Planit

Go anywhere on the planet with planit.

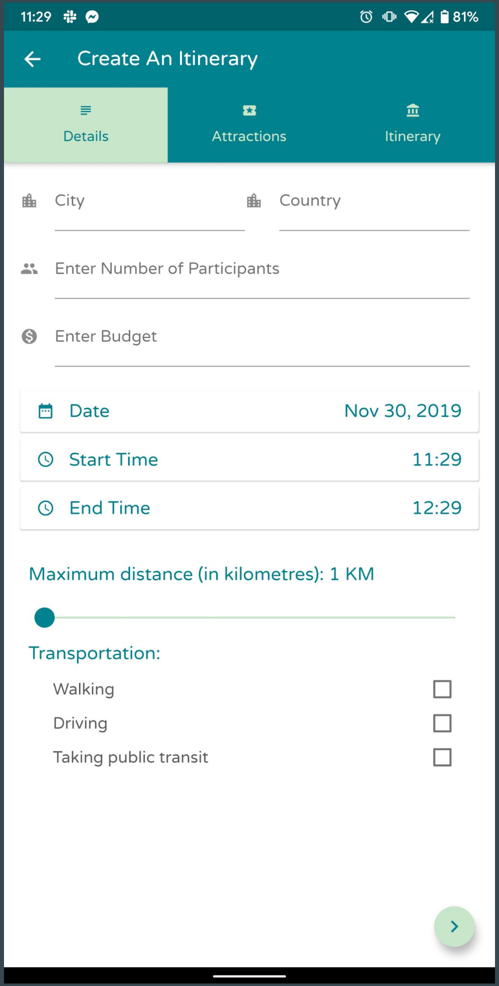
•••

Team Planners

Team

- Sakina Gadriwala, *Scrum Master*
- Kemar Harris, *Project Manager*
- Dineshan Pathmanathan, *Project Manager*
- Saad Shah, *Visual Design*
- Seemin Syed, *Visual Design*
- Jaya Thirugnanasampanthan, *Business Analyst*

What is planit?



11:29 81%

Create An Itinerary

City

Country

Enter Number of Participants

Enter Budget

Date Nov 30, 2019

Start Time 11:29

End Time 12:29

Maximum distance (in kilometres): 1 KM

Transportation:

Walking ☐

Driving ☐

Taking public transit ☐

¯_(ツ)_/¯ Why should we care?

- Save users’ time from having to search for events
- Have various services available on one platform
- Introduce users to new events that align with their preferences

Competitive Analysis

Global competition

- TripAdvisor
- Expedia

Local competition

- Other CSCC01 groups working on planit

Key Features

- Feature 1: Generating the itinerary
- Feature 2: Searching and changing the itinerary
- Feature 3: Viewing previous itineraries

Process

Highlights

- Communication via Slack & Google Hangouts
- Using proper github branching structure
- Using JIRA to track work

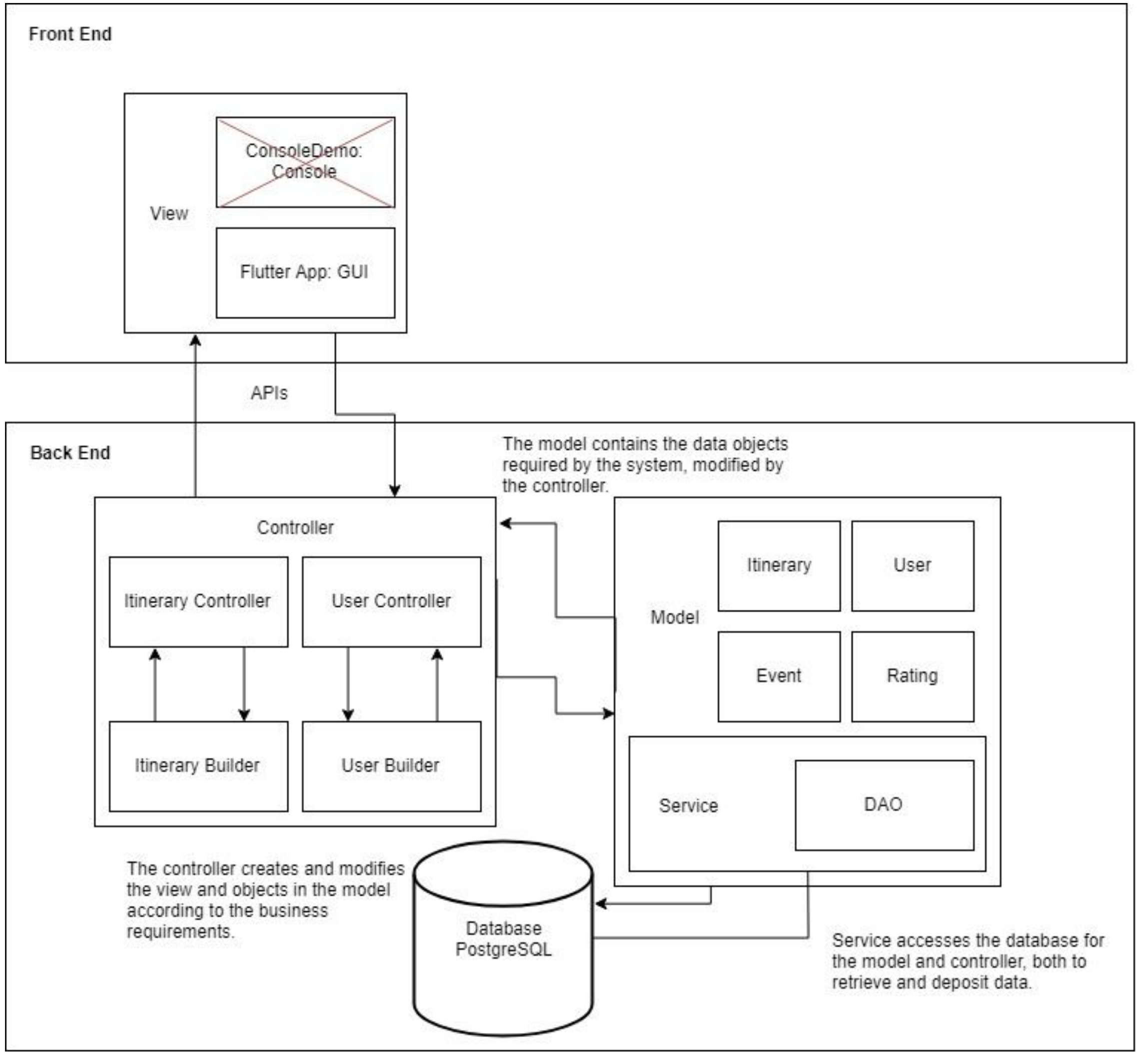
Difficulties

- Lack of a common schedule
- Foursquare API
- Adjusting to agile practices
- Learning Flutter

Techniques

- Holding Sprint Retrospectives
- Split code into front end and back end
- Following the MVC Design Pattern

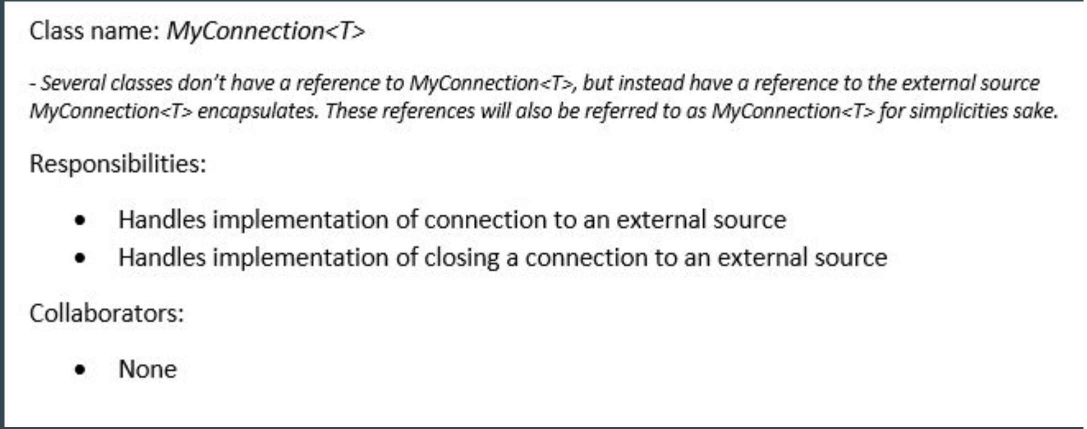
Software Architecture



The model contains the data objects required by the system, modified by the controller.

The controller creates and modifies the view and objects in the model according to the business requirements.

Service accesses the database for the model and controller, both to retrieve and deposit data.



Class name: *MyConnection<T>*

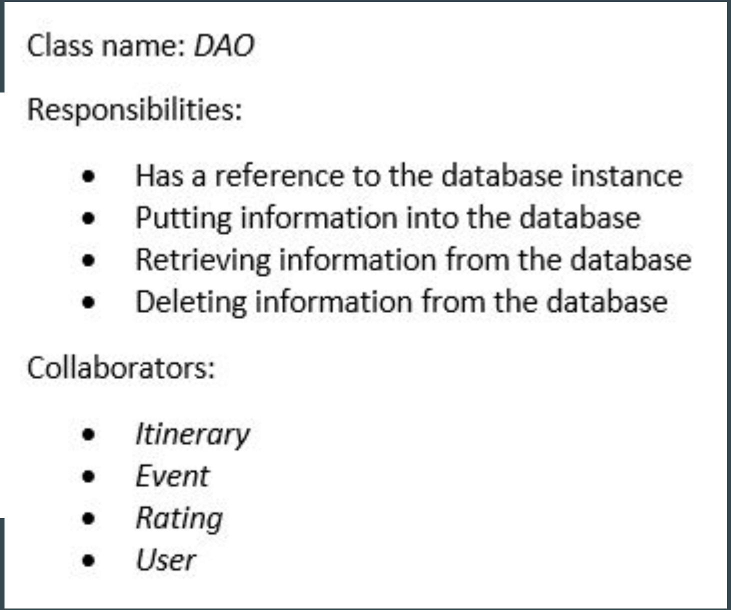
- Several classes don't have a reference to MyConnection<T>, but instead have a reference to the external source MyConnection<T> encapsulates. These references will also be referred to as MyConnection<T> for simplicities sake.

Responsibilities:

- Handles implementation of connection to an external source
- Handles implementation of closing a connection to an external source

Collaborators:

- None



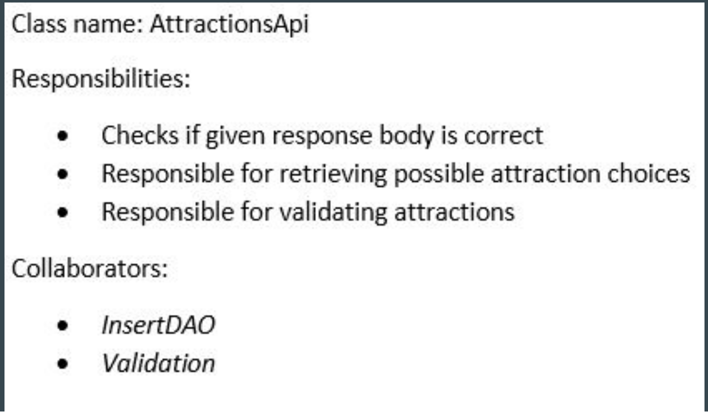
Class name: *DAO*

Responsibilities:

- Has a reference to the database instance
- Putting information into the database
- Retrieving information from the database
- Deleting information from the database

Collaborators:

- *Itinerary*
- *Event*
- *Rating*
- *User*

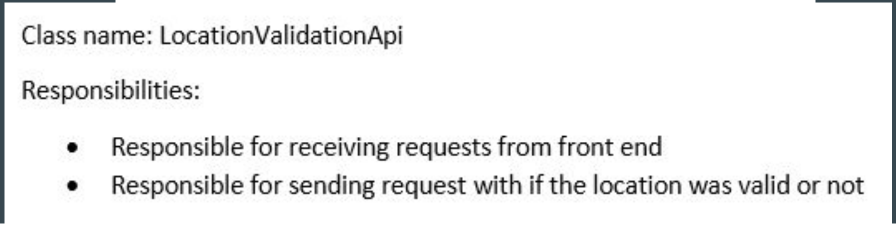

Class name: AttractionsApi

Responsibilities:

- Checks if given response body is correct
- Responsible for retrieving possible attraction choices
- Responsible for validating attractions

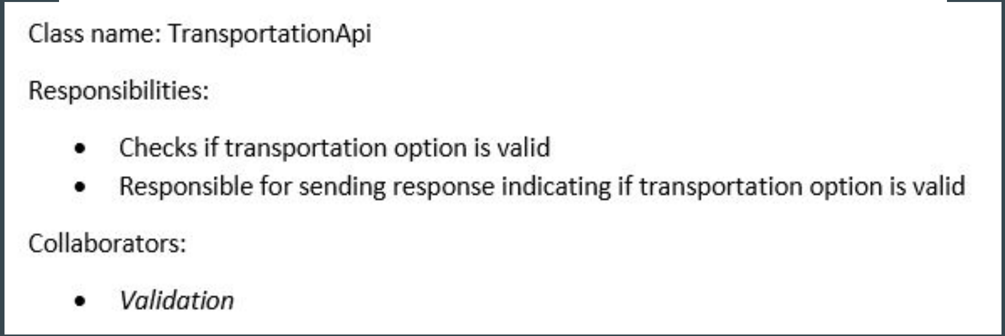
Collaborators:

- *InsertDAO*
- *Validation*


Class name: LocationValidationApi

Responsibilities:

- Responsible for receiving requests from front end
- Responsible for sending request with if the location was valid or not

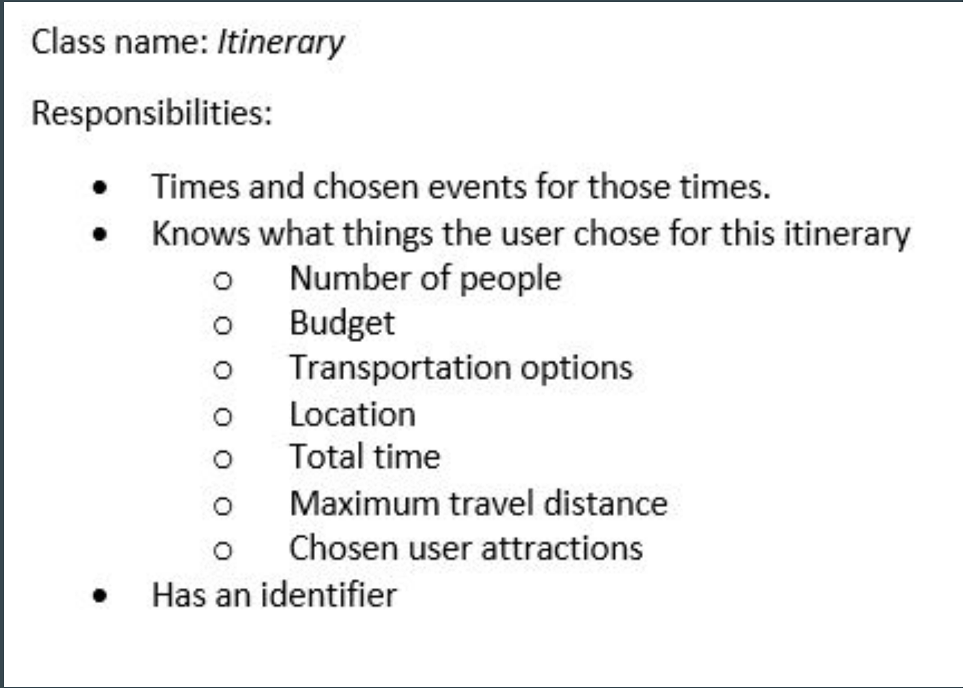

Class name: TransportationApi

Responsibilities:

- Checks if transportation option is valid
- Responsible for sending response indicating if transportation option is valid

Collaborators:

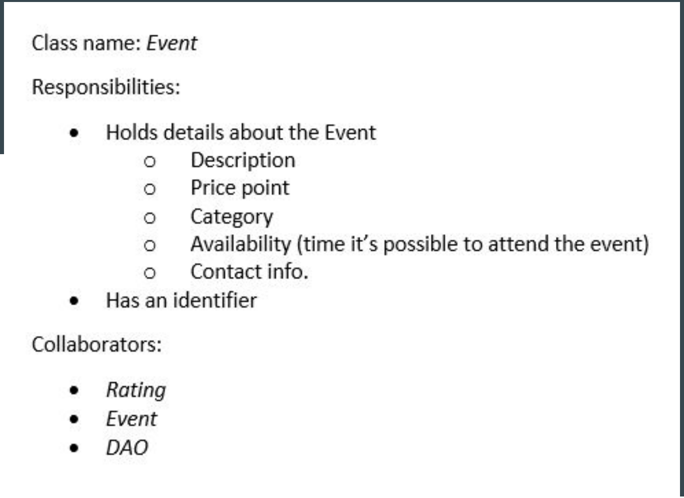
- *Validation*



Class name: *Itinerary*

Responsibilities:

- Times and chosen events for those times.
- Knows what things the user chose for this itinerary
    - Number of people
    - Budget
    - Transportation options
    - Location
    - Total time
    - Maximum travel distance
    - Chosen user attractions
- Has an identifier



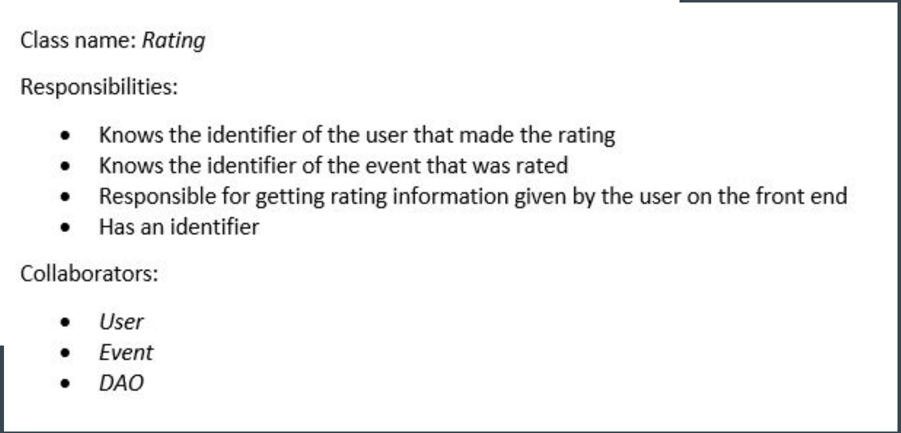
Class name: *Event*

Responsibilities:

- Holds details about the Event
    - Description
    - Price point
    - Category
    - Availability (time it's possible to attend the event)
    - Contact info.
- Has an identifier

Collaborators:

- *Rating*
- *Event*
- *DAO*



Class name: *Rating*

Responsibilities:

- Knows the identifier of the user that made the rating
- Knows the identifier of the event that was rated
- Responsible for getting rating information given by the user on the front end
- Has an identifier

Collaborators:

- *User*
- *Event*
- *DAO*

Technologies used during developing planit

Java

Android Virtual Device

Postman

Flutter

PostgreSQL

Flutter

- Open-source User Interface software development kit used to develop multi-platform applications
- Created by Google

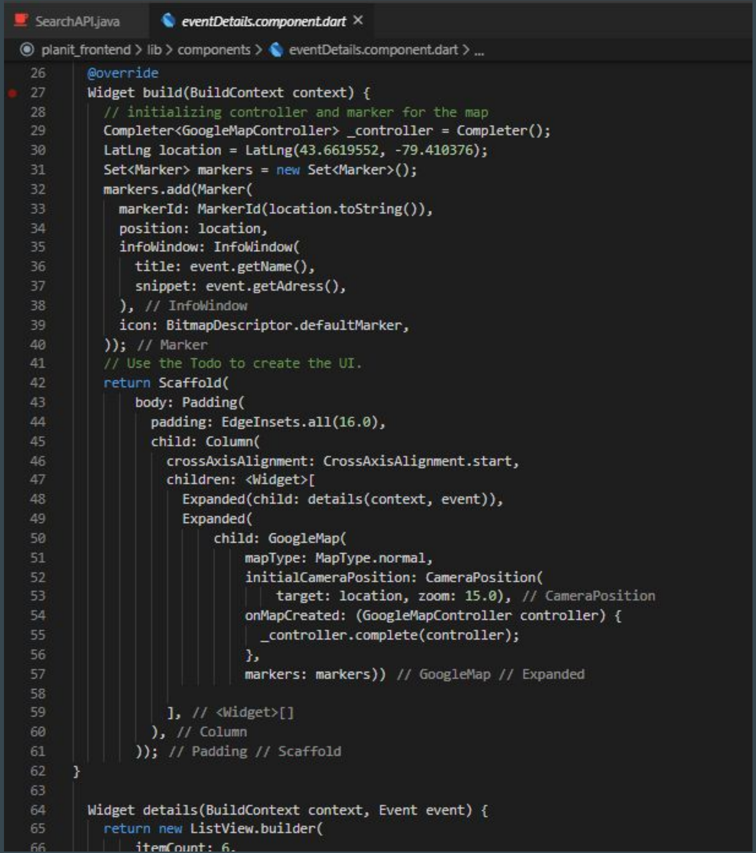
Technical Challenges

- Learning curve
- Compilation required a lot system resources

... and Their Solutions

- Reading documentation and asking other team members
- Running code on Android Devices

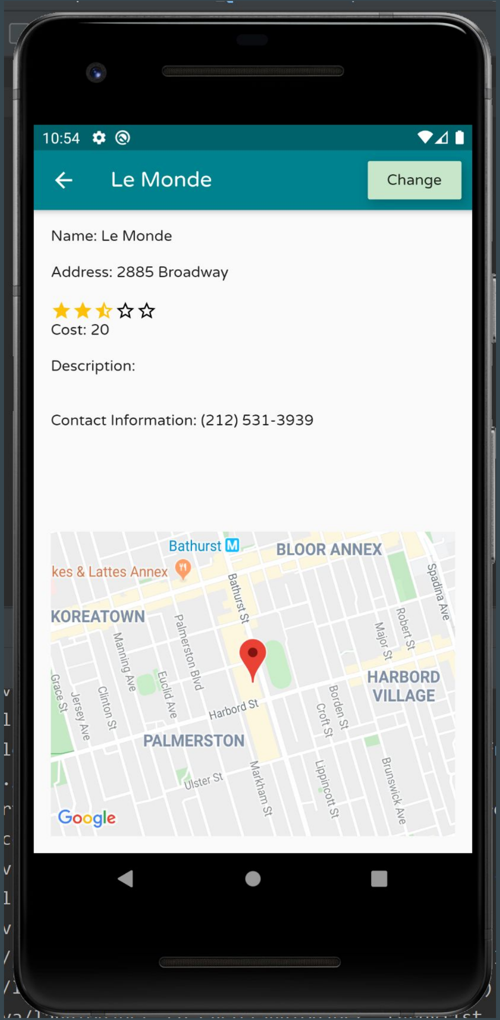
Flutter


SearchAPI.java
eventDetails.component.dart ×
planit_frontend > lib > components > eventDetails.component.dart > ...

```
 @override
 Widget build(BuildContext context) {
  // initializing controller and marker for the map
  Completer<GoogleMapController> _controller = Completer();
  LatLng location = LatLng(43.6619552, -79.410376);
  Set<Marker> markers = new Set<Marker>();
  markers.add(Marker(
   markerId: MarkerId(location.toString()),
   position: location,
   infoWindow: InfoWindow(
    title: event.getName(),
    snippet: event.getAdress(),
   ), // InfoWindow
   icon: BitmapDescriptor.defaultMarker,
  )); // Marker
  // Use the Todo to create the UI.
  return Scaffold(
    body: Padding(
     padding: EdgeInsets.all(16.0),
     child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: <Widget>[
       Expanded(child: details(context, event)),
       Expanded(
         child: GoogleMap(
           mapType: MapType.normal,
           initialCameraPosition: CameraPosition(
             target: location, zoom: 15.0), // CameraPosition
           onMapCreated: (GoogleMapController controller) {
            _controller.complete(controller);
           },
           markers: markers)) // GoogleMap // Expanded

      ], // <Widget>[]
     ), // Column
    )); // Padding // Scaffold
}

 Widget details(BuildContext context, Event event) {
  return new ListView.builder(
   itemCount: 6,
```



Name: Le Monde

Address: 2885 Broadway

★★★☆☆
Cost: 20

Description:

Contact Information: (212) 531-3939

Java

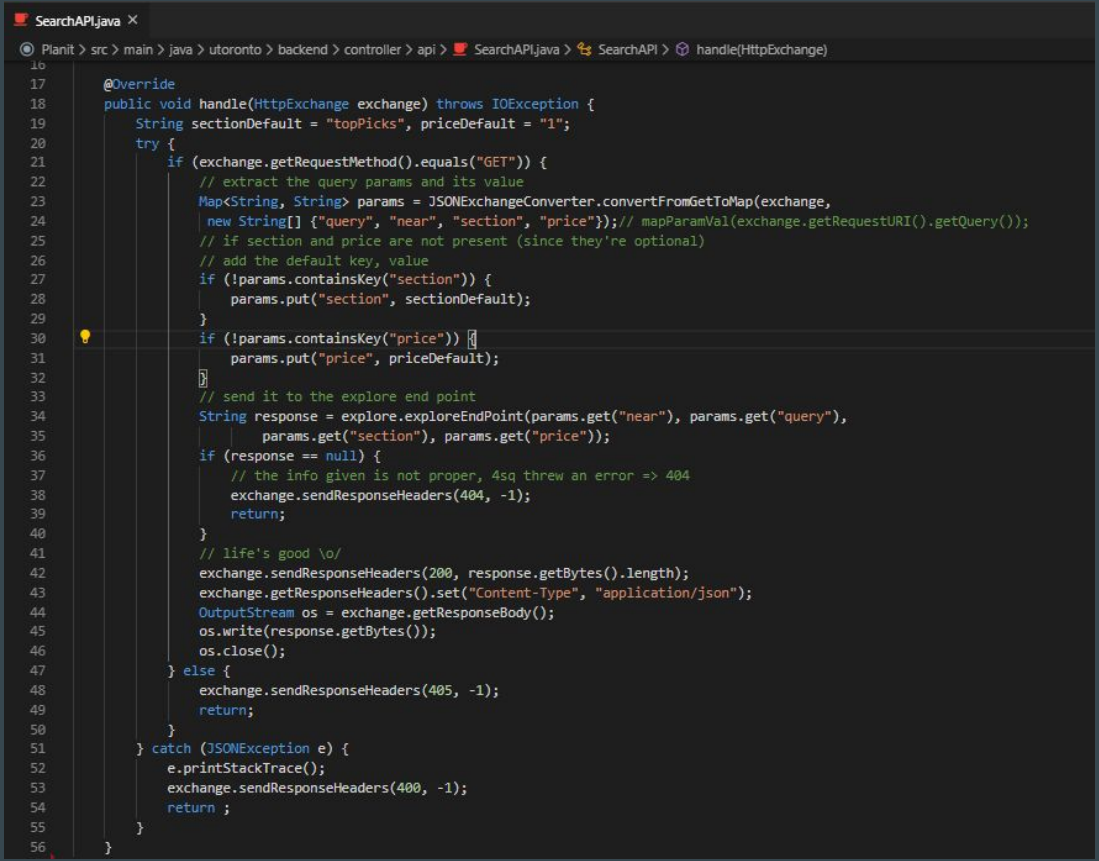
- Java is a object-oriented programming language
- Used to develop the project's APIs

Technical Challenges

- Refactoring GET APIs to accept query parameters

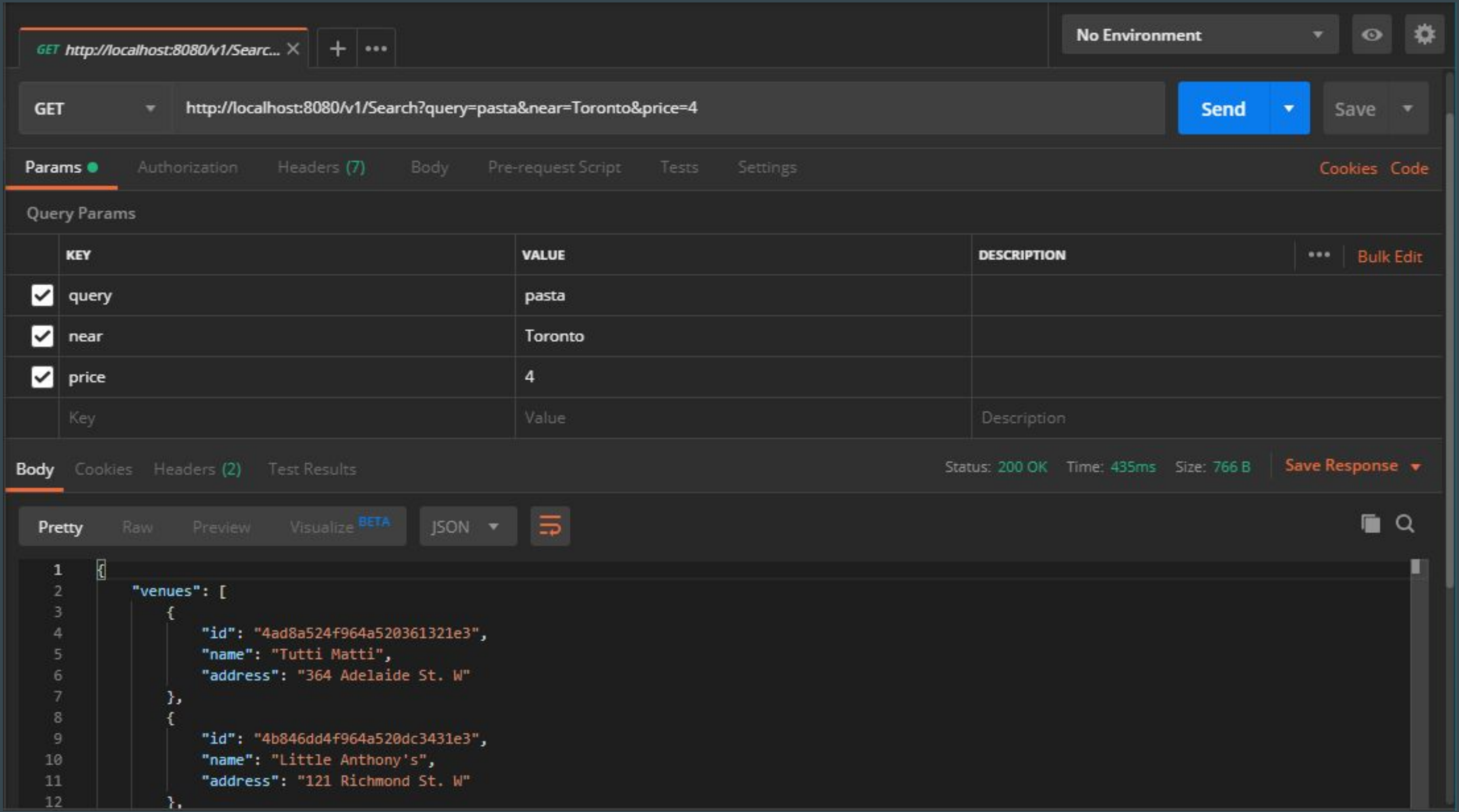
... and Their Solutions

- Reading documentation of HttpHandler
- Researching how other people implemented it through Stackoverflow


SearchAPI.java
Planit > src > main > java > utoronto > backend > controller > api > SearchAPI.java > SearchAPI > handle(HttpExchange)

```java
    @Override
    public void handle(HttpExchange exchange) throws IOException {
        String sectionDefault = "topPicks", priceDefault = "1";
        try {
            if (exchange.getRequestMethod().equals("GET")) {
                // extract the query params and its value
                Map<String, String> params = JSONExchangeConverter.convertFromGetToMap(exchange,
                 new String[] {"query", "near", "section", "price"});// mapParamVal(exchange.getRequestURI().getQuery());
                // if section and price are not present (since they're optional)
                // add the default key, value
                if (!params.containsKey("section")) {
                    params.put("section", sectionDefault);
                }
                if (!params.containsKey("price")) {
                    params.put("price", priceDefault);
                }
                // send it to the explore end point
                String response = explore.exploreEndPoint(params.get("near"), params.get("query"),
                        params.get("section"), params.get("price"));
                if (response == null) {
                    // the info given is not proper, 4sq threw an error => 404
                    exchange.sendResponseHeaders(404, -1);
                    return;
                }
                // life's good \o/
                exchange.sendResponseHeaders(200, response.getBytes().length);
                exchange.getResponseHeaders().set("Content-Type", "application/json");
                OutputStream os = exchange.getResponseBody();
                os.write(response.getBytes());
                os.close();
            } else {
                exchange.sendResponseHeaders(405, -1);
                return;
            }
        } catch (JSONException e) {
            e.printStackTrace();
            exchange.sendResponseHeaders(400, -1);
            return ;
        }
    }
```

Postman



GET http://localhost:8080/v1/Searc...

No Environment

GET http://localhost:8080/v1/Search?query=pasta&near=Toronto&price=4

Query Params

KEY	VALUE	DESCRIPTION
✓ query	pasta	
✓ near	Toronto	
✓ price	4	
Key	Value	Description

Status: 200 OK Time: 435ms Size: 766 B

```
{
    "venues": [
        {
            "id": "4ad8a524f964a520361321e3",
            "name": "Tutti Matti",
            "address": "364 Adelaide St. W"
        },
        {
            "id": "4b846dd4f964a520dc3431e3",
            "name": "Little Anthony's",
            "address": "121 Richmond St. W"
        },
```

PostgreSQL

- PostgreSQL is an object-relational, open source database management system

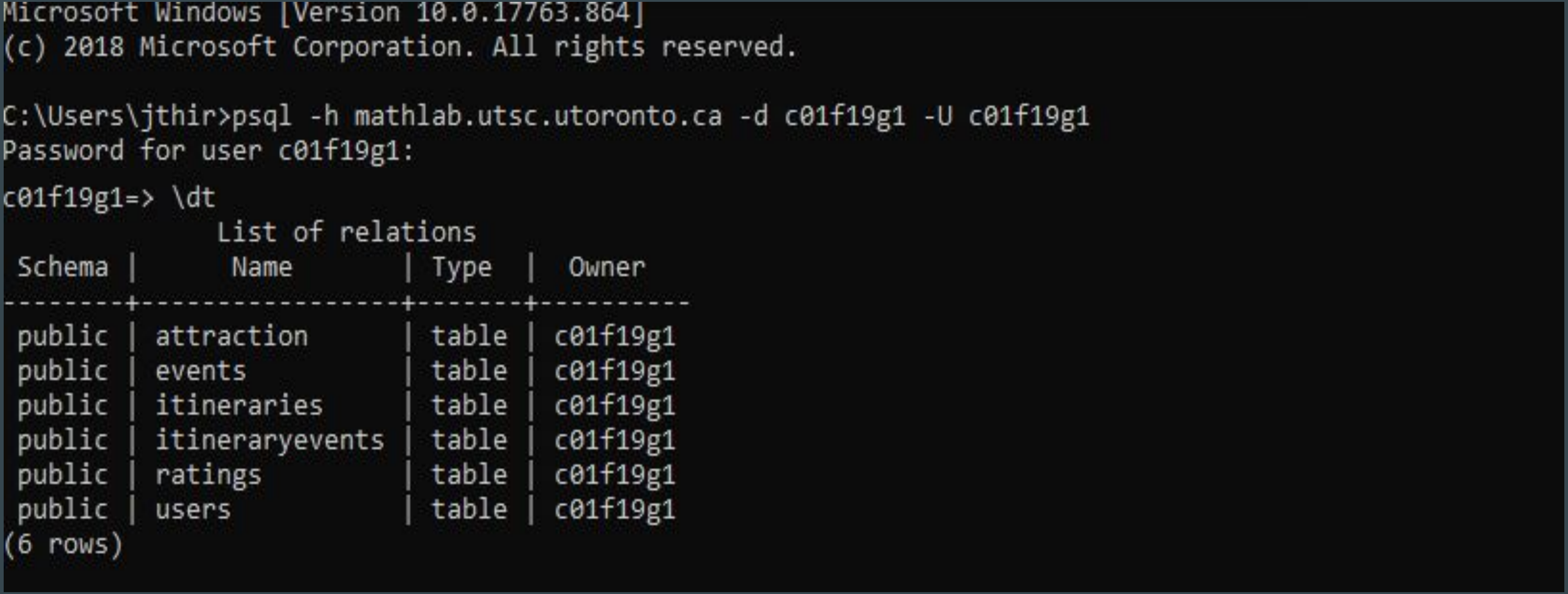
Technical Challenges

- Accessing a remote database server off-campus

... and Their Solutions

- Communicating with System Administrator to allow our off-campus IP address to bypass the system

PostgreSQL



```
Microsoft Windows [Version 10.0.17763.864]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\jthir>psql -h mathlab.utsc.utoronto.ca -d c01f19g1 -U c01f19g1
Password for user c01f19g1:
c01f19g1=> \dt
              List of relations
 Schema |       Name       | Type  |  Owner
--------+------------------+-------+----------
 public | attraction       | table | c01f19g1
 public | events           | table | c01f19g1
 public | itineraries      | table | c01f19g1
 public | itineraryevents  | table | c01f19g1
 public | ratings          | table | c01f19g1
 public | users            | table | c01f19g1
(6 rows)
```