

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа № 2

“Тестирование, разработка и документирование RESTful API”

Выполнил:

Ле Хоанг Чыонг

Чан Дык Минь

Группа:

K33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2024 г.

Задача

Нужно реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Вариант №3: Сервис-помощник в планировании путешествий.

Требуемый функционал:

- регистрация, авторизация,
- создание профиля,
- создание поездки с заполнением информации о ней,
- поиск по существующим предложениям для поездок,
- предложения для новой созданной поездки на основе интересов пользователя.

Ход работы

1. Проект будет иметь следующую структуру:

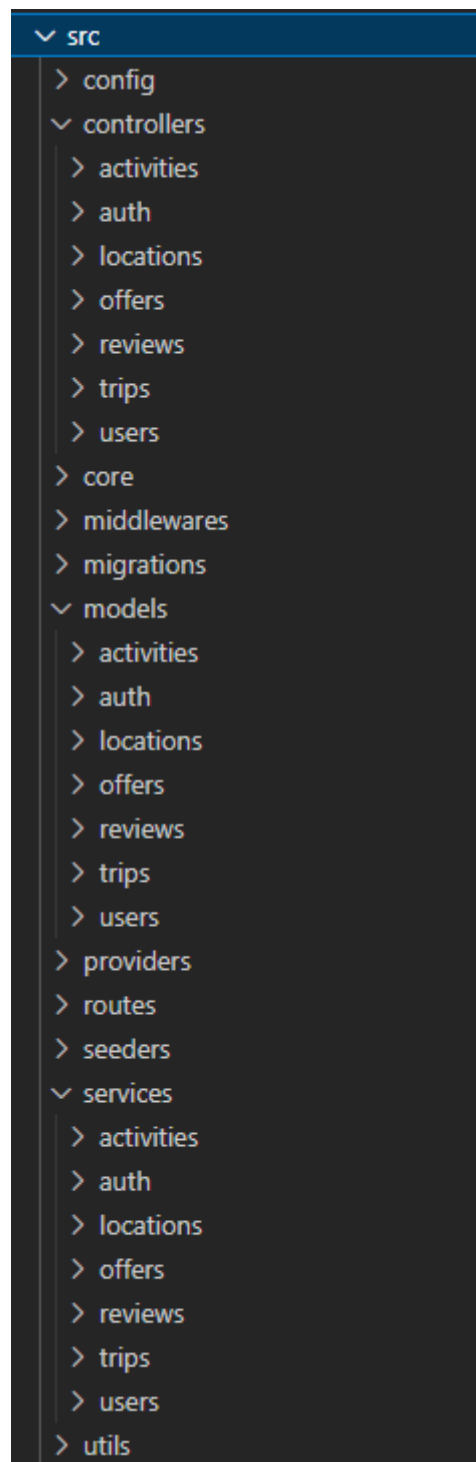


Рисунок 1 — Файловая структура проекта

2. Вход и регистрация были реализованы еще в первой лабораторной работе.

3. Создание профиля.

Часть кода из файла `src>models>users>user.ts`:

```
@Table
export class User extends Model<UserAttributes,UserCreationAttributes> {
  @PrimaryKey
  @Column({
    type: DataType.UUID,
    defaultValue: DataType.UUIDV4,
  })
  id : string;

  @Column
  name: string;

  @Unique
  @Column
  email: string

  @AllowNull(false)
  @Column
  password: string

  @BeforeCreate
  @BeforeUpdate
  static generatePasswordHash(instance: User) {
    const { password } = instance

    if (instance.changed('password')) {
      instance.password = hashPassword(password)
    }
  }

  @BelongsToMany(() => Activity, () => UserActivity)
  activities: Activity[];

  @BelongsToMany(() => Location, () => Review)
  locations: Location[];
}

export default User
```

Рисунок 2 — Модель User

Часть кода из файла `src>controllers>auth>auth.ts`:

```
register = async (request: any, response: any) => {  
  const { body } = request  
  try {  
    const data: any = await this.authService.register(body)  
    response.status(201).send(data)  
  } catch (error: any) {  
    response.status(400).send(getErrorMessage(error))  
  }  
}
```

Рисунок 3 — Auth Controller

```
createUserActivity = async (request: any, response: any) => {  
  try {  
    const { body } = request;  
    const userId = request.user.id;  
    const data: any = await this.userActivityService.create({ ...body, user_id: userId })  
    response.status(201).send(data)  
  } catch (error: any) {  
    response.status(404).send(getErrorMessage(error))  
  }  
}
```

Рисунок 4 — UserActivity Controller

4. Создание поездки с заполнением информации о ней

Часть кода из файла `src>models>trips>trip.ts`:

```

@Table
export class Trip extends Model<TripAttributes, TripCreationAttributes> {
  @PrimaryKey
  @Column({
    type: DataType.UUID,
    defaultValue: DataType.UUIDV4,
  })
  id: string;

  @Column
  name: string;

  @Column
  description: string;

  @Column({ type: DataType.DATE })
  dateStart: Date;

  @Column({ type: DataType.DATE })
  dateEnd: Date;

  @Column
  budget: number;

  @Column({
    type: DataType.ENUM(...Object.values(TripType)),
  })
  type: TripType;

  @ForeignKey(() => User)
  @Column({
    type: DataType.UUID,
  })
  userId: string;

  @BelongsTo(() => User)
  user: User;

  @BelongsToMany(() => Location, () => TripLocation)
  locations: Location[];
}

```

Рисунок 5 — Модель Trip

Часть кода из файла `src>controllers>trips>trip.ts`:

```

create = async (request: any, response: any) => {
  try {
    const { body } = request;
    const userId = request.user.id;
    const trip: any = await this.tripService.create({ ...body, userId: userId });
    response.status(201).send(trip)
  } catch (error: any) {
    response.status(404).send(getErrorMessage(error))
  }
}

```

Рисунок 6 — Controller Trip

Часть кода из файла src>services>trips>trip.ts:

```

async create(tripData: any): Promise<Trip> {
  try {
    const trip = await Trip.create(tripData);
    return trip;
  } catch (error) {
    throw error;
  }
}

```

Рисунок 7 — Trip service

5. Поиск по существующим предложениям для поездок

Каждому пользователю при регистрации необходимо будет сохранить свои любимые занятия, затем на основе этих интересов (например: поход на пляж, восхождение в горы, еды и т. д.) программа найдет место, где есть эти занятия, оттуда вы найдете из предложений этого места

Часть кода из файла `src>services>offers>offers.ts`:

```
async getOfferForUser(user_id: string, type?: string, price?: number): Promise<Offer[]> {
  try {
    const activities = await UserActivity.findAll({ where: { user_id: user_id } });

    if (!activities || activities.length === 0) {
      return [];
    }
    const activityIds = activities.map(activity => activity.activity_id);

    const locationSet: Set<string> = new Set();

    const locationPromises = activityIds.map(async activityId => {
      const locationsWithActivity = await LocationActivity.findAll({ where: { activity_id: activityId } });
      locationsWithActivity.forEach(location => {
        locationSet.add(location.location_id);
      });
    });
    await Promise.all(locationPromises);

    const locationIds: string[] = [...locationSet];

    const offerPromises: Promise<Offer[]>[] = [];

    locationIds.forEach(locationId => {
      let whereCondition: any = { location_id: locationId };
      if (type) {
        whereCondition.type = type;
      }

      if (price) {
        whereCondition.price = { [Op.lte]: price };
      }

      const offerPromise = Offer.findAll({ where: whereCondition });
      offerPromises.push(offerPromise);
    });

    const allOffers = await Promise.all(offerPromises);

    const flattenedOffers = allOffers.flat();
    return flattenedOffers;
  } catch (error) {
    throw error;
  }
}
```

Рисунок 8 — Offer service

Часть кода из файла `src>controllers>offers>offers.ts`:

```
getOfferForUser = async (request: any, response: any) => {
  try {
    const user_id = request.user.id;
    const { type, max_price } = request.body;

    const offers: any = await this.offerService.getOfferForUser(user_id, type, max_price);

    response.status(200).send(offers);
  } catch (error: any) {
    response.status(404).send(getErrorMessage(error));
  }
}
```

Рисунок 8 — Offer controller

6. Предложения для новой созданной поездки на основе интересов пользователя.

Так же, как и при поиске offer для пользователей, программа на основе любимых занятий пользователя найдет любимые места, а затем вернет данные пользователю.

Часть кода из файла `src>controllers>users>user.ts`:

```
async getLocationForUser(user_id: string, rating?: number): Promise<Location[]> {
  try {
    const activities = await UserActivity.findAll({ where: { user_id: user_id } });

    if (!activities || activities.length === 0) {
      return [];
    }

    const activityIds = activities.map(activity => activity.activity_id);

    const locationSet: Set<string> = new Set();

    const locationPromises = activityIds.map(async activityId => {
      const locationsWithActivity = await LocationActivity.findAll({ where: { activity_id: activityId } });
      locationsWithActivity.forEach(location => {
        locationSet.add(location.location_id);
      });
    });

    await Promise.all(locationPromises);

    const locationIds: string[] = [...locationSet];
    let locations: Location[];

    if (rating) {
      locations = await Location.findAll({
        where: {
          id: locationIds,
          rating: { [Op.gte]: rating }
        },
        order: [['rating', 'DESC']],
      });
    } else {
      locations = await Location.findAll({
        where: { id: locationIds },
        order: [['rating', 'DESC']],
      });
    }

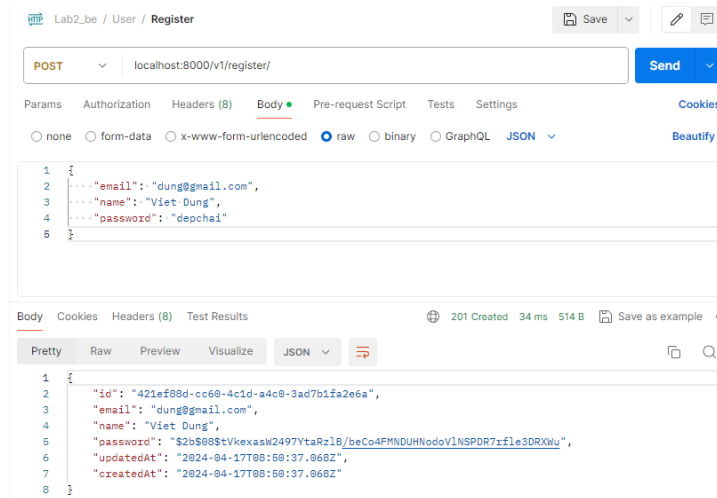
    return locations;
  } catch (error) {
    throw error;
  }
}
```

Часть кода из файла `src>services>users>user.ts`:

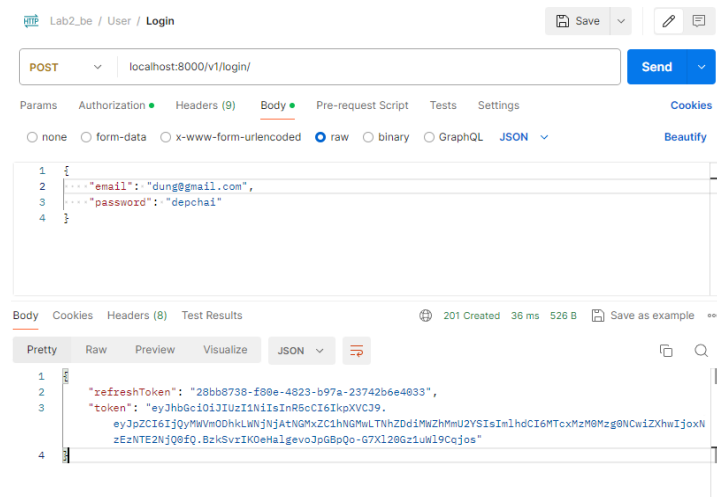
```
getLocationForUser = async (request: any, response: any) => {  
  try {  
    const user_id = request.user.id;  
    const { min_rating } = request.body;  
    const locations: Location[] = await this.userService.getLocationForUser(user_id, min_rating);  
    response.status(201).send(locations)  
  } catch (error: any) {  
    response.status(404).send(getErrorMessage(error))  
  }  
}
```

Postman

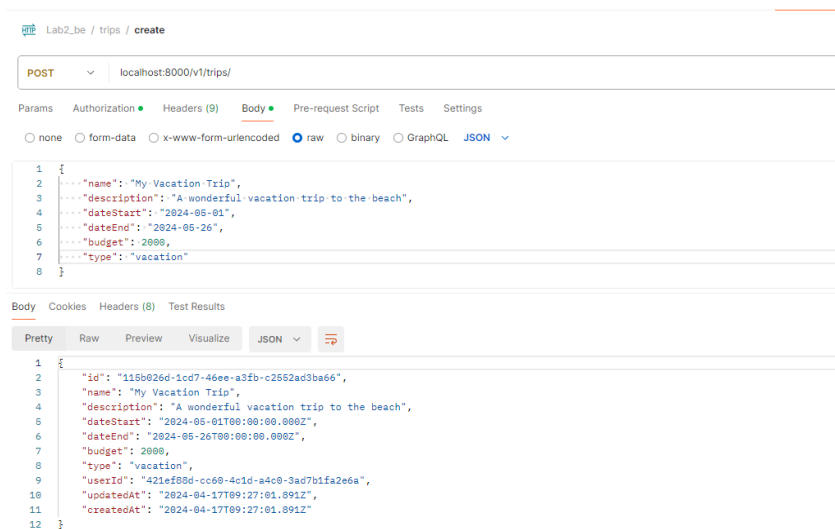
Зарегистрировать нового пользователя



Авторизоваться



Создать новую поездку



Обновить поездку

HTTP

Lab2_be / trips / update

PUT

localhost:8000/v1/trips/fb358484-676c-4001-a358-1b5a58d17465

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

{

2

"name": "du lich ha noi",

3

"description": "di choi 10 ngay",

4

"dateStart": "2024-03-20",

5

"dateEnd": "2024-03-30",

6

"budget": 4000,

7

"type": "vacation"

8

}

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"id": "fb358484-676c-4001-a358-1b5a58d17465",

3

"name": "du lich ha noi",

4

"description": "di choi 10 ngay",

5

"dateStart": "2024-03-20T00:00:00.000Z",

6

"dateEnd": "2024-03-30T00:00:00.000Z",

7

"budget": 4000,

8

"type": "vacation",

9

"userId": "c645f697-4791-4c73-a202-3b60952f98b9",

10

"createdAt": "2024-04-09T18:16:16.789Z",

11

"updatedAt": "2024-04-17T09:46:28.591Z"

12

}

Удалить поездку

HTTP

Lab2_be / trips / delete trip

DELETE

localhost:8000/v1/trips/71eb3f8a-9fba-42fa-8d2f-b97bd8796178

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

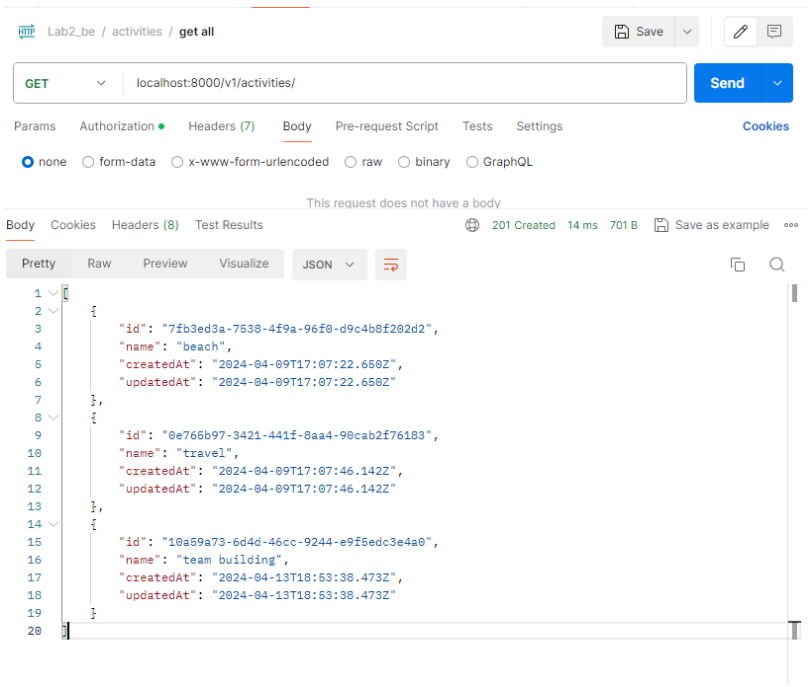
Visualize

HTML

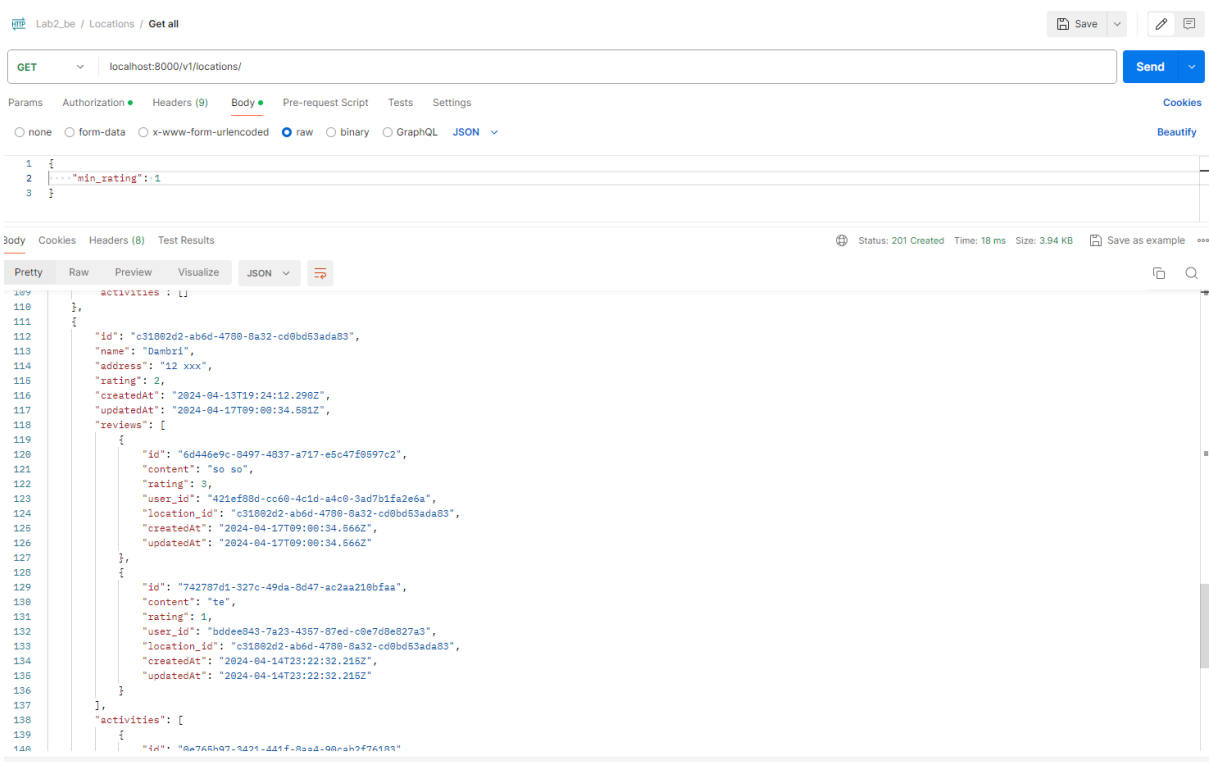
1

Trip has been successfully deleted

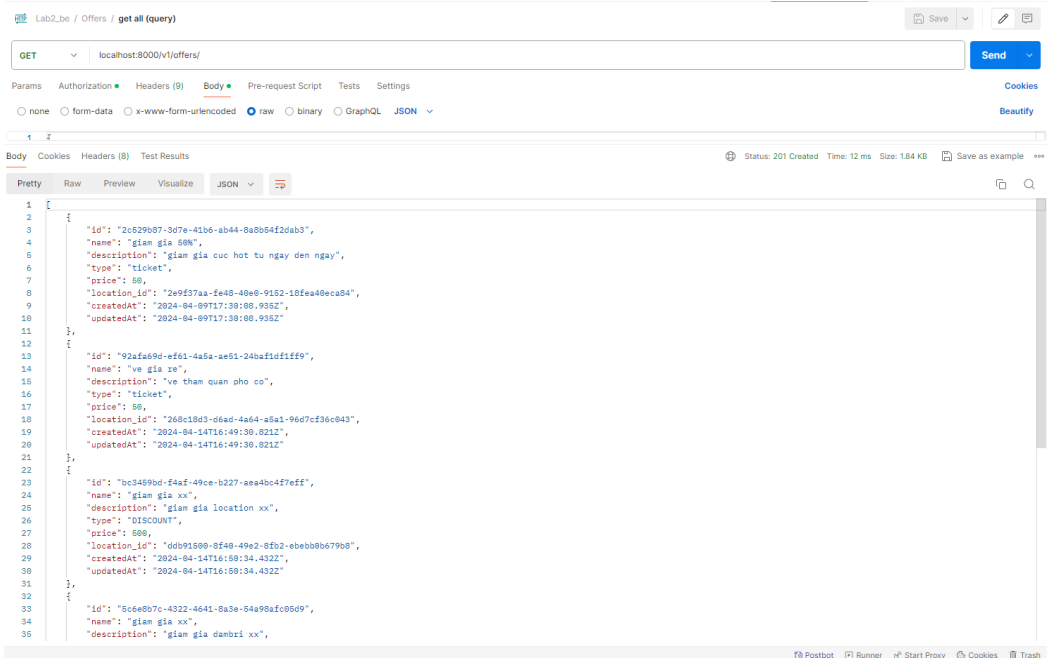
Получить activities (варианты по интересам пользователя)



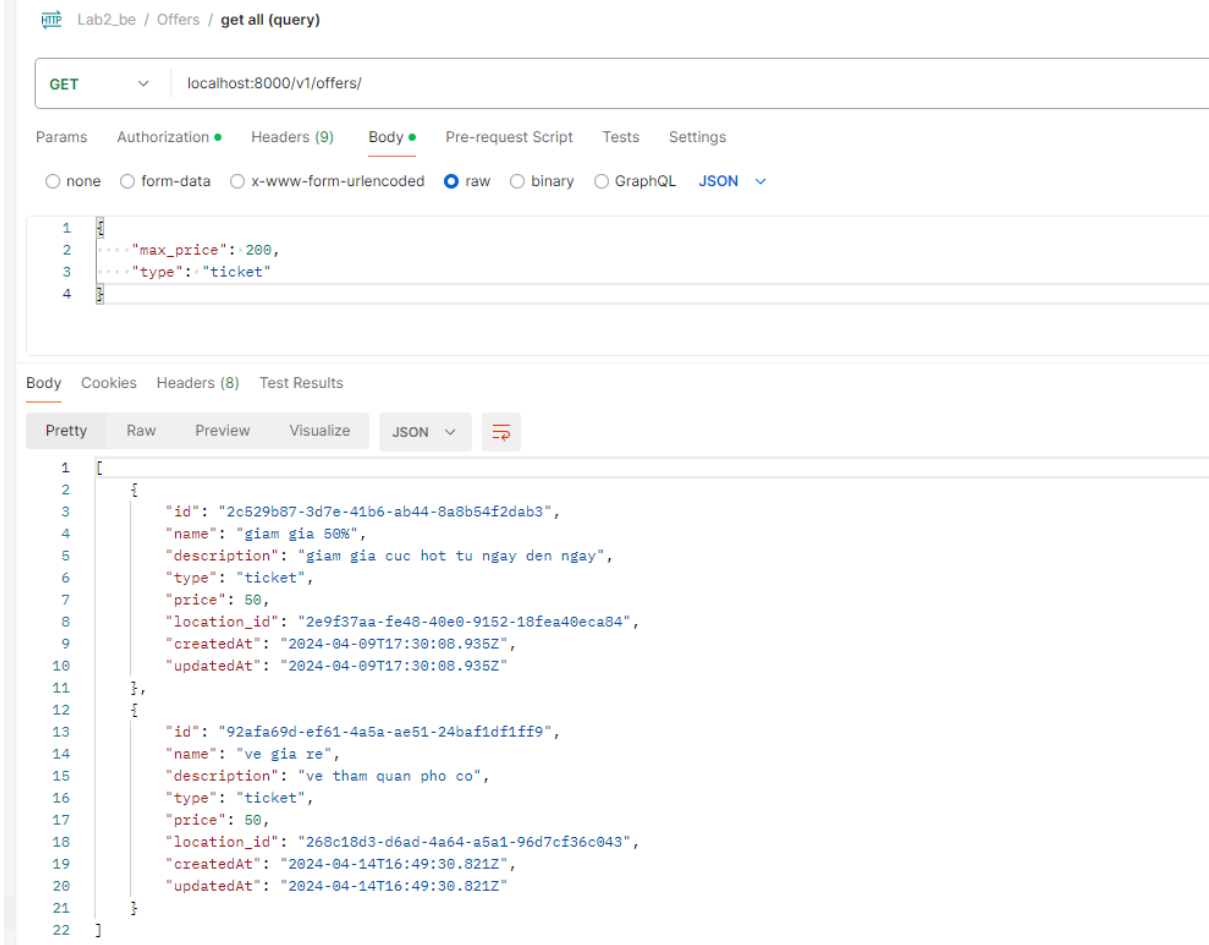
Получить locations, locations ответа будет вложено в reviews и activities.



Получить все предложения



Получить предложение по запросам (цена ниже 200, тип «билет»)



Получить locations, соответствующие интересам пользователя

Lab2_be / User / get high recommend location

GETlocalhost:8000/v1/users/recommend-location

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

```
1 {
2   ...// "min_rating": 0
3 }
```

BodyCookiesHeaders (8)Test Results

PrettyRawPreviewVisualizeJSON

```
1 [
2   {
3     "id": "2e9f37aa-fe48-40e0-9152-18fea40eca84",
4     "name": "Bai chay",
5     "address": "123 abc",
6     "rating": 5,
7     "createdAt": "2024-04-09T17:24:48.573Z",
8     "updatedAt": "2024-04-09T17:24:48.573Z"
9   },
10  {
11    "id": "c31802d2-ab6d-4780-8a32-cd0bd53ada83",
12    "name": "Dambri",
13    "address": "12 xxx",
14    "rating": 2,
15    "createdAt": "2024-04-13T19:24:12.290Z",
16    "updatedAt": "2024-04-17T09:00:34.581Z"
17  }
18 ]
```

Получить предложения, соответствующие интересам пользователей

Lab2_be / User / Get offer for current user

GETlocalhost:8000/v1/offers/me

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

```
1 {
2   ...// "max_price": 181,
3   ...// "type": "ticket"
4 }
```

BodyCookiesHeaders (8)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   {
3     "id": "2c629b87-3d7e-41b6-ab44-8a8b64f2dab3",
4     "name": "giam gia 58n",
5     "description": "giam gia cuc hot tu ngay den ngay",
6     "type": "ticket",
7     "price": 80,
8     "location_id": "2e9f37aa-fe48-40e0-9152-18fea40eca84",
9     "createdAt": "2024-04-09T17:30:00.935Z",
10    "updatedAt": "2024-04-09T17:30:00.935Z"
11  },
12  {
13    "id": "5c6e8b7c-4322-4641-8a3e-54a98afc05d9",
14    "name": "giam gia xx",
15    "description": "giam gia dambri xx",
16    "type": "DISCOUNT",
17    "price": 500,
18    "location_id": "c31802d2-ab6d-4780-8a32-cd0bd53ada83",
19    "createdAt": "2024-04-14T16:50:54.945Z",
20    "updatedAt": "2024-04-14T16:50:54.945Z"
21  },
22  {
23    "id": "29ef4246-4de7-46bf-9a07-6c13c64aad73",
24    "name": "dambri mua 2 duoc 3",
25    "description": "mua 2 ve dambri duoc 3 ve",
26    "type": "BONUS",
27    "price": 280,
28    "location_id": "c31802d2-ab6d-4780-8a32-cd0bd53ada83",
29    "createdAt": "2024-04-14T16:51:40.188Z",
30    "updatedAt": "2024-04-14T16:51:40.188Z"
31  }
32 }
```

Предложения по запросам соответствуют интересам пользователей

HTTP Lab2_be / User / Get offer for current user

GET localhost:8000/v1/offers/me

Params Authorization Headers (9) Body Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL [JSON](#)

```
1 {
2   "max_price": 50,
3   "type": "ticket"
4 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": "2c529b87-3d7e-41b6-ab44-8a8b54f2dab3",
4     "name": "giam gia 50%",
5     "description": "giam gia cuc hot tu ngay den ngay",
6     "type": "ticket",
7     "price": 50,
8     "location_id": "2e9f37aa-fe48-40e0-9152-18fea40eca84",
9     "createdAt": "2024-04-09T17:30:08.935Z",
10    "updatedAt": "2024-04-09T17:30:08.935Z"
11  }
12 ]
```

Вывод

В рамках 2-й лаборатории мы научились создавать базовый API для приложения, применять шаблонный код и правильно делить структуру проекта, а также через эту лабораторию мы научились эффективно работать в команде.