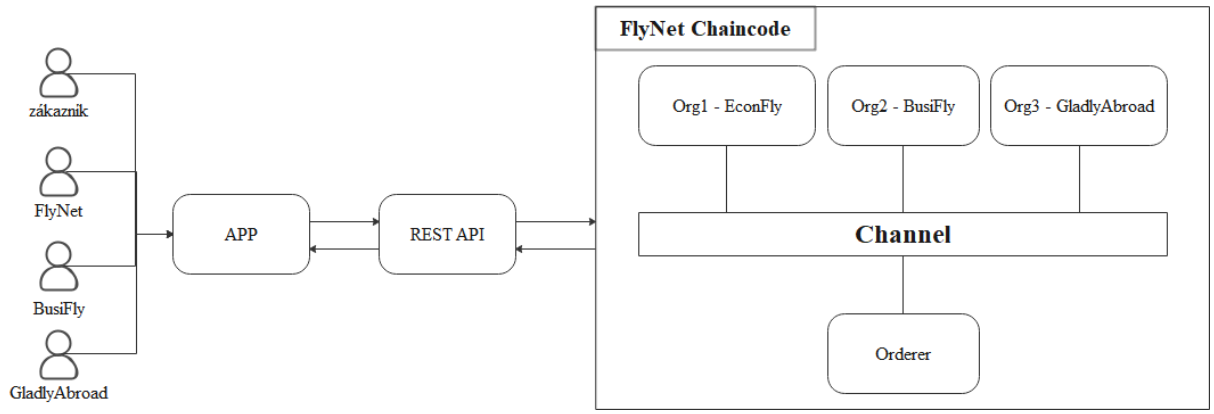
Akademicky rok: 2021/2022

Obsah

1. Návrh siete

V našom návrhu sme sa rozhodli využiť iba jeden kanál, do ktorého vstupujú všetky 3 organizácie a samozrejme orderer. Dve organizácie sú vytvorené pre letecké spoločnosti EconFly a BusiFly a tretia pre cestovnú kanceláriu GladlyAbroad.

Zákazníci a letecké spoločnosti komunikujú so systémom cez frontend aplikáciu, pričom po vytvorení objednávky zákazníkom sú jeho dáta spracovávané cestovnou kanceláriou Gladly Abroad, ktorá ich posiela ďalej a vytvára požadovanú rezerváciu.



2. Implementované časti kódu

2.1. Chaincode

Chaincode sa nachádza v súbore assetTransfer.js v adresári chaincode-javascript/lib a je napísaný v jazyku javascript.

Function createFlight(flyFrom, flyTo, dateTime, seats)

Funkcia overuje, ktorá z organizácií transakciu zavolala pomocou funkcie ctx.clientIdentity.getMSPID(). V tomto prípade môžu lety vytvárať iba aerolínie EconFly a BusiFly.

Funkcia vytvorí let s unikátnym ID. Ak bola transakcia zavolaná spoločnosťou EconFly, ID bude začínať "EC" a vprípade že bola zavolaná spoločnosťou BusiFly začína "BS". Let má ako seats uložené pole miest v lietadle. Pokiaľ je miesto voľné, na tomto indexe je string "free", inak sa tam nachádza pazažierovo meno a jeho passportID.

Function checks which organization sent the transaction using ctx.clientIdentity.getMSPID(). In this case, only airlines can create a new flight. There is a change in flight attributes. In this solution additional attribute seats: array[] represents all seats in the plane. For example, if seat 47 is free, seats[47] == "free", otherwise, there will be a string with customer name and passport information. Function creates flight with new unique id which begins with "EC" if function was invoked from EconFly and "BS" if invoked by BusiFly.

Function reserveSeats(flightNr, seatNum, customerNames[], customerEmail)

Sedadlá môžu byť rezervované iba cestovkou GladlyAbroad. Vznikne nová rezervácia s inukátnym číslom a jej stav je nastavený na "pending". Číslo rezervácie je vytvorené automaticky a vždy začína písmenom R, napríklad "R0", "R17".

Function bookSeats(reservationNr)

Túto transakciu môže volať iba aerolínia, ktorá vytvorila let, pre ktorý je zadaná rezervácia. Ak boli sedadlá zarezervované správne, status rezervácie je nastavený na "completed".

Function checkIn(reservationNr, passportIDs[], seats[])

Check in môže byť volaný iba cestovnou kanceláriou GladlyAbroad. Funkcia kontroluje stav zadanej rezervácie a pre úspešné vykonanie check inu musí byť stav "completed" (prebehlo zabookovanie a teda potvrdenie leteckou spoločnosťou).

Argument passportIDs je pole json objektov vo formáte:

{name: "Bohus Bohus", passport: "OP123456"}

Funkcia kontrolu, či sa každá osoba v poli passportIDs nachádza v príslušnej rezervácii. V poli seats sú čísla sedadiel, ktoré si zákazníci vybrali a podľa indexov sa tak v zozname letu prepisujú voľné polia na informácie konkrétnych zákazníkov. Status rezervácie je nastavený na "checked-in" a uložený v ledgery. Funkcia vráti lístky pre každého zákazníka z danej rezervácie.

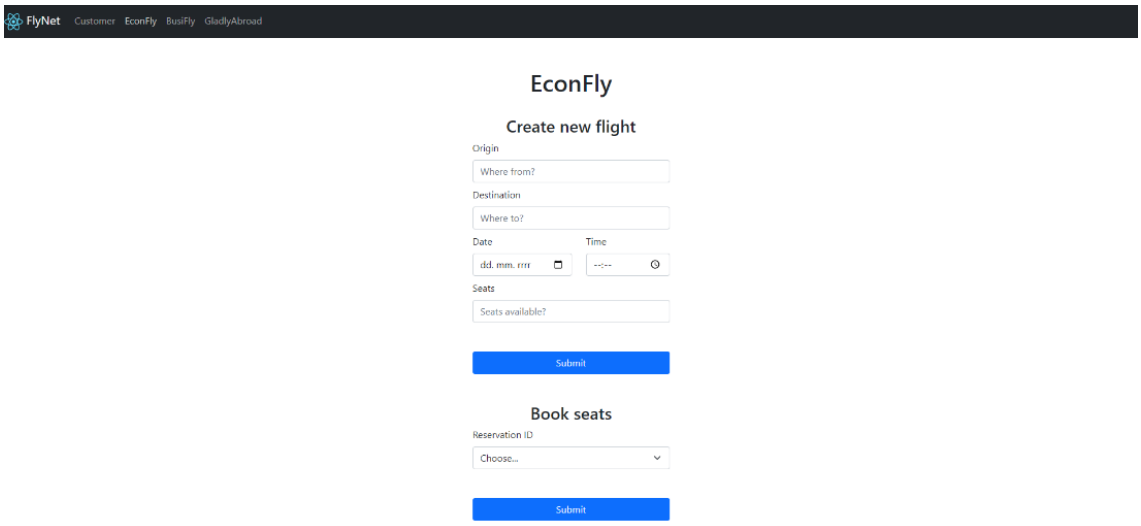
Implementovali sme aj funkcie ako getAllFlights() a getFlight(id), ktoré môžu slúžiť na overenie alebo debugovanie.

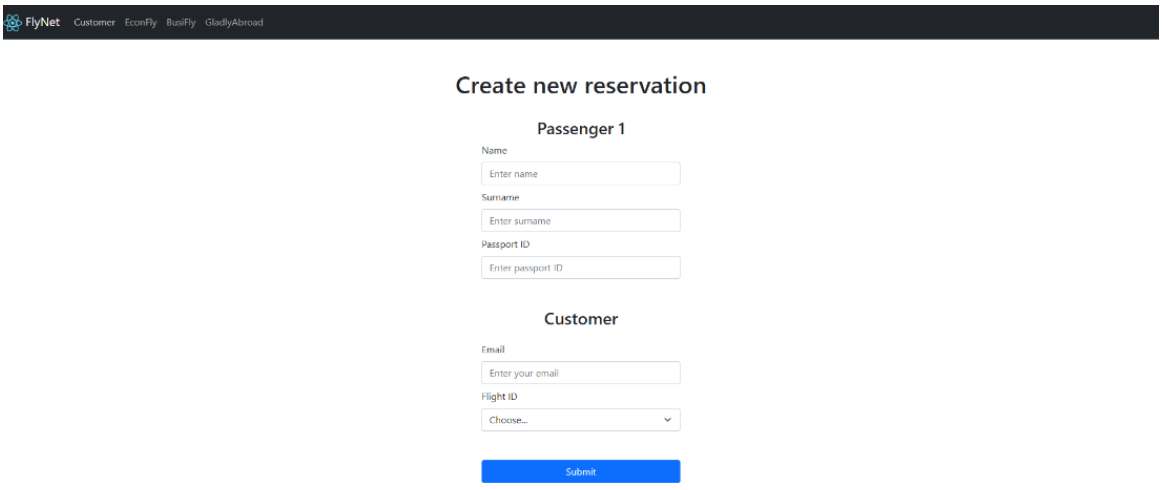
Všetky spomenuté funkcie akceptujú argumenty iba vo formáte string.

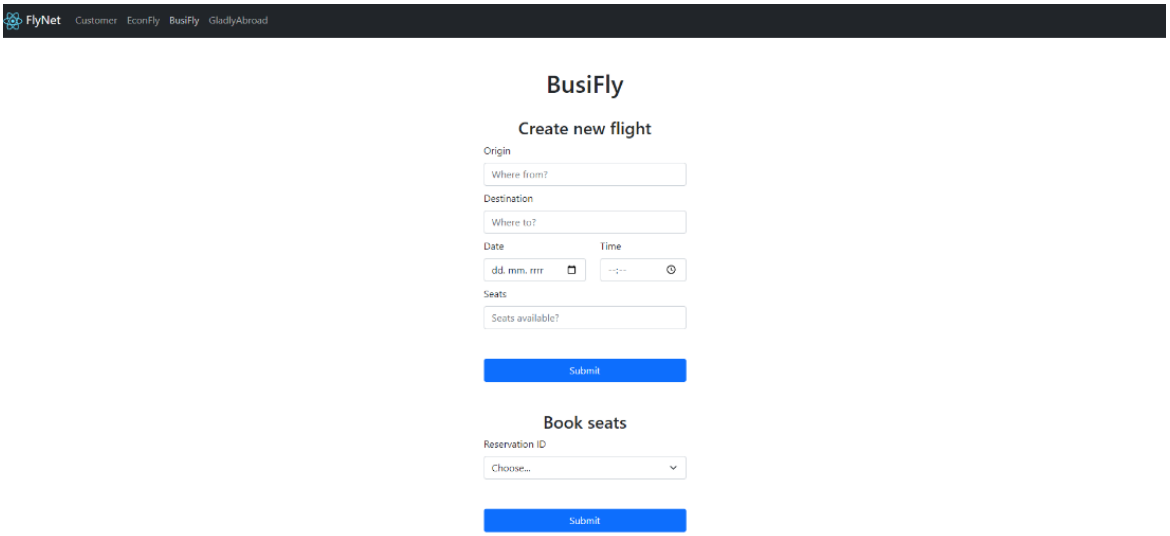
2.2. Používateľské rozhranie

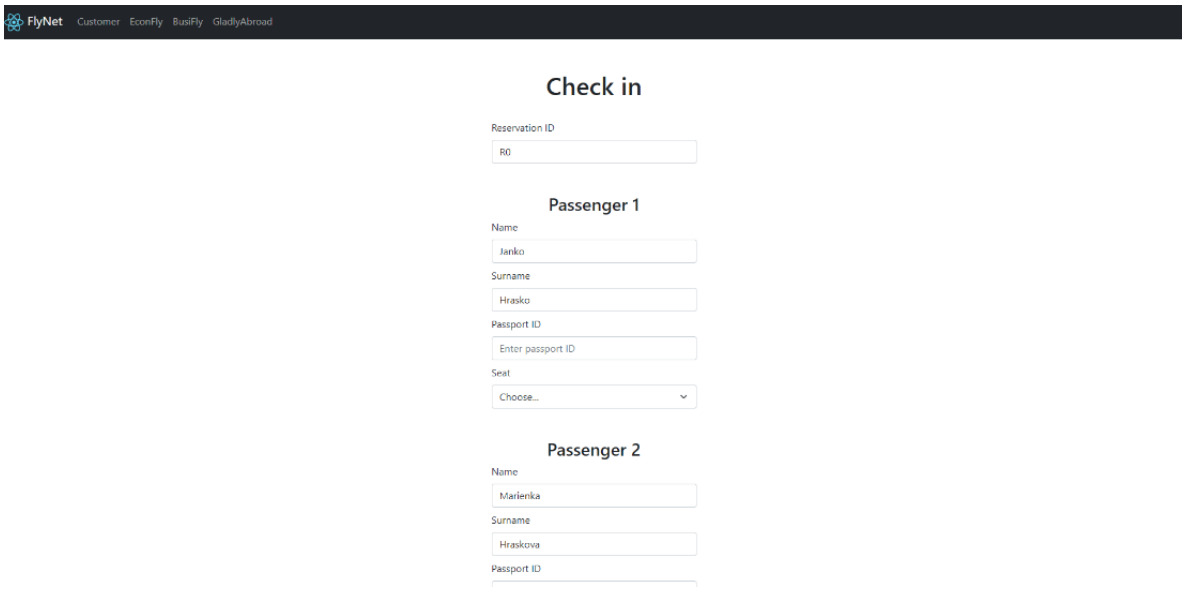
Ako používateľské rozhranie sme vytvorili webovú aplikáciu, ktorá získava dáta z chaincodu pomocou volaní endpointov na REST API serveri (FlyNet/app/server.js). Server požiadavku spracuje a preposiela v požadovanom tvare na chaincode, pričom na frontend vždy posiela celú odpoveď chaincodu.

V rámci webovej aplikácie je možné sa prepínať medzi jednotlivými používateľmi - letecké spoločnosti EconFLy a BusiFly, cestovnou kanceláriou Gladly Abroad a zákazníkom. Toto riešenie zamedzuje situácii, kedy by volania na chaincode boli neoprávnené - zákazník nemá možnosť vytvárať let a podobne.






FlyNet
Customer
EconFly
BusiFly
GladlyAbroad
BusiFly
Create new flight
Origin
Where from?
Destination
Where to?
Date
dd. mm. rrrr
Time
--:--
Seats
Seats available?
Submit
Book seats
Reservation ID
Choose...
Submit


FlyNet
Customer
EconFly
BusiFly
GladlyAbroad
Check in
Reservation ID
R0
Passenger 1
Name
Janko
Surname
Hrasko
Passport ID
Enter passport ID
Seat
Choose...
Passenger 2
Name
Marienka
Surname
Hraskova
Passport ID

3. Implementačné prostredie

Zvolili sme Hyperledger Fabric verziu 2.4.2, node vo verzii 12.16.3. Pri voľbe implementačného prostredia sme sa zhodli na jazyku Javascript. Využili sme preto NodeJs a na frontend tiež ReactJs. Pri všetkých častiach projektu sme pracovali v prostredí VS Code.

4. Testovanie

Súbor pre testovanie assetTransfer.test.js sa nachádza v adresári chaincode-javascript/test a je napísaný tiež v javascripte s využitím mocha, chai a sinon. Pre testovanie je potrebné v adresári chaincode-javascript spustiť nasledujúce príklady:

1. Npm install
2. Npm test

```
Tests
  Test createFlight
    ✓ should return error on createFlight (create flight by GladlyAbroad)
    ✓ should return success on createFlight
  Test reserveSeats
    ✓ should return success on reserveSeats
  Test bookSeats
    ✓ should return success on bookSeats
  Test checkIn
    ✓ should return success on checkIn
  Check the whole ledger content
    ✓ Check the whole ledger content


6 passing (27ms)
```

Testy reprezentujú jednoduché scenáre ako vytvorenie letu, rezervovanie sedadiel cestovnou kanceláriou, ich zabookovanie leteckou spoločnosťou a check in pasažierov. Nakoniec je výsledný stav ledgeru porovnávaný s jeho očakávaným stavom po týchto transakciách

Výsledky pokrytia testov:

```
============================== Coverage summary ==============================
Statements   : 72.06% ( 98/136 )
Branches     : 51.67% ( 31/60 )
Functions    : 83.33% ( 10/12 )
Lines        : 71.21% ( 94/132 )
==============================================================================
```

5. Čím sa chceme pochváliť

Napísali sme vlastný skript startFlyNet.sh, ktorý zabezpečí:

- Spustenie siete s vytvorením nového kanála
- Vytvorenie a pridanie troch organizácií do systému
- Deploy kontraktu
- Vykonanie príkazu docker ps –a, ktorý vypíše všetky bežiace docker procesy s ich statusmi

6. Otázky

Podľa vášho názoru, je takéto blockchain-based riešenie najlepšia možnosť na vyriešenie daných výziev? Aké sú výhody a nevýhody používania technológie distribuovanej účtovnej knihy (distributed ledger technology) v porovnaní s centralizovaným systémom na tomto konkrétnom prípade použitia? Napíšte krátke vysvetlenie a vaše názory.

Myslíme si, že centralizované riešenie je v prípade systému pre aerolínie lepším riešením ako blockchainové a to z viacerých dôvodov. V prvom rade veríme, že v takomto prípade blockchain značne obmedzuje šírku pásma a vo všeobecnosti plytvá výpočtovými zdrojmi, pretože každú transakciu je potrebné overiť niekoľkými uzlami, čo v prípade centralizovaného riešenia nie je potrebné.

Ďalší z problémov blockchain riešenia je fakt, že je náročné na vývoj a tiež niekoľkonásobne náročnejšie aj na údržbu.

Jednou z výhod takéhoto riešenia však môže byť jeho dostupnosť. Centralizované databázy sú obmedzené na jednu lokáciu a môžu ľahko zlyhať. Blockchain je distribuovaný na veľa zariadení, avšak takmer nikdy nezlyhá.

7. Záver

Rozdelenie práce:

- Andrii – chaincode, testovanie, dokumentácia
- Lucia – server, dokumentácia
- Lukáš - frontend aplikácia, dokumentácia

Vďaka práci na zadaní sme mali možnosť bližšie pochopiť fungovanie privátnych blockchain sietí a tiež spoznať a využiť nové technológie ako React a NodeJS. Aj napriek všetkej snahe sa nám pri komunikácii s chaincodom v súbore server.js podarí vytvoriť iba administrátora jednej organizácie, a tak je využitie webovej aplikácie značne obmedzené. Webové rozhranie je však dobre spravené a preto nás mrzí, že nám jeho prepojenie s chaincodom nefunguje úplne správne. Z tohto dôvodu odporúčame funkčnosť samotného chaincodu testovať priamo v konzole podľa postupu v prílohe, prípadne pri aktuálnom nastavení vo webovej aplikácii vytvoriť nový let pre spoločnosť Econ Fly.

Používateľská príručka pre webovú aplikáciu

1. Spustite terminál v adresári FlyNet/network
2. Zadajte príkaz: ./startFlyNet.sh
3. Presuňte sa do adresára aplikácie:
   cd ../app
4. Zadajte príkazy:
   npm install
   npm start
5. Otvorte si webovú aplikáciu na porte 8080
   localhost:8080
6. Ak chcete ukončiť prácu so systémom alebo začať od začiatku, ukončite beh server.js stlačením Ctrl+C, odstráňte vytvorený adresár wallet a presuňte a do adresára FlyNet/network kde zadajte príkaz:
   ./network.sh down

Používateľská príručka pre konzolovú aplikáciu

Je potrebné spustenie siete podľa krokov 1.-2. v príručke pre webovú aplikáciu. Rovnako je možné využívať príkaz ./network.sh down pre vypnutie siete a odstránenie všetkých vykonaných transakcií. V konzole je ďalej možné aplikáciu využívať podľa nižšie opísaných volaní.

- Nastavenie premenných
  export PATH=${PWD}/../bin:$PATH
  export FABRIC_CFG_PATH=$PWD/../config/
- Pre volanie transakcií v mene leteckej spoločnosti EconFly – Org1 zadajte tieto príkazy:
  export CORE_PEER_TLS_ENABLED=true
  export CORE_PEER_LOCALMSPID="Org1MSP"
  export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
  export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
  export CORE_PEER_ADDRESS=localhost:7051

- Pre volanie transakcií v mene leteckej spoločnosti BusiFly– Org2 zadajte tieto príkazy:
  export CORE_PEER_TLS_ENABLED=true
  export CORE_PEER_LOCALMSPID="Org2MSP"
  export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
  export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
  export CORE_PEER_ADDRESS=localhost:9051

- Pre volanie transakcií v mene cestovnej kancelárie Gladly Abroad – Org3 zadajte tieto príkazy:
  export CORE_PEER_TLS_ENABLED=true
  export CORE_PEER_LOCALMSPID="Org3MSP"

export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp
export CORE_PEER_ADDRESS=localhost:11051

Príklady volania funkcií:

1. Vytvorenie letu

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"createFlight","Args":["BUD", "TXL", "05032021-1034", "10"]}'
```

2. Vytvorenie rezervácie

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"reserveSeats","Args":["EC0", "1", "[\"Andrii\"]", "lol@gmail.com"]}'
```

3. Zabookovanie sedadiel

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"bookSeats","Args":["R0"]}'
```

4. Check in

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"checkIn","Args":["R0", "[{\"name\": \"Andrii\", \"passport\": \"OP123456\"}]", "[5]"]}'
```

5. Získanie konkrétneho letu

```
peer chaincode query -C mychannel -n basic -c '{"function":"getFlight","Args":["EC0"]}'
```

6. Získanie všetkých letov

```
peer chaincode query -C mychannel -n basic -c '{"Args":["getAllFlights"]}'
```