

ExpoSure

Your Companion for Expos and Events. Having the power to catalogue exhibition products while making them at the finger tips of visitors who can browse and save their favorite products on their mobiles

MOBILE
DEVELOPMENT
PROJECT PROPOSAL

TABLE OF CONTENTS

Introduction	1
Functionality	1
The App is divided into three work flows.	1
Organizers	1
Businesses	1
Visitors	1
Mockup.....	2
Level 1	2
Class UML Diagram.....	3
Entity Relationship Diagram	4
System Architecture	5
Back End Code Snippets.....	6
General View	6
Setting up Express Server	7
Route Code Snippet Example.....	8
Model Code Snippet	9
Controller Snippet	10
View Snippets (Android Application)	11
General View	11
REST CLIENT API.....	12
API Interface Snippet	13
Adapter Snippet	14
Android Model Snippet.....	15
Activity Snippet	16
Database Script Sample	17
Application ScreenShots	18
List All Events	18
Creating New Events	19

INTRODUCTION

ExpoSure is an app for businesses, event organizers and visitors alike. Events are listed together with details such as location and dates giving businesses the ability of displaying their products to visitors. Visitors can save a product to favorites before hand so that they get to have a shortlist of their coveted products to see during the event. This way they will know the booth number associated with their favorite products saving time and focusing on what matters. Moreover, they can use their cameras to quickly add products to their favorite list on location. Organizers can coordinate with businesses making event guides that help visitors find booths they desire.

FUNCTIONALITY

THE APP IS DIVIDED INTO THREE WORK FLOWS.

ORGANIZERS

The Organizer role is to create events. They can view a list of events. Also, organizers have the ability to create businesses and add them to events.

BUSINESSES

Businesses can create products but cannot create events. Each product can be identified by a QR-Code. They can view a list of all their products.

VISITORS

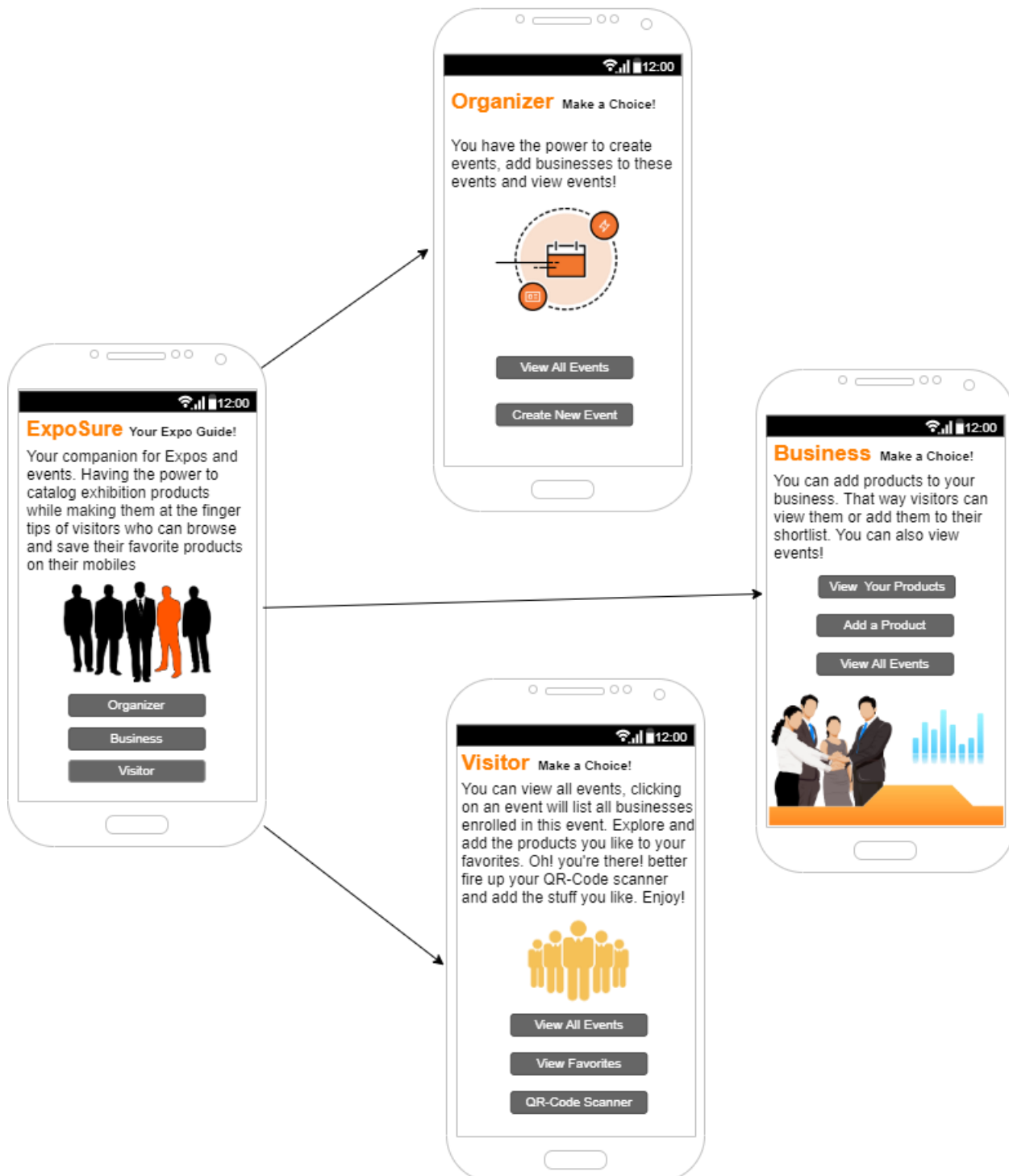
Visitors are the main focus of this app; they are capable of viewing events in a list. Each event informs the visitor of the venue's location and event date. They can browse for businesses enrolled in events and furthermore, browse the products belonging to each business. They can do this beforehand. This way they can plan their visit carefully ahead of time. Visitors can also, use their phone cameras to scan a QR-Code sticker provided by the business owners to add their chosen products to favorites.

MOCKUP

LEVEL 1

ExpoSure Level 1 Mobile Mockup

Amr Abdallah - C0744378

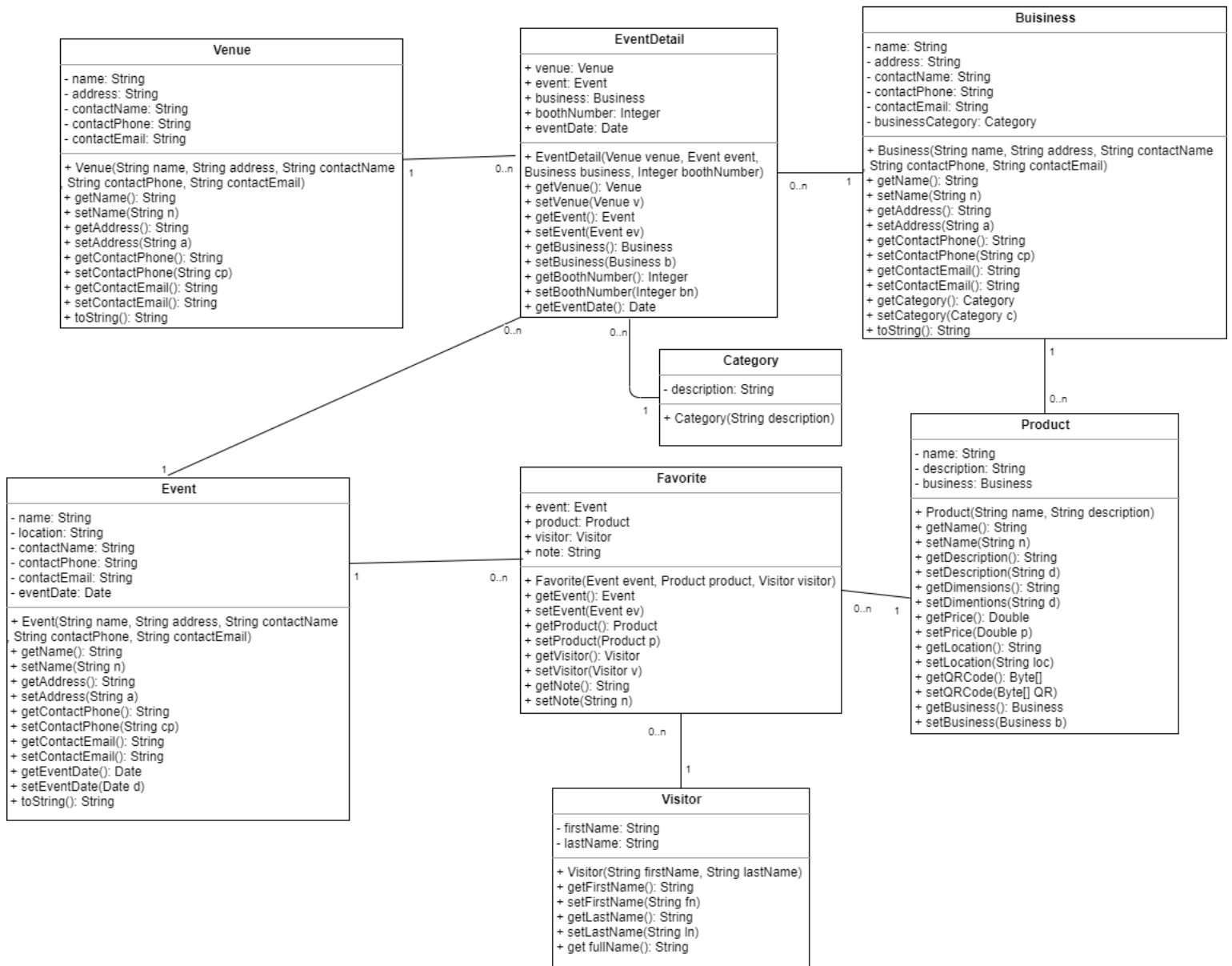


CLASS UML DIAGRAM

ExpoSure UML

Your Guide For Expos And Events

Amr Abdallah - C0744378

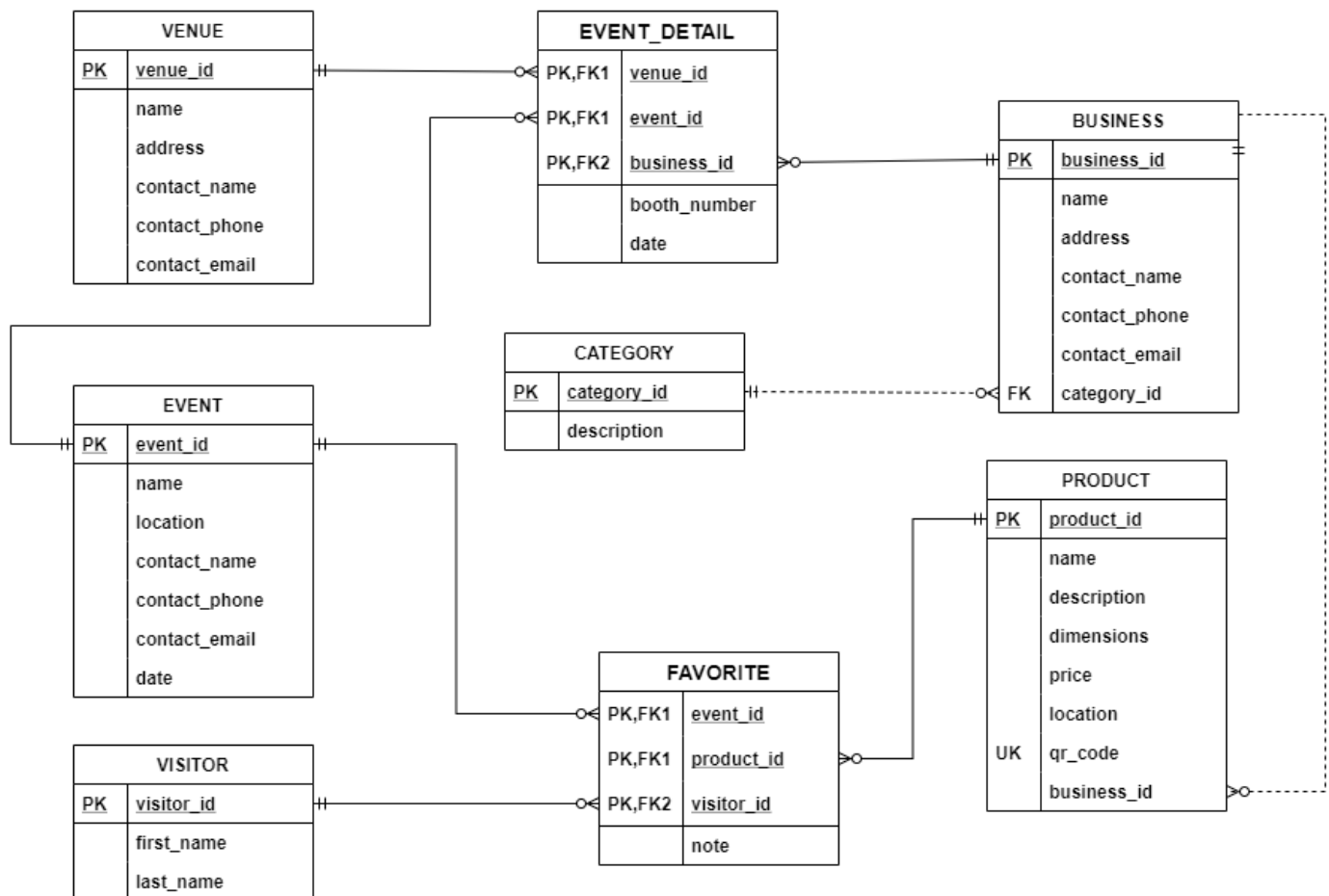


ENTITY RELATIONSHIP DIAGRAM

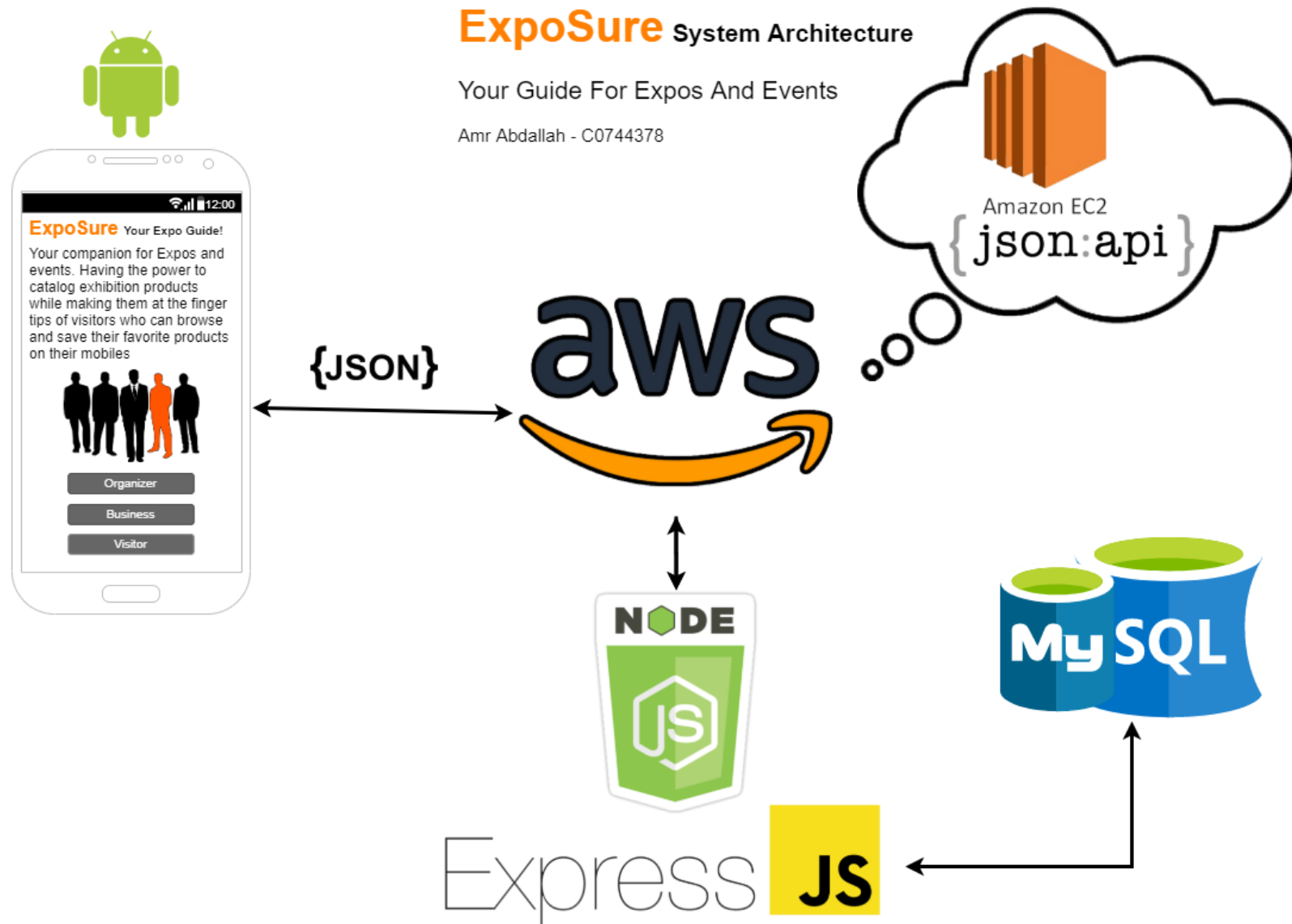
ExpoSure Entity Relationship Diagram

Your Guide For Expos And Events

Amr Abdallah - C0744378



SYSTEM ARCHITECTURE



BACK END CODE SNIPPETS

GENERAL VIEW

```

File Edit Selection View Go Run Terminal Help
Response(12ms) - ExpoSure_Node [SSH: Android_ec2-User_AWS] - Visual Studio Code

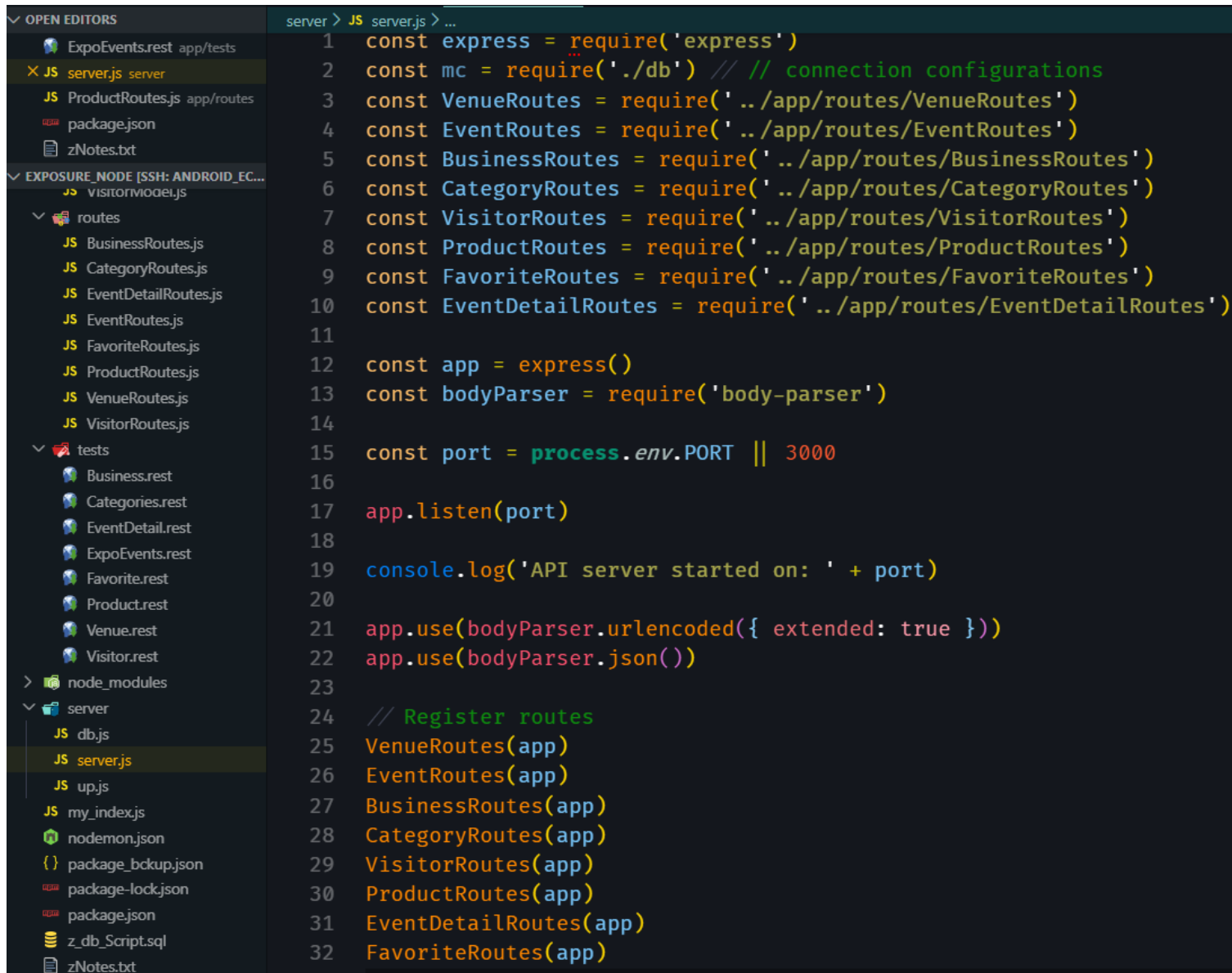
EXPLORER
OPEN EDITORS
  GROUP 1
    ExpoEvents.rest app/tests
    JS ProductRoutes.js app/routes
    package.json
    zNotes.txt
  GROUP 2
    Response(12ms)
  EXPOSURE_NO...
    JS EventDetailModel.js
    JS EventModel.js
    JS FavoriteModel.js
    JS ProductModel.js
    JS VenueModel.js
    JS VisitorModel.js
    routes
      JS BusinessRoutes.js
      JS CategoryRoutes.js
      JS EventDetailRoutes.js
      JS EventRoutes.js
      JS FavoriteRoutes.js
      JS ProductRoutes.js
      JS VenueRoutes.js
      JS VisitorRoutes.js
    tests
      Business.rest
      Categories.rest
      EventDetail.rest
      ExpoEvents.rest
      Favorite.rest
      Product.rest
      Venue.rest
      Visitor.rest
    node_modules
    server
      JS db.js
      JS server.js
      JS up.js
  OUTLINE
  TIMELINE
  NPM SCRIPTS

app > tests > ExpoEvents.rest
1 # https://marketplace.visualstudio.com/items?itemName=humao.
2 # Change url from vscode Settings.json
3 # [ctrl+.] → type json → click Settings.json → Very Botto
0 references
4 @event_Id = 5
6 references
5 @contentType = application/json
6
7 ### Get ALL Events
Send Request
8 GET {{url}}/Expo_Events HTTP/1.1
9 Accept: {{contentType}}
10
11 ### Get Event By ID
Send Request
12 GET {{url}}/Expo_Events/13 HTTP/1.1
13 Accept: {{contentType}}
14
15 ### Get Event By Name
Send Request
16 POST {{url}}/expo_events_by_name
17 Content-Type: {{contentType}}
18
19 {
20   "name": "data"
21 }
22
23 ### Create Event
Send Request
24 POST {{url}}/expo_events
25 Content-Type: {{contentType}}
26
27 {
28   "name": "Ibrahim",
29   "location": "Bla St.",
30   "contact_name": "Bla Contact",
31   "contact_phone": "516-455-5688",
32   "contact_email": "bla@bla.com",
33   "date": "2020/12/02 13:00",
34 }

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 7361
5 ETag: W/"1cc1-UGLECVjLgsh9169C6P7HKBs4Aks"
6 Date: Tue, 14 Apr 2020 03:22:06 GMT
7 Connection: close
8
9 √ [
10 √ {
11   "event_id": 1,
12   "name": "Data Connectors Toronto Cybersecurity Conference",
13   "location": "6 Prairieview Junction",
14   "contact_name": "Brantley March",
15   "contact_phone": "504(666)608-7746",
16   "contact_email": "bmarch0@google.cn",
17   "photo": "https://images.unsplash.com/photo-1523580494863-6f3031224c94?ixlib
18   =rb-1.2.1&ixid=eyJhcHBfawQjOjEyMDd9&auto=format&fit=crop&w=1050&q=80",
19   "date": "2019-10-18T14:00:00.000Z"
20 },
21 {
22   "event_id": 2,
23   "name": "Digital Talent Acquisition Summit (DTA)",
24   "location": "98 Bellgrove Hill",
25   "contact_name": "Constancy Newall",
26   "contact_phone": "62(904)847-1968",
27   "contact_email": "cnewall1@storify.com",
28   "photo": "https://images.unsplash.com/photo-1556761175-4b46a572b786?ixlib=rb
29   -1.2.1&ixid=eyJhcHBfawQjOjEyMDd9&auto=format&fit=crop&w=967&q=80",
30   "date": "2020-12-12T14:08:00.000Z"
31 },
32 {
33   "event_id": 3,
34   "name": "The World Leading AI Summit",
35   "location": "61820 Jackson Circle",
36   "contact_name": "Papagena Stacey",
37   "contact_phone": "7(519)212-4580",
38   "contact_email": "pstacey2@ucoz.ru",
39   "photo": "https://images.unsplash.com/photo-1560298803-1d998f6b5249?ixlib=rb

```


SETTING UP EXPRESS SERVER

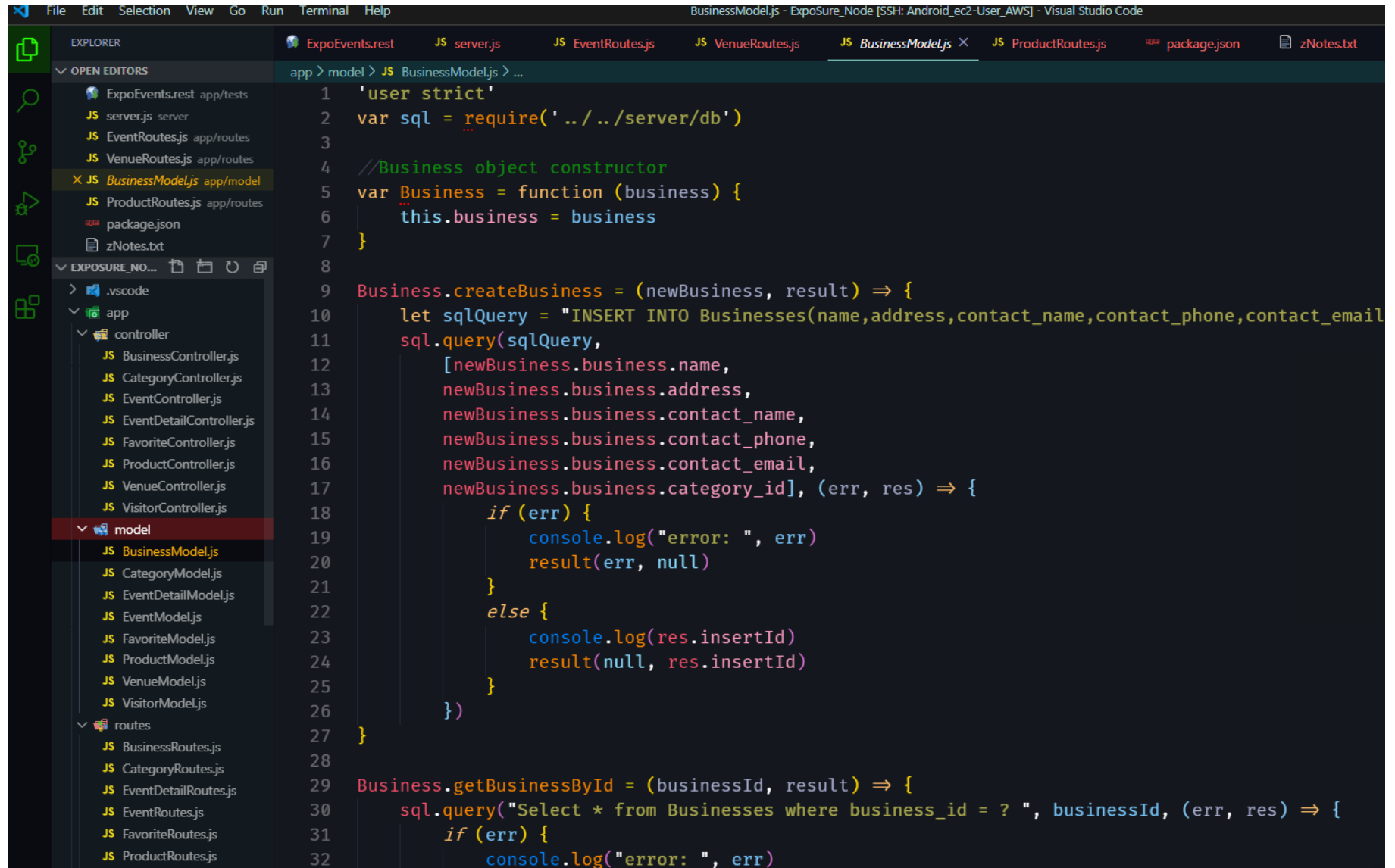


```
server > JS server.js > ...
1  const express = require('express')
2  const mc = require('./db') // // connection configurations
3  const VenueRoutes = require('../app/routes/VenueRoutes')
4  const EventRoutes = require('../app/routes/EventRoutes')
5  const BusinessRoutes = require('../app/routes/BusinessRoutes')
6  const CategoryRoutes = require('../app/routes/CategoryRoutes')
7  const VisitorRoutes = require('../app/routes/VisitorRoutes')
8  const ProductRoutes = require('../app/routes/ProductRoutes')
9  const FavoriteRoutes = require('../app/routes/FavoriteRoutes')
10 const EventDetailRoutes = require('../app/routes/EventDetailRoutes')
11
12 const app = express()
13 const bodyParser = require('body-parser')
14
15 const port = process.env.PORT || 3000
16
17 app.listen(port)
18
19 console.log('API server started on: ' + port)
20
21 app.use(bodyParser.urlencoded({ extended: true }))
22 app.use(bodyParser.json())
23
24 // Register routes
25 VenueRoutes(app)
26 EventRoutes(app)
27 BusinessRoutes(app)
28 CategoryRoutes(app)
29 VisitorRoutes(app)
30 ProductRoutes(app)
31 EventDetailRoutes(app)
32 FavoriteRoutes(app)
```

ROUTE CODE SNIPPET EXAMPLE

```
pp > routes > JS EventRoutes.js > <unknown> > module.exports
1  'use strict';
2  module.exports = (app) => {
3      var expoEvent = require(' ../controller/EventController');
4
5      // expoEvent Routes
6      app.route('/expo_events')
7          .get(expoEvent.list_all_events)
8          .post(expoEvent.create_an_event);
9
10     app.route('/expo_events/:eventId')
11         .get(expoEvent.read_an_event)
12         .put(expoEvent.update_an_event)
13         .delete(expoEvent.delete_an_event);
14     any
15     app.route('/expo_events_by_name')
16         .post(expoEvent.get_expo_events_by_name);
17 };
18
```

MODEL CODE SNIPPET

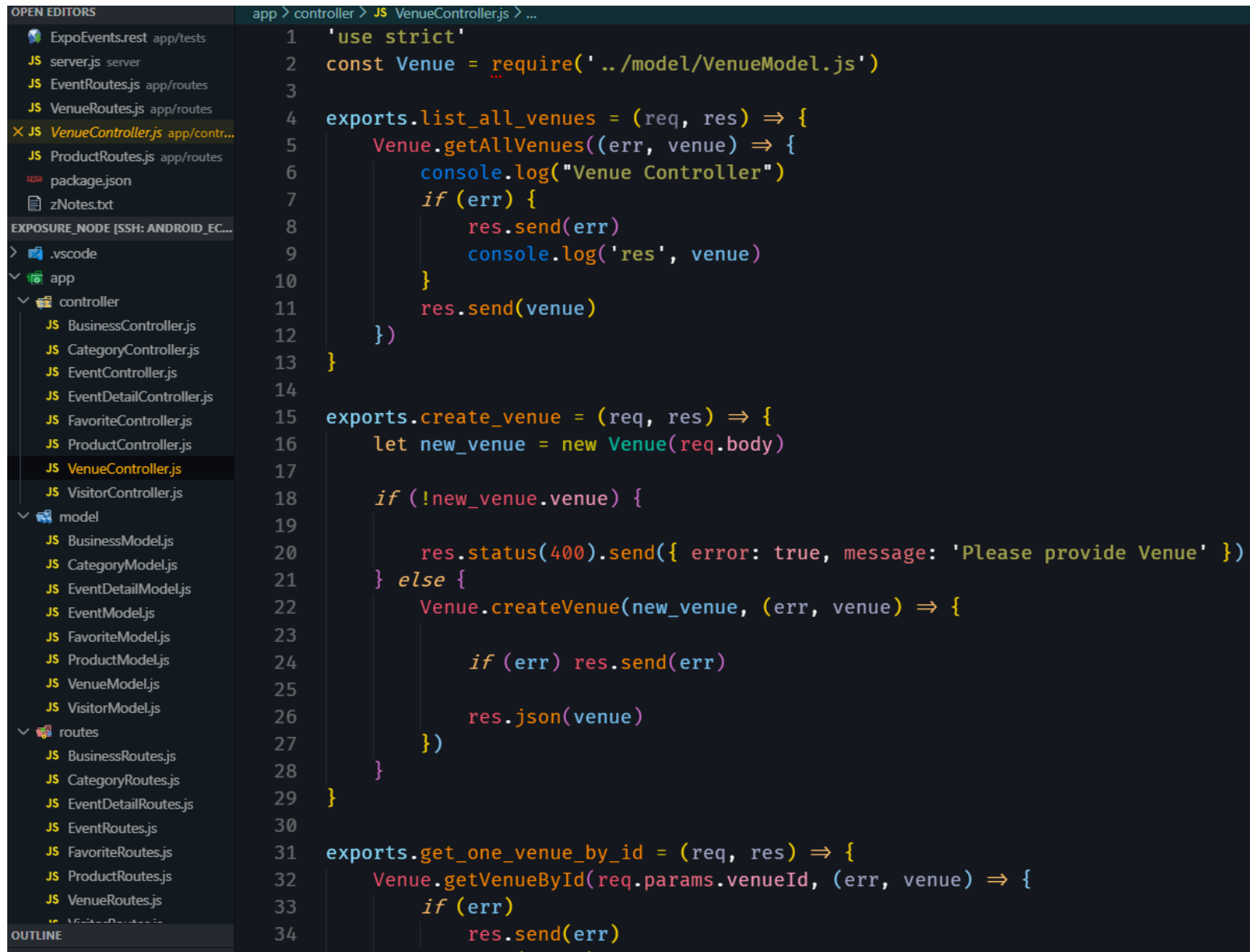


```
File Edit Selection View Go Run Terminal Help
BusinessModel.js - ExpoSure_Node [SSH: Android_ec2-User_AWS] - Visual Studio Code

EXPLORER
OPEN EDITORS
ExpoEvents.rest app/tests
JS server.js server
JS EventRoutes.js app/routes
JS VenueRoutes.js app/routes
X JS BusinessModel.js app/model
JS ProductRoutes.js app/routes
package.json
zNotes.txt
EXPOSURE_NO...
.vscode
app
controller
JS BusinessController.js
JS CategoryController.js
JS EventController.js
JS EventDetailController.js
JS FavoriteController.js
JS ProductController.js
JS VenueController.js
JS VisitorController.js
model
JS BusinessModel.js
JS CategoryModel.js
JS EventDetailModel.js
JS EventModel.js
JS FavoriteModel.js
JS ProductModel.js
JS VenueModel.js
JS VisitorModel.js
routes
JS BusinessRoutes.js
JS CategoryRoutes.js
JS EventDetailRoutes.js
JS EventRoutes.js
JS FavoriteRoutes.js
JS ProductRoutes.js

app > model > JS BusinessModel.js > ...
1 'user strict'
2 var sql = require(' ../../server/db')
3
4 //Business object constructor
5 var Business = function (business) {
6   this.business = business
7 }
8
9 Business.createBusiness = (newBusiness, result) => {
10   let sqlQuery = "INSERT INTO Businesses(name,address,contact_name,contact_phone,contact_email
11   sql.query(sqlQuery,
12     [newBusiness.business.name,
13     newBusiness.business.address,
14     newBusiness.business.contact_name,
15     newBusiness.business.contact_phone,
16     newBusiness.business.contact_email,
17     newBusiness.business.category_id], (err, res) => {
18     if (err) {
19       console.log("error: ", err)
20       result(err, null)
21     }
22     else {
23       console.log(res.insertId)
24       result(null, res.insertId)
25     }
26   })
27 }
28
29 Business.getBusinessById = (businessId, result) => {
30   sql.query("Select * from Businesses where business_id = ? ", businessId, (err, res) => {
31     if (err) {
32       console.log("error: ", err)
```

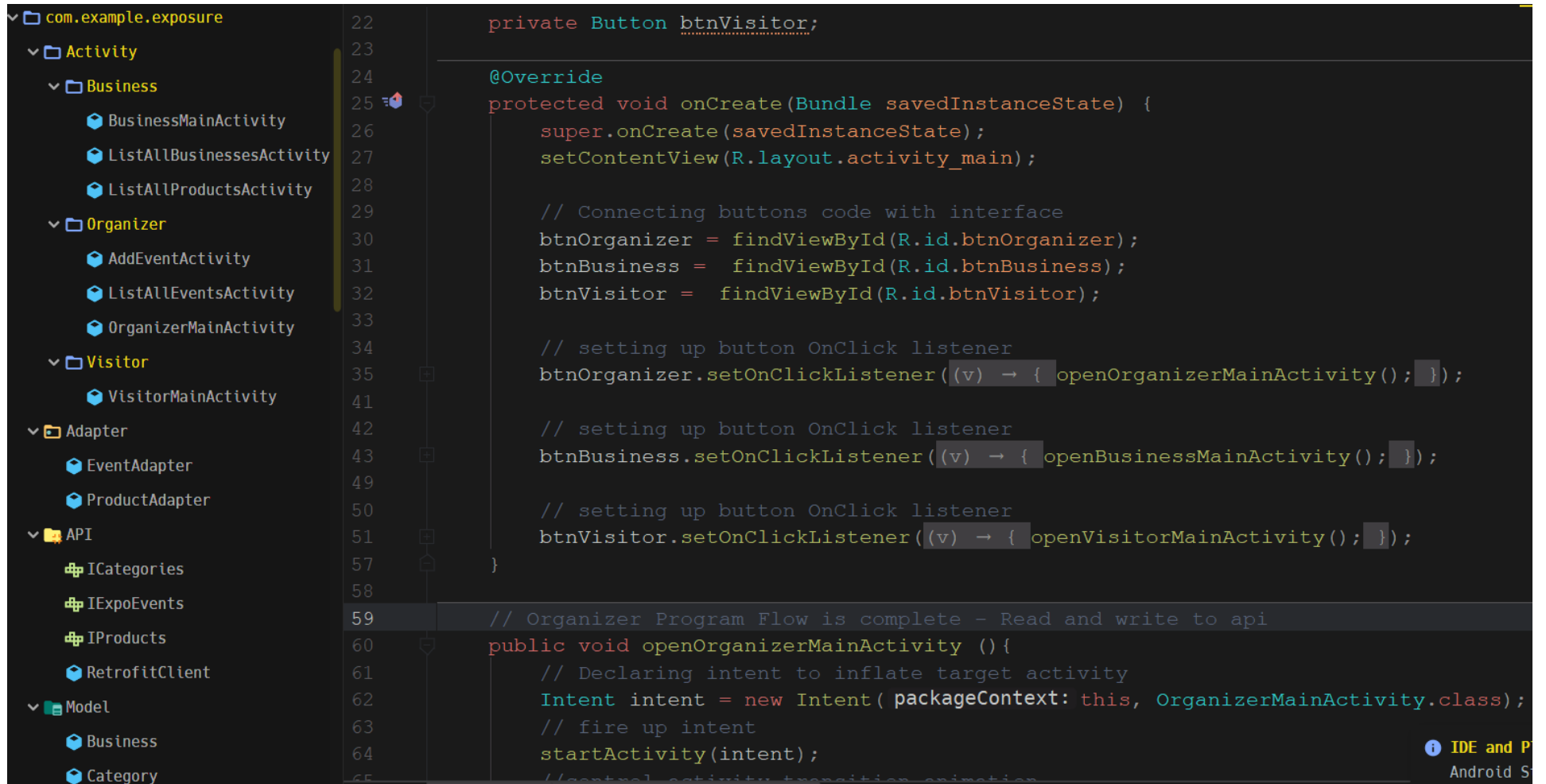
CONTROLLER SNIPPET



```
1  'use strict'
2  const Venue = require('../model/VenueModel.js')
3
4  exports.list_all_venues = (req, res) => {
5    Venue.getAllVenues((err, venue) => {
6      console.log("Venue Controller")
7      if (err) {
8        res.send(err)
9        console.log('res', venue)
10     }
11     res.send(venue)
12   })
13 }
14
15 exports.create_venue = (req, res) => {
16   let new_venue = new Venue(req.body)
17
18   if (!new_venue.venue) {
19     res.status(400).send({ error: true, message: 'Please provide Venue' })
20   } else {
21     Venue.createVenue(new_venue, (err, venue) => {
22       if (err) res.send(err)
23       res.json(venue)
24     })
25   }
26 }
27
28
29
30
31 exports.get_one_venue_by_id = (req, res) => {
32   Venue.getVenueById(req.params.venueId, (err, venue) => {
33     if (err)
34       res.send(err)
```

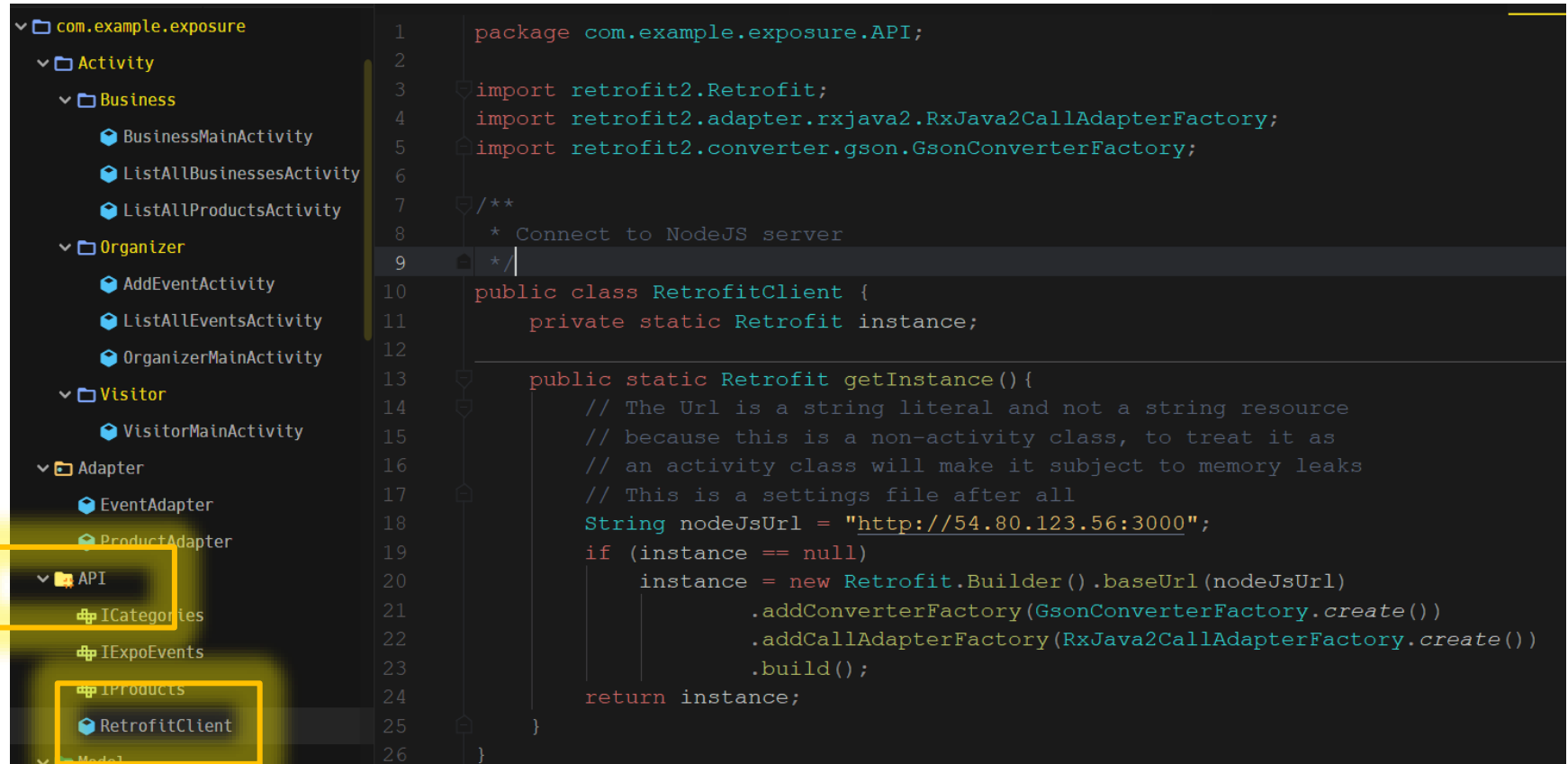
VIEW SNIPPETS (ANDROID APPLICATION)

GENERAL VIEW

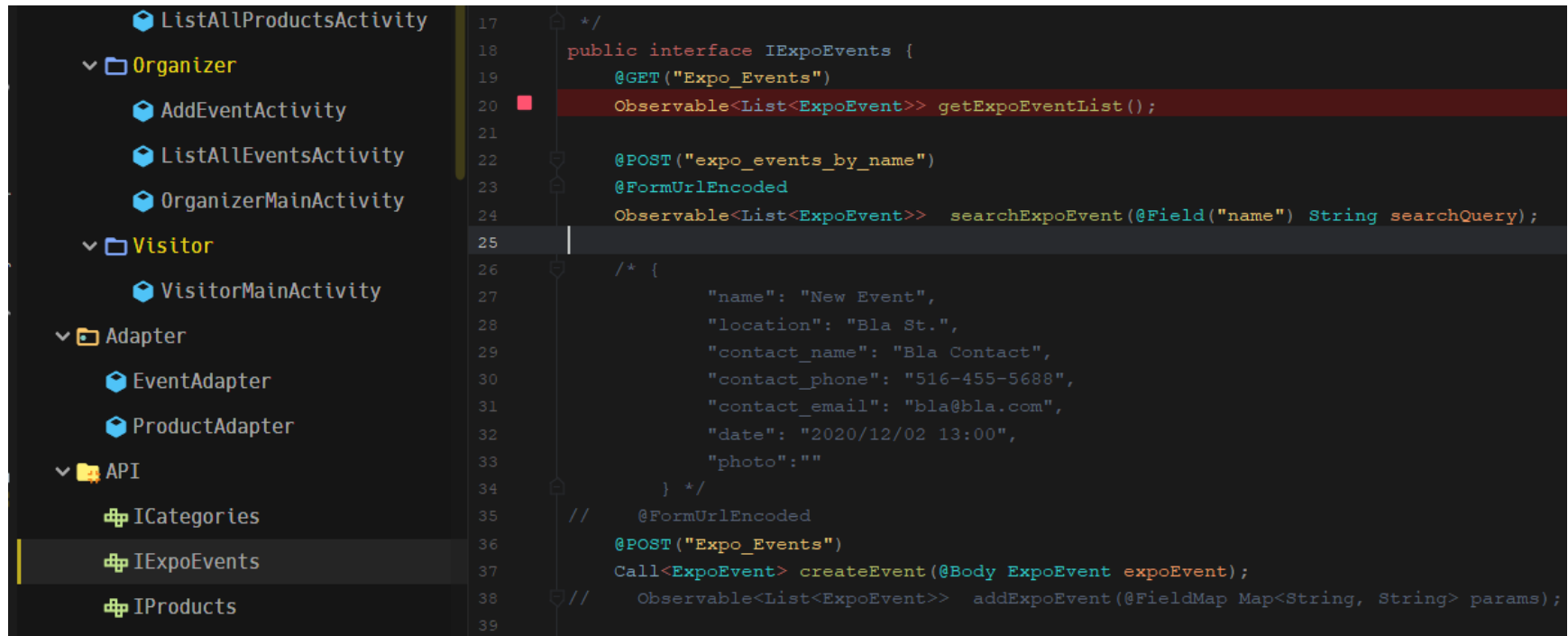


```
22 private Button btnVisitor;
23
24 @Override
25 protected void onCreate(Bundle savedInstanceState) {
26     super.onCreate(savedInstanceState);
27     setContentView(R.layout.activity_main);
28
29     // Connecting buttons code with interface
30     btnOrganizer = findViewById(R.id.btnOrganizer);
31     btnBusiness = findViewById(R.id.btnBusiness);
32     btnVisitor = findViewById(R.id.btnVisitor);
33
34     // setting up button onClick listener
35     btnOrganizer.setOnClickListener((v) -> { openOrganizerMainActivity(); });
36
37     // setting up button onClick listener
38     btnBusiness.setOnClickListener((v) -> { openBusinessMainActivity(); });
39
40     // setting up button onClick listener
41     btnVisitor.setOnClickListener((v) -> { openVisitorMainActivity(); });
42
43     // Organizer Program Flow is complete - Read and write to api
44     public void openOrganizerMainActivity () {
45         // Declaring intent to inflate target activity
46         Intent intent = new Intent( packageContext: this, OrganizerMainActivity.class);
47         // fire up intent
48         startActivity(intent);
49         //control activities transition animation
50     }
51 }
```

REST CLIENT API



API INTERFACE SNIPPET



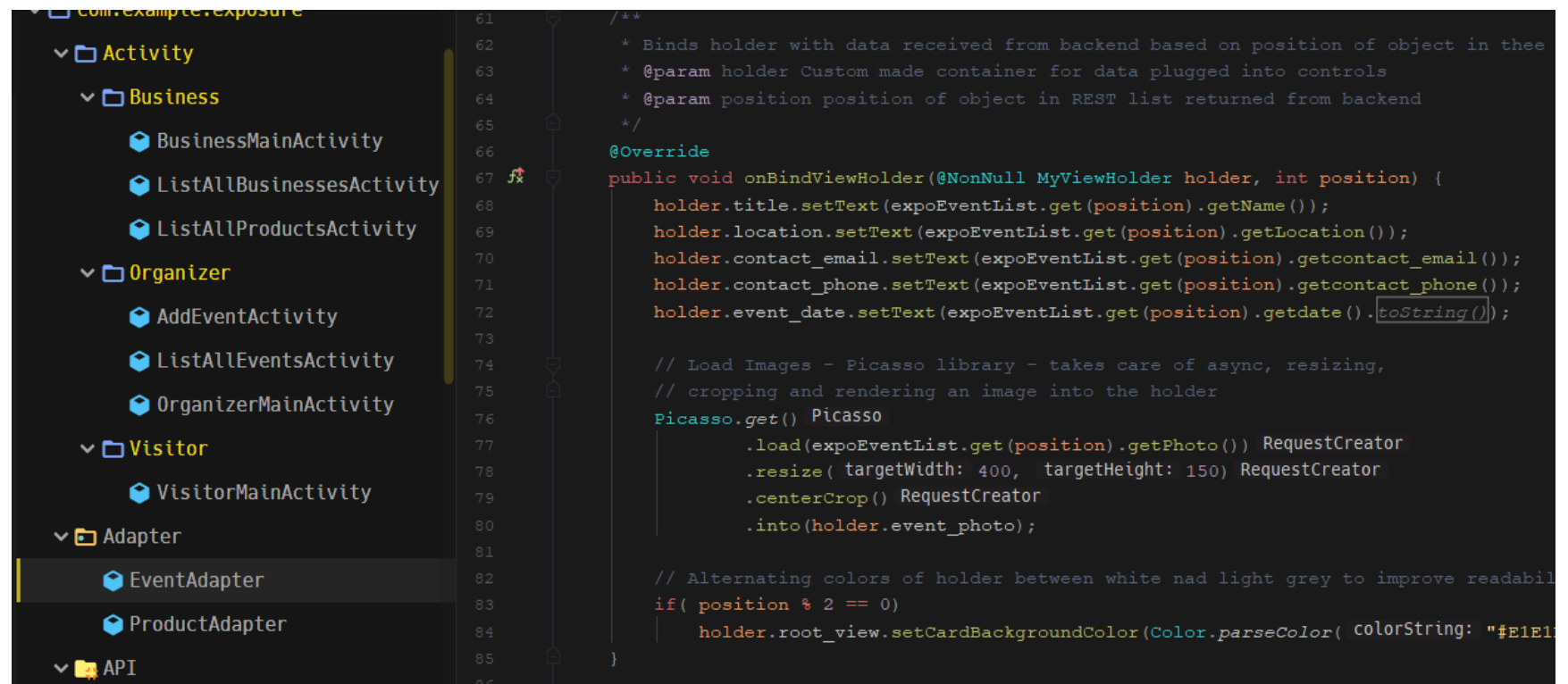
The screenshot displays an IDE interface. On the left, a project structure tree is visible with the following hierarchy:

- ListAllProductsActivity
- Organizer
 - AddEventActivity
 - ListAllEventsActivity
 - OrganizerMainActivity
- Visitor
 - VisitorMainActivity
- Adapter
 - EventAdapter
 - ProductAdapter
- API
 - ICategories
 - IExpoEvents (highlighted)
 - IProducts

On the right, a code editor shows the implementation of the `IExpoEvents` interface. The code is as follows:

```
17  */
18  public interface IExpoEvents {
19      @GET("Expo_Events")
20      Observable<List<ExpoEvent>> getExpoEventList();
21
22      @POST("expo_events_by_name")
23      @FormUrlEncoded
24      Observable<List<ExpoEvent>> searchExpoEvent(@Field("name") String searchQuery);
25
26      /* {
27          "name": "New Event",
28          "location": "Bla St.",
29          "contact_name": "Bla Contact",
30          "contact_phone": "516-455-5688",
31          "contact_email": "bla@bla.com",
32          "date": "2020/12/02 13:00",
33          "photo":""
34      } */
35      // @FormUrlEncoded
36      @POST("Expo_Events")
37      Call<ExpoEvent> createEvent(@Body ExpoEvent expoEvent);
38      // Observable<List<ExpoEvent>> addExpoEvent(@FieldMap Map<String, String> params);
39  }
```

ADAPTER SNIPPET



ANDROID MODEL SNIPPET

```
@SerializedName("contact_email")
@Expose
private String contact_email;

@SerializedName("photo")
@Expose
private String photo;

@SerializedName("date")
@Expose
private String date;

/**
 * Events Constructor with id
 * @param event_id Event Id
 * @param name Event Name
 * @param location Event Location
 * @param contact_name Event contact name
 * @param contact_phone Event contact phone
 * @param contact_email Event contact email
 * @param date Event date
 * @param photo Event photo
 */
public ExpoEvent(String event_id, String name, String location, String contact_name,
                 String contact_phone, String contact_email, String date, String photo) {
    this.event_id = event_id;
    this.name = name;
    this.location = location;
    this.contact_name = contact_name;
    this.contact_phone = contact_phone;
    this.contact_email = contact_email;
    this.date = date;
    this.photo = photo;
}

ExpoEvent > getevent_id()
```

ACTIVITY SNIPPET

```
150 // plugs into adapter and searches for event name
151 private void startSearch(String query) {
152     materialSearchBar.clearFocus();
153     compositeDisposable.add(myAPI.searchExpoEvent(query)
154         .subscribeOn(Schedulers.io())
155         .observeOn(AndroidSchedulers.mainThread())
156         .subscribe((Consumer) (expoEvents) → {
157             adapter = new EventAdapter(expoEvents);
158             recycler_event_search.setAdapter(adapter);
159         }, (Consumer) (throwable) → {
160             String msg = "Not Found";
161             Toast.makeText(context: ListAllEventsActivity.this, msg, Toast.LENGTH_SHORT).show();
162         }));
163 }
164
165 // plugs into adapter and gets ExpoEvents
166 private void getAllEvents(){
167     compositeDisposable.add(myAPI.getExpoEventList()
168         .subscribeOn(Schedulers.io())
169         .observeOn(AndroidSchedulers.mainThread())
170         .subscribe((Consumer) (expoEvents) → {
171             adapter = new EventAdapter(expoEvents);
172             recycler_event_search.setAdapter(adapter);
173         }, (Consumer) (throwable) → {
174             String msg = "Not Found";
175             Toast.makeText(context: ListAllEventsActivity.this, msg, Toast.LENGTH_SHORT).show();
176         }));
177 }
178
179 }
```

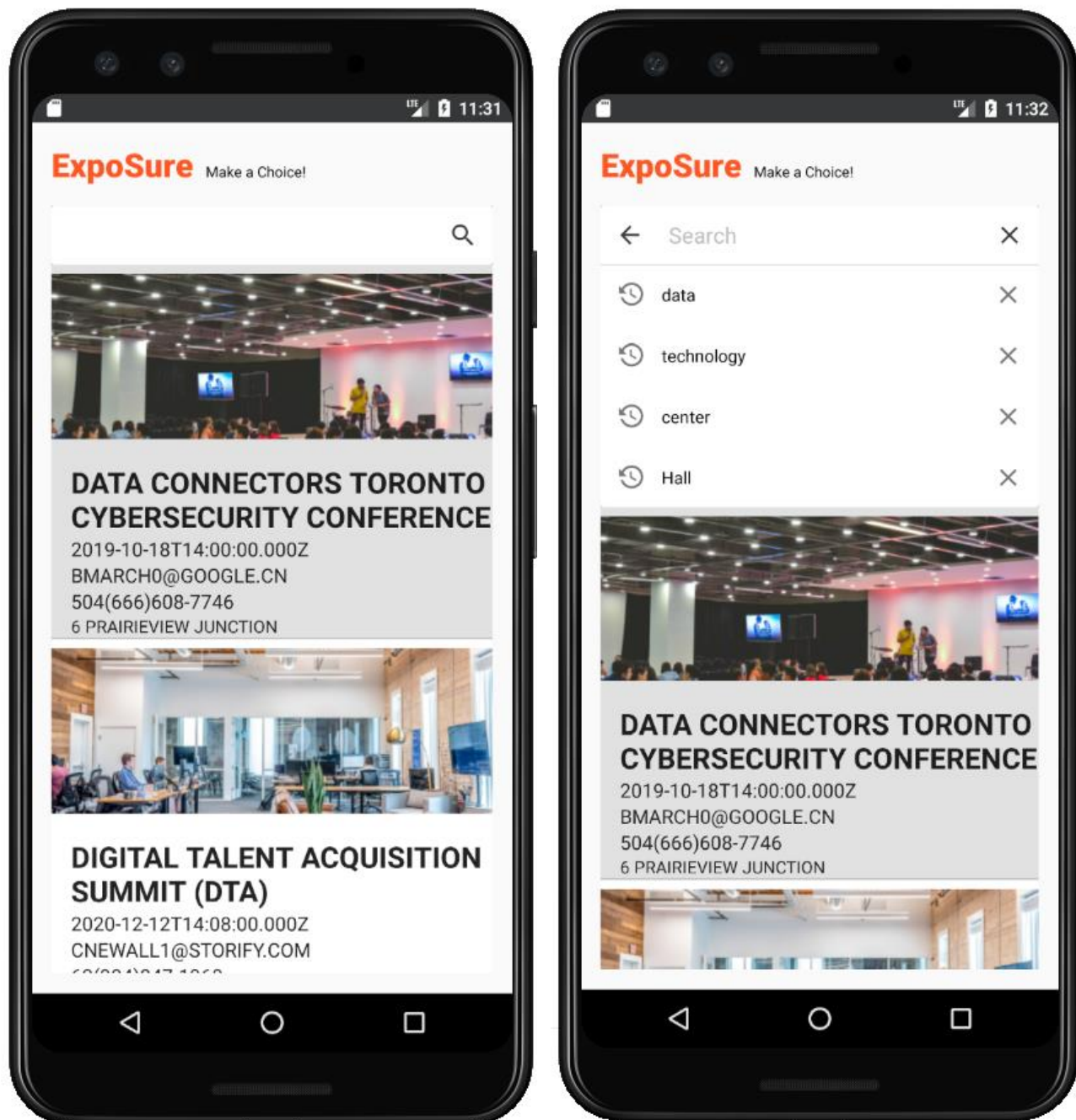
DATABASE SCRIPT SAMPLE

```
136 -- -----
137 -- Table `ExpoSure`.`Favorites`
138 -- -----
139 DROP TABLE IF EXISTS `ExpoSure`.`Favorites` ;
140
141 CREATE TABLE IF NOT EXISTS `ExpoSure`.`Favorites` (
142     `Products_product_id` INT NOT NULL,
143     `Visitors_visitor_id` INT NOT NULL,
144     `Expo_Events_event_id` INT NOT NULL,
145     `note` VARCHAR(200) NULL DEFAULT NULL,
146     PRIMARY KEY (`Products_product_id`, `Visitors_visitor_id`, `Expo_Events_event_id`),
147     INDEX `fk_Favorites_Products1_idx` (`Products_product_id` ASC) VISIBLE,
148     INDEX `fk_Favorites_Visitors1_idx` (`Visitors_visitor_id` ASC) VISIBLE,
149     INDEX `fk_Favorites_Expo_Events1_idx` (`Expo_Events_event_id` ASC) VISIBLE,
150     CONSTRAINT `fk_Favorites_Products1`
151         FOREIGN KEY (`Products_product_id`)
152         REFERENCES `ExpoSure`.`Products` (`product_id`)
153         ON DELETE CASCADE
154         ON UPDATE CASCADE,
155     CONSTRAINT `fk_Favorites_Visitors1`
156         FOREIGN KEY (`Visitors_visitor_id`)
157         REFERENCES `ExpoSure`.`Visitors` (`visitor_id`)
158         ON DELETE CASCADE
159         ON UPDATE CASCADE,
160     CONSTRAINT `fk_Favorites_Expo_Events1`
161         FOREIGN KEY (`Expo_Events_event_id`)
162         REFERENCES `ExpoSure`.`Expo_Events` (`event_id`)
163         ON DELETE CASCADE
164         ON UPDATE CASCADE)
165 ENGINE = InnoDB
166 DEFAULT CHARACTER SET = utf8mb4
167 COLLATE = utf8mb4_0900_ai_ci;
168
```

APPLICATION SCREENSHOTS

LIST ALL EVENTS

All the events saved to the database will be retrieved on this screen. The user can use the search bar to filter events by name.



CREATING NEW EVENTS

The image displays two side-by-side smartphone screens showing the 'ExpoSure Create Event' form. The left screen shows the form with empty input fields, while the right screen shows the form filled with example data.

ExpoSure Create Event

Enter event details below.

Name
Name

Location
Location

Contact Name
Contact Name

Contact Phone
Contact Phone

Email
Email

Date
Date **GET DATE**

Time
Time **GET TIME**

CREATE EVENT

ExpoSure Create Event

Enter event details below.

Name
Totoronto Technology Summit

Location
The Great Dome

Contact Name
Mike Finch

Contact Phone
555-555-5555

Email
mike@finch.com

Date
Thursday, 30 Apr 2020 **GET DATE**

Time
08:45 AM **GET TIME**

CREATE EVENT

Typo is intentional to show search capability.

