Þróun Hugbúnaðar

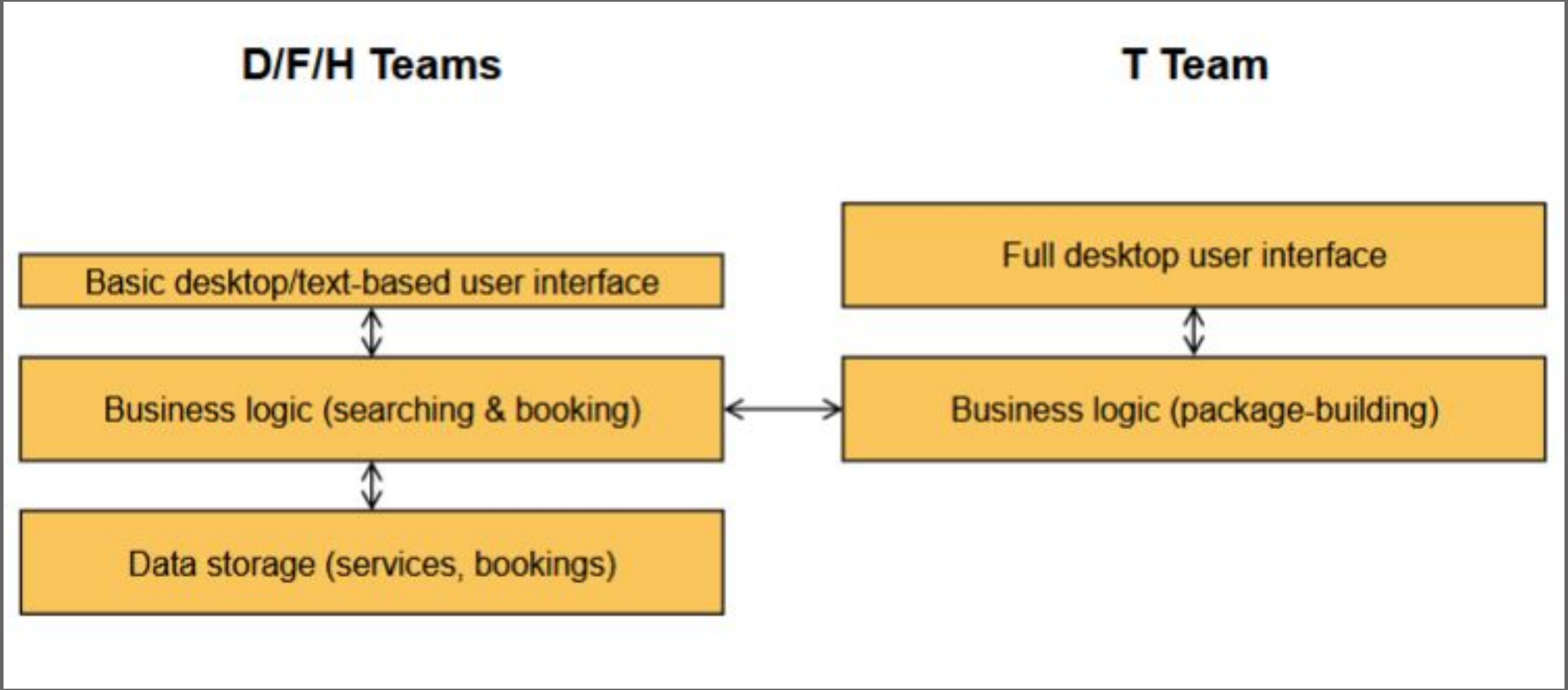
Cluster 2

Demonstration of the Travel planner application

Let us switch windows real quick.

Overview of the system architectures

Project Architecture that we followed


D/F/H Teams
T Team
Basic desktop/text-based user interface
Business logic (searching & booking)
Data storage (services, bookings)
Full desktop user interface
Business logic (package-building)

Core Architecture

Frontend

- Javafx: Application and UI
- Maven: Structure

Backend

- SQLite3: Database
- Java: BusinessLogic and connecting databases

Group D - Daytour Search

Daytour System architecture pt. 1

- One database
- Four controllers:
- HomeController
- TourController
- CheckoutController
- ReceiptController
- Four model components:
- User
- Tour
- Tourlisti
- Receipt

Daytour System architecture pt. 2

- Four visual components:
- Heima-view
- Tour-view
- Checkout-view
- Receipt-view
- Each controller is responsive to user input in a visual component
- Model components have classes that store information
- That information is stored in the database

Group F - Flight search

Integrated Product

Project Goal:

- Build a simple flight booking system with:
    - Flight search
    - Booking and cancellation
    - Seat tracking (increment/decrement)
    - SQLite database integration

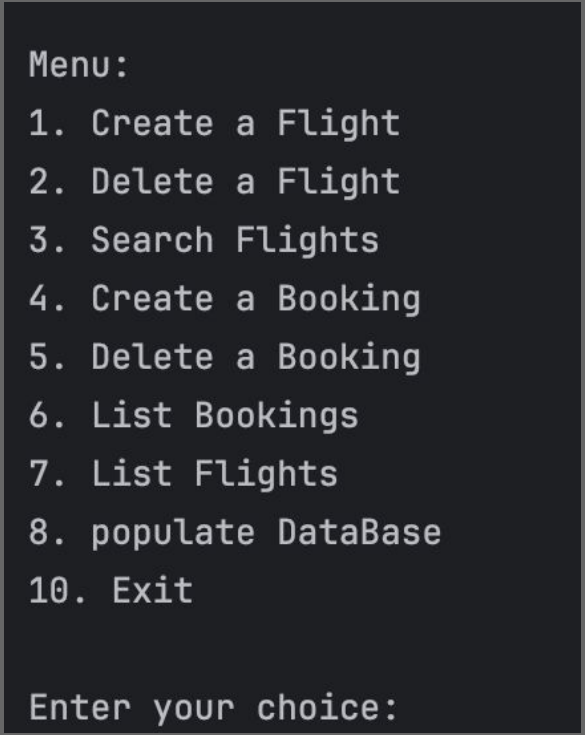
Overall Structure

MainApp: Terminal-based UI

FlightRepo / BookingRepo: Database layer (SQLite)

FlightService / BookingService: Logic/controllers

Flight / Booking / User: Data models



```
Menu:
1. Create a Flight
2. Delete a Flight
3. Search Flights
4. Create a Booking
5. Delete a Booking
6. List Bookings
7. List Flights
8. populate DataBase
10. Exit

Enter your choice:
```

Database (SQLite)

Two database files:

- flights.db: stores flight information
- bookings.db: stores user bookings
- Tables created if they don’t exist (CREATE TABLE IF NOT EXISTS)
- Database actions like addFlight, confirmBooking, deleteBooking, etc.

Flights

Each flight has:

- `flightID`, `date`, `destination`, `status`, `availableSeats`

Users can:

- Add, delete, and search flights
- Book seats (decrement), cancel bookings (increment)

Bookings

Bookings use a unique ID: flightID-userID

Safety check to prevent duplicate or overbooked seats

Integration with seat count in flight database

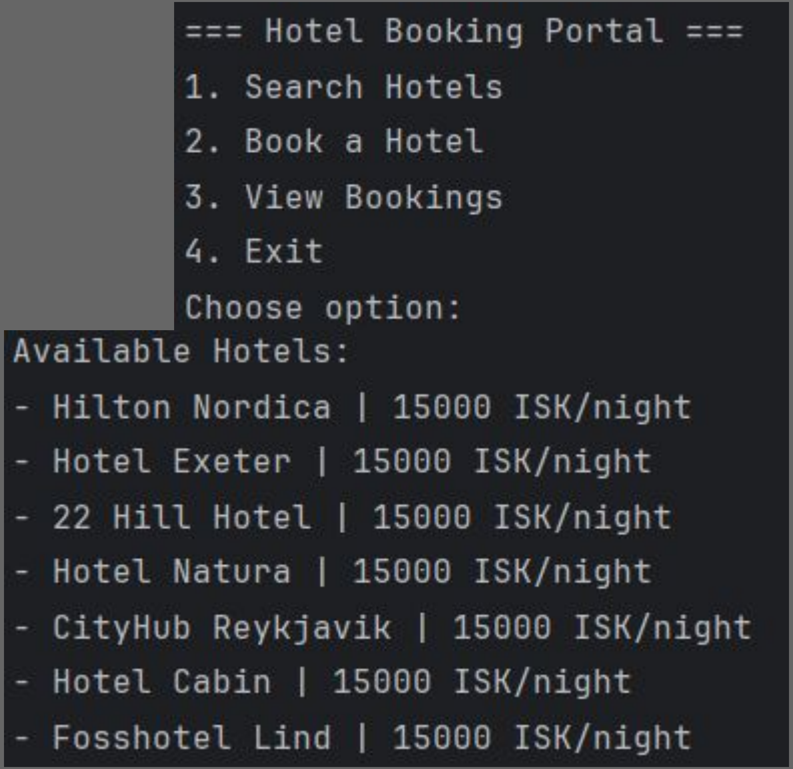
Edge cases handled:

- Two users trying to book the last seat
- Booking a flight that doesn't exist
- Canceling a booking that doesn’t exist

Group H - Hotel search

Functionality

- Simple command line interface
- Enter location, number of guests, check-in- and check-out date to search
- Enter name of hotel, number of guests, check-in- and check-out date to book



```
=== Hotel Booking Portal ===
1. Search Hotels
2. Book a Hotel
3. View Bookings
4. Exit
Choose option:
Available Hotels:
- Hilton Nordica | 15000 ISK/night
- Hotel Exeter | 15000 ISK/night
- 22 Hill Hotel | 15000 ISK/night
- Hotel Natura | 15000 ISK/night
- CityHub Reykjavik | 15000 ISK/night
- Hotel Cabin | 15000 ISK/night
- Fosshotel Lind | 15000 ISK/night
```

Hotel

Constructor:

Hotel(int id, String name, String location, int totalBeds, int pricePerNight)

Booking

Constructor:

Booking(int hotelId, String hotelName, int guests, LocalDate checkIn, LocalDate checkOut)

Hotel repo

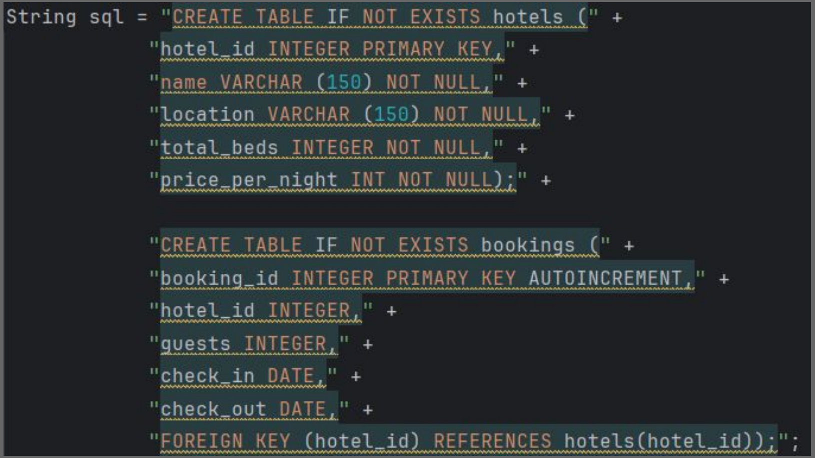
- List<Hotel> searchHotels(String location)
- Hotel getHotelByName(String name)

Booking repo

- int createBooking(int hotelId, int guests, LocalDate checkIn, LocalDate checkOut)
- void countBookedbeds(int hotelId, LocalDate checkIn, LocalDate checkOut)
- List<Booking> getAllBookings()

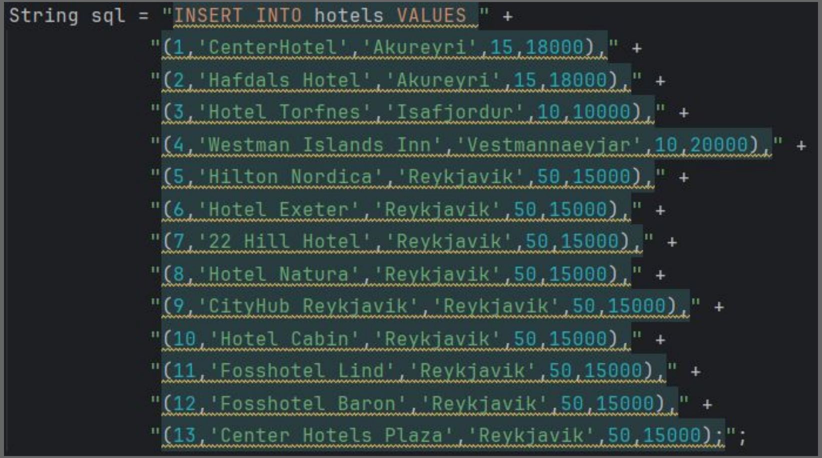
DBHelper

- private static void initializeDatabase(Connection conn)
- private static void createTables(Connection conn)
- private static void seedDataIfEmpty(Connection conn)
- private static void seedHotelsData(Connection conn)



```
String sql = "CREATE TABLE IF NOT EXISTS hotels (" +
            "hotel_id INTEGER PRIMARY KEY," +
            "name VARCHAR (150) NOT NULL," +
            "location VARCHAR (150) NOT NULL," +
            "total_beds INTEGER NOT NULL," +
            "price_per_night INT NOT NULL);" +

            "CREATE TABLE IF NOT EXISTS bookings (" +
            "booking_id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "hotel_id INTEGER," +
            "guests INTEGER," +
            "check_in DATE," +
            "check_out DATE," +
            "FOREIGN KEY (hotel_id) REFERENCES hotels(hotel_id));";
```



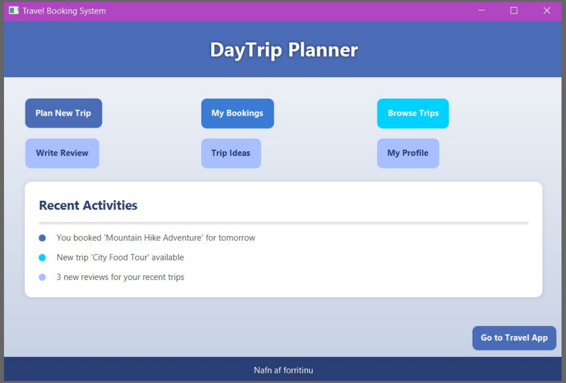
```
String sql = "INSERT INTO hotels VALUES " +
            "(1,'CenterHotel','Akureyri',15,18000)," +
            "(2,'Hafdals Hotel','Akureyri',15,18000)," +
            "(3,'Hotel Torfnes','Isafjordur',10,10000)," +
            "(4,'Westman Islands Inn','Vestmannaeyjar',10,20000)," +
            "(5,'Hilton Nordica','Reykjavik',50,15000)," +
            "(6,'Hotel Exeter','Reykjavik',50,15000)," +
            "(7,'22 Hill Hotel','Reykjavik',50,15000)," +
            "(8,'Hotel Natura','Reykjavik',50,15000)," +
            "(9,'CityHub Reykjavik','Reykjavik',50,15000)," +
            "(10,'Hotel Cabin','Reykjavik',50,15000)," +
            "(11,'Fosshotel Lind','Reykjavik',50,15000)," +
            "(12,'Fosshotel Baron','Reykjavik',50,15000)," +
            "(13,'Center Hotels Plaza','Reykjavik',50,15000);";
```

Team T

Combined the data and functionality from flights, hotels and day tours and created a uniform standard service wrapper to ease the use inside of our module.

Implemented a custom meta-search engine which uses modular functionality to accurately search the representing data set.

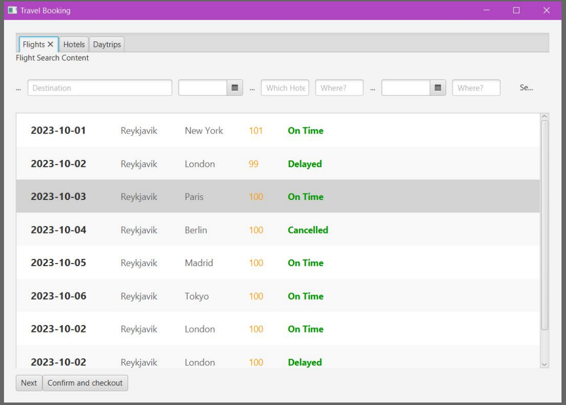
Presented with a usable user interface.



Recent Activities

- You booked 'Mountain Hike Adventure' for tomorrow
- New trip 'City Food Tour' available
- 3 new reviews for your recent trips

Nafn af forritinu



2023-10-01	Reykjavik	New York	101	On Time
2023-10-02	Reykjavik	London	99	Delayed
2023-10-03	Reykjavik	Paris	100	On Time
2023-10-04	Reykjavik	Berlin	100	Cancelled
2023-10-05	Reykjavik	Madrid	100	On Time
2023-10-06	Reykjavik	Tokyo	100	On Time
2023-10-02	Reykjavik	London	100	On Time
2023-10-02	Reykjavik	London	100	Delayed

Retrospective

Did we reach our goal?

YES (mostly)

What went well

- Implementation of features
- Facing challenges, finding solutions

What could been improved

- Time management
- Communication
- Better documentation during development

Main takeaway and biggest lessons

- Start ASAP
- Communication is KEY
- Make realistic plans and schedules
- Think from different perspectives
- Teamwork makes the dream work

Q&A