Architecture and Administration Basics

Workshop **Day 2 – Labs**
C API

Eyal Nussbaum, Solutions Architect
https://github.com/couchbase-ps/cb-workshop-2d/

1 Installation & Configuration SDK

SDK C - Sample Application

We will be using the sample application for this workshop, the description of which can be found:

- https://docs.couchbase.com/c-sdk/current/hello-world/sample-application.html

The sample application consists of 3 components:

- Couchbase Server (we will be using 7.1.4)
- Backend API (we will be implementing)
- Frontend Application (premade)

The backend API will be based on the C SDK - libcouchbase (https://github.com/couchbase/libcouchbase) and use Kore.io for the web server framework.

The sample application contains a lot of “wrapper code“ for configuration, error handling, communication which we are outside the scope of this workshop.

Documentation & Examples

Open the documentation for libcouchbase!

- https://docs.couchbase.com/c-sdk/current/hello-world/start-using-sdk.html
- https://docs.couchbase.com/c-sdk/current/howtos/kv-operations.html
- https://docs.couchbase.com/c-sdk/current/howtos/subdocument-operations.html
- https://docs.couchbase.com/c-sdk/current/howtos/n1ql-queries-with-sdk.html

Load and build the Project

For simplicity, we will be running all 3 Sample Application components as Docker containers. As we implement the API, we will rebuild the backend container with updated functionality.

To begin, if you have not already done so, clone the workshop git repo:

- git clone https://github.com/couchbase-ps/cb-workshop-2d
- Open your preferred IDE and load the folder:
  - try-cb-lcb-labs project => This is where you start.
  - try-cb-lcb-solution project => This is the solution (for reference).

Inside the working directory, run

`docker-compose -f docker-compose-7.yml up --build`

to initialize the Docker instances. You will note in the docker-compose-7.yml file that the backend API is built locally while the other components are remote.

Verify Build Process

If all components loaded correctly, your terminal should show the following:

```
Starting couchbase-sandbox-7.1.4-wss ... done
Starting try-cb-api-wss-7.1          ... done
Starting try-cb-fe-wss-7.1           ... done
Attaching to couchbase-sandbox-7.1.4-wss, try-cb-api-wss-7.1, try-cb-fe-wss-7.1
couchbase-sandbox-7.1.4-wss | Starting Couchbase Server -- Web UI available at http://<ip>:8091
couchbase-sandbox-7.1.4-wss | and logs available in /opt/couchbase/var/lib/couchbase/logs
couchbase-sandbox-7.1.4-wss | Container previously configured.
try-cb-api-wss-7.1 | CB_SCHEME=couchbase://
try-cb-api-wss-7.1 | CB_HOST=db
try-cb-api-wss-7.1 | CB_USER=Administrator
couchbase-sandbox-7.1.4-wss | Couchbase Admin UI: http://localhost:8091
couchbase-sandbox-7.1.4-wss | Login credentials: Administrator / password
try-cb-api-wss-7.1 | wait-for-couchbase: checking http://db:8091/pools/default/buckets/travel-sample/
try-cb-api-wss-7.1 | wait-for-couchbase: polling for '.scopes | map(.name) | contains(["inventory", "
try-cb-fe-wss-7.1 | wait-for-it: waiting 400 seconds for backend:8080
try-cb-api-wss-7.1 | wait-for-couchbase: checking http://db:8094/api/cfg
try-cb-api-wss-7.1 | wait-for-couchbase: polling for '.status == "ok"'
try-cb-api-wss-7.1 | wait-for-couchbase: checking http://db:8094/api/index/hotels-index
try-cb-api-wss-7.1 | wait-for-couchbase: polling for '.status == "ok"'
try-cb-api-wss-7.1 | wait-for-couchbase: index already exists
try-cb-api-wss-7.1 | wait-for-couchbase: checking http://db:9102/api/v1/stats
try-cb-api-wss-7.1 | wait-for-couchbase: polling for '.indexer.indexer_state == "Active"'
try-cb-api-wss-7.1 | wait-for-couchbase: polling for '. | keys | contains(["travel-sample:def_airport
try-cb-api-wss-7.1 | wait-for-couchbase: polling for '. | del(.indexer) | del(.["travel-sample:def_na
try-cb-api-wss-7.1 | wait-for-couchbase: polling for '. | del(.indexer) | map(.num_pending_requests =
try-cb-api-wss-7.1 | + exec kodev run
try-cb-fe-wss-7.1 | wait-for-it: backend:8080 is available after 11 seconds
try-cb-fe-wss-7.1 |
try-cb-fe-wss-7.1 | > try-cb-frontend-v2@0.1.0 serve
try-cb-fe-wss-7.1 | > vue-cli-service serve --port 8081
try-cb-fe-wss-7.1 |
try-cb-fe-wss-7.1 |  INFO  Starting development server...
try-cb-fe-wss-7.1 | Browserslist: caniuse-lite is outdated. Please run:
try-cb-fe-wss-7.1 |   npx browserslist@latest --update-db
try-cb-fe-wss-7.1 |   Why you should do it regularly: https://github.com/browserslist/browserslist#browsers-data-updating
try-cb-fe-wss-7.1 |  DONE  Compiled successfully in 5099ms3:07:47 PM
try-cb-fe-wss-7.1 |
try-cb-fe-wss-7.1 |
try-cb-fe-wss-7.1 |   App running at:
try-cb-fe-wss-7.1 |   - Local:   http://localhost:8081/
try-cb-fe-wss-7.1 |
try-cb-fe-wss-7.1 |   It seems you are running Vue CLI inside a container.
try-cb-fe-wss-7.1 |   Access the dev server via http://localhost:<your container's external mapped port>/
try-cb-fe-wss-7.1 |
try-cb-fe-wss-7.1 |   Note that the development build is not optimized.
try-cb-fe-wss-7.1 |   To create a production build, run npm run build.
try-cb-fe-wss-7.1 |
```

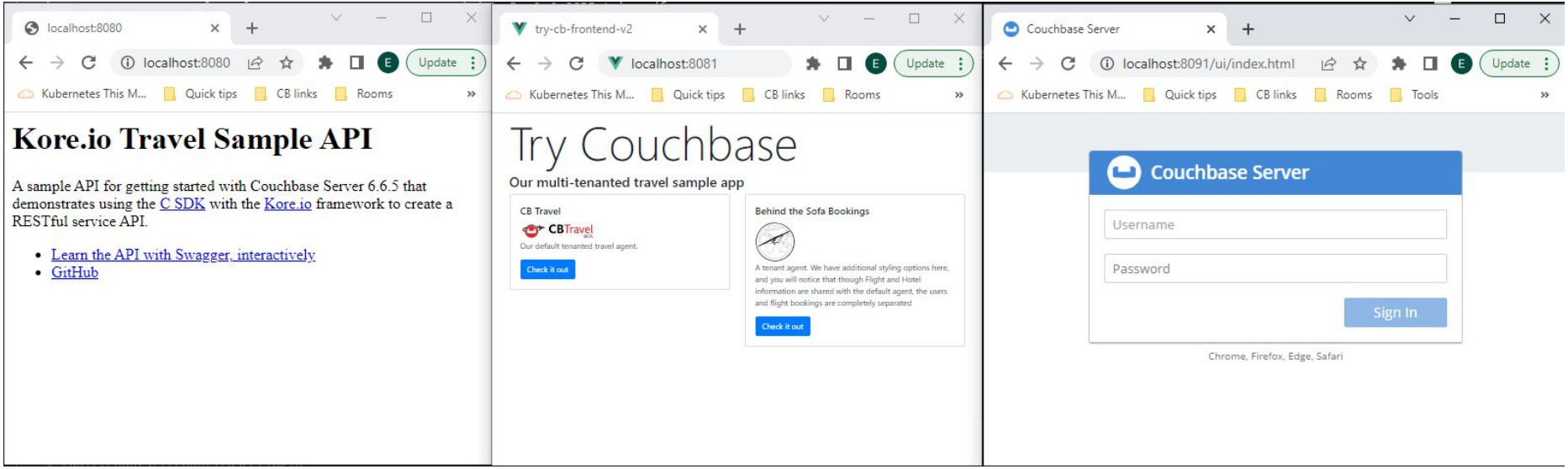
Verify Build Process

You should be able to access the the difference components on your localhost.

localhost:8080 - API

localhost:8081 - Web App

localhost:8091 - Couchbase Server UI

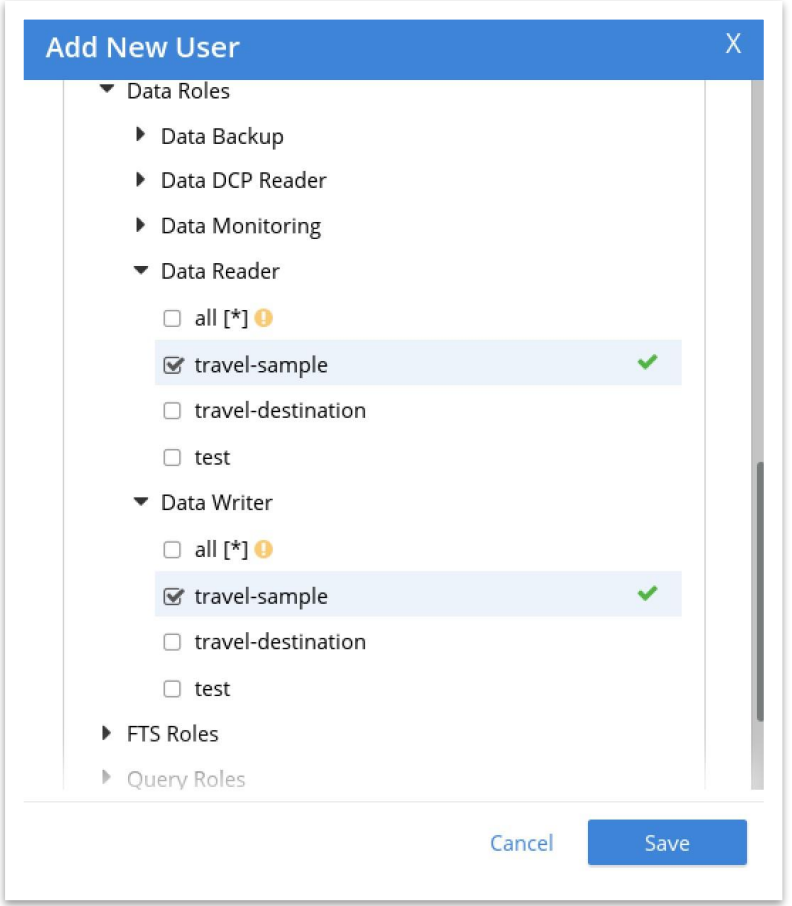


2 Managing Connections

Create an applicative User via RBAC

Perform the following operations:

- Browse to the Couchbase Server UI http://localhost:8091/
- Go the Security tab in Couchbase
- Create a new user.
- Username = “application”
- Password = “password”
- Roles
  - Data Reader on `travel-sample`
  - Data Writer on `travel-sample`
  - Query Select on `travel-sample`



Connecting to Couchbase with RBAC

In the try-cb-lcb-labs.c source code file, edit the kore_worker_configure() method:
Look for the "// LAB – Couchbase bootstrap" comments to implement the following:

- Configure the connection
- Create the instance
- Connect to the cluster
- Check the bootstrap status
- Install callbacks
- Open bucket

Edit the destroy_cb_instance() method to cleanup and destroy the couchbase connection instance:

Look for the "// LAB – Couchbase shutdown" comment to implement

- Destroy connection/instance

Rebuild the project

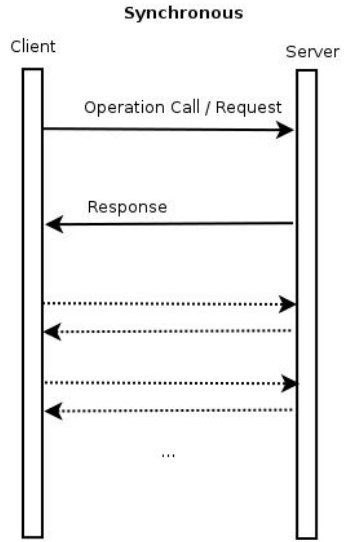
Test your implementation by calling the Test API: localhost:8080/api/test

This will perform a basic k/v GET from the `travel-sample` bucket.

3 Working with Documents

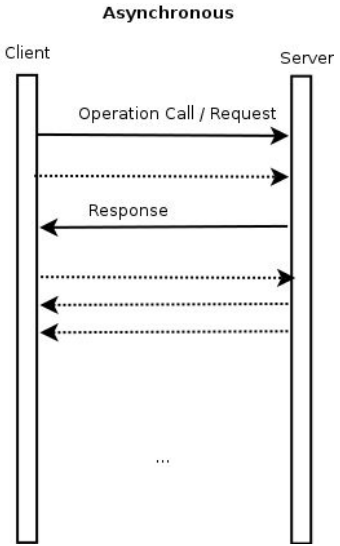
SDK API Blocking

- Libcouchbase is designed to use non-blocking I/O
  - Scheduled operations
- But lcb_wait() blocks by default
  - Waits for pending requests
  - Used for synchronous operation execution
- Callback functions are used
  - e.g. storage_callback



SDK API Non-Blocking

- External event loop integration
  - Provides mechanism to execute a callback function when a specific event occurs
- Asynchronous operation execution
- No need for lcb_wait()



Insert a Document

Make sure that the travel-sample data is installed!
Edit the api-user-auth.c source file:
Modify the insert_user() method to insert a new document:
Look for the "// LAB – Insert" comments to implement the following:

- Create the command
- Set the document ID
- Set the document value
- Store the document

We are running Couchbase 7+ don't forget to add functionality to specify the collection/scope!

Rebuild the project

Open your browser to the travel sample app website: http://localhost:8081/.

Choose the CBTravel application and register a new user account:

username: travel, password: 123456

Get a (Sub)Document

Edit the get_user_password() method:

Look for the "// LAB – Get Subdoc" comments to implement the following:

- Create the Command
- Specify the Document ID
- Create Subdoc Spec
- Specify the Subdoc Field to return
- Add Subdoc operations to command
- Run Subdoc operation

Don't forget to add functionality to specify the collection/scope!

Rebuild the project

Open your browser to the travel sample app website: http://localhost:8081/ and choose the CBTravel application.

In the front-end application, login using the user account you added in the previous step.

Create/Update a Document

Edit the api-user-flights.c source file:
Modify the upsert-new-flight() method to upsert a new document:
Look for the "// LAB – Upsert" comments to implement the following:

- Create the command
- Set the document ID
- Set the document value
- Store the document

Notice the similarity to Inserting a new document!

Unfortunately, this and the subsequent steps will not be testable until the N1QL query step has been completed.

Update a (Sub)Document

Modify the add_user_booking() method:
Look for the "// LAB – Update Subdoc" comments to implement the following:

- Create Command
- Specify the Document ID
- Create Subdoc Spec
- Specify the Subdoc Field to append to the array
- Add Subdoc operations to command
- Schedule Subdoc operation

Get a (Sub)Document

Modify the get_user_bookings() method:
Look for the "// LAB – Get Subdoc" comments to implement the following:

- Create Command
- Specify the Document ID
- Create Subdoc Spec
- Specify the Subdoc Field to return
- Add Subdoc operations to command
- Schedule Subdoc operation

Get a Document

Modify the get_flight_booking() method:
Look for the "// LAB – Get Document" comments to implement the following:

- Create Command
- Specify the Document ID
- Perform Get

4 | N1QL Queries

Query via SQL++

Make sure that the Secondary Indexes on *'faa'* and *'airportname'* are there!

Edit the api-flight-paths.c source file:

Modify the tcblcb-api-fpaths() method to query using positional parameters:

Look for the "// LAB – Position Param Query" comments to implement the following:

- Create the Query command
- Add the SQL++ query to the query command
- Add the positional parameters to the query
- Set query formatting
- Specify Adhoc = FALSE
- Set the callback function
- Send the query to the cluster

Query via SQL++

Continue modifying the tcblcb-api-fpaths() method to query using positional parameters:

Look for the "// LAB – Named Param Query" comments to implement the following:

- Reset the query command
- Create the Query command
- Add the SQL++ to the query command
- Add named parameters to the query
- Set query formatting
- Specify Adhoc = FALSE
- Set the callback function
- Send the query to the cluster

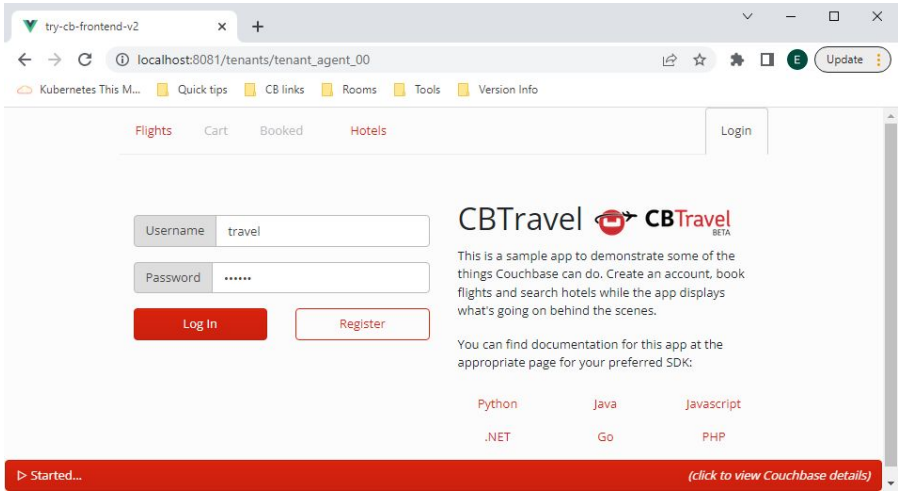
5 | Travel Sample Application

A Sample Application

Rebuild the project

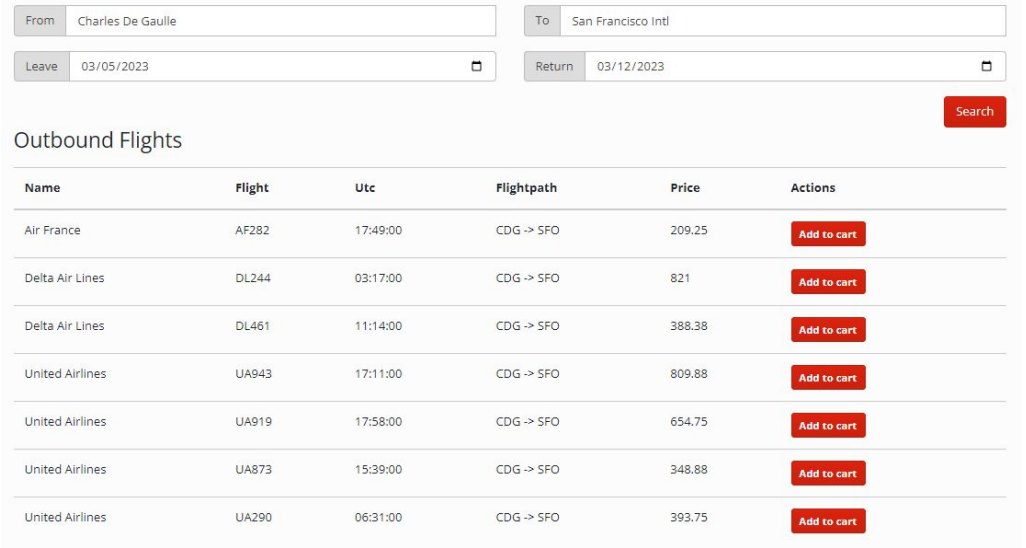
Open your browser to the travel sample app website: http://localhost:8081/ choose the CBTravel application.

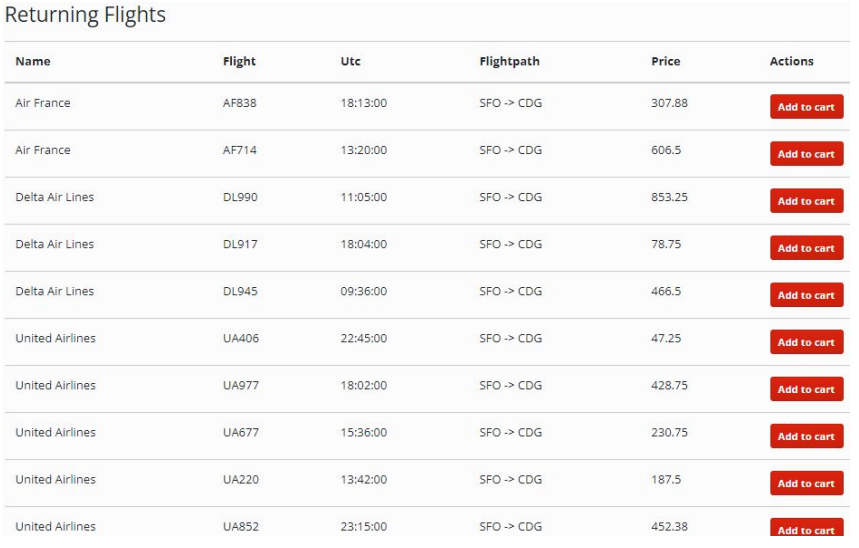
Log in using the ‘travel’ user you registered.



A Sample Application

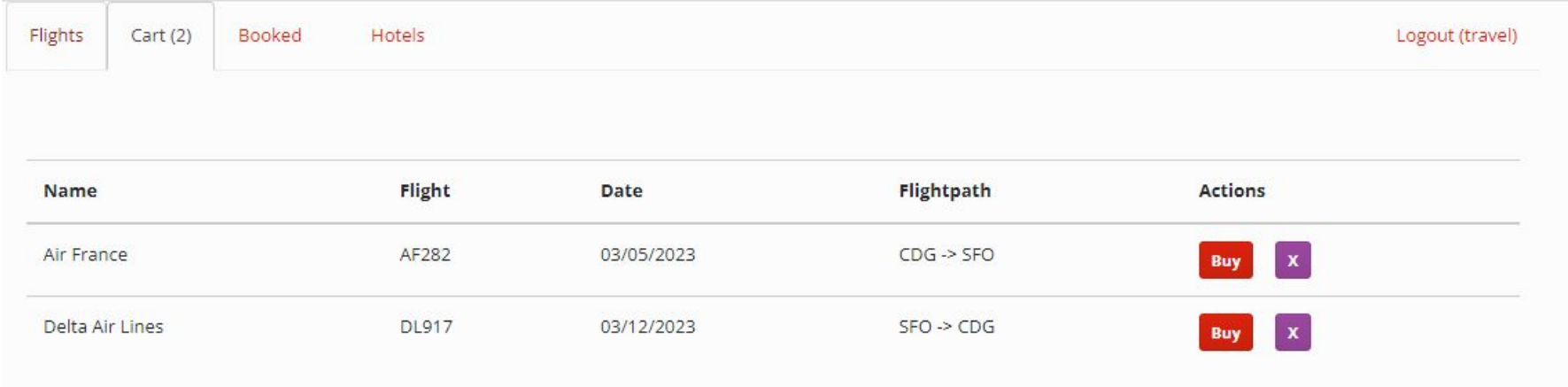
Search for flights from “Charles De Gaulle” to “San Francisco Intl” (or other airports of your choice, using the airport name). Specify travel dates. Select one or more flights from both the outbound and returning flights and add to cart.





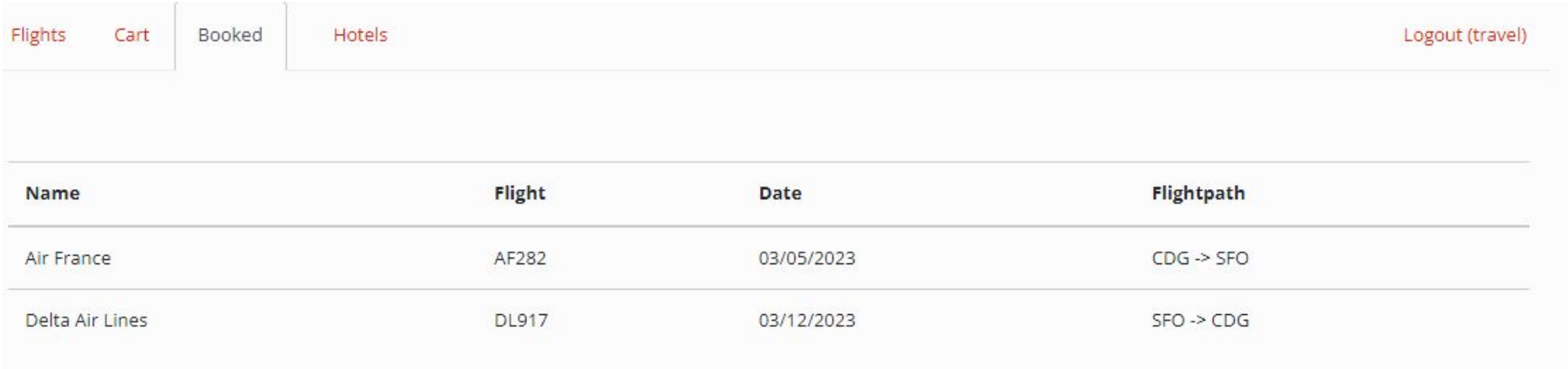
A Sample Application

In your cart, click to buy one or more flights.



A Sample Application

The purchased flights should appear in your booked flights page.



Thank you