دانشکده مهندسی کامپیوتر

مهندسی نرم افزار 1

"فاز **5** پروژه"

موضوع پروژه : Facilities & Transportation Service

استاد درس : دکتر کلباسی

نام گروه: Axiom

فهرست مطالب

توضیح جدول‌ها

جدول Users :

توضیح کلی:

این جدول اطلاعات کاربران سیستم را ذخیره می‌کند. هر کاربر دارای یک شناسه یکتا است که از طریق آن شناسایی می‌شود.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
user_id	شناسه یکتای هر کاربر که توسط دیتابیس تولید می‌شود.
name	نام کاربر. نمی‌تواند خالی باشد.
lastname	نام خانوادگی کاربر. نمی‌تواند خالی باشد.
username	نام‌کاربری. نمی‌تواند خالی باشد(باید یکتا باشد)
email	ایمیل کاربر. نمی‌تواند خالی باشد.
phone_number	شماره تلفن کاربر. باید یکتا باشد.
password	پسورد کاربر. نمی‌تواند خالی باشد. باید به صورت هش (Hash) شده ذخیره شود.
created_at	تاریخ و زمان ایجاد حساب کاربری.

داده نمونه(Sample data):

user_id	name	lastname	username	email	phone_number	password	created_at
1	Ali	Rezaei	Ali._.R	Ali1234@gmail.com	09121234567	a10V9J7O5qCjB3UjKNKhvh8eOaE$2ZzHZKvJ0t7H1yX.hMvqrmB4p./YO	2024-11-20 12:00:00
2	Sara	Davoudi	Sarrrrra	Sara2000@yahoo.com	null	a27c74d8a529d3c9d899ab8de7b518f1eae6a5a5008b2e91e44bc5c8f9f91db3	2024-11-22 15:30:00

جدول Places :

توضیح کلی:

جدول Places اطلاعات مکان‌ها را ذخیره می‌کند. هر مکان شامل شناسه یکتا، نوع مکان (مانند رستوران، بیمارستان، موزه و غیره)، آدرس، و موقعیت جغرافیایی (عرض و طول جغرافیایی) است.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
place_id	شناسه یکتای مکان که به صورت UUID ذخیره می‌شود. این فیلد به عنوان کلید اصلی عمل می‌کند.
type	نوع مکان (برای مثال: رستوران، بیمارستان، موزه، تفریحی، هتل). این فیلد باید از لیست مشخصی انتخاب شود (با استفاده از **CHECK**).
city	نام شهر محل مکان.
address	آدرس مکان.
location	مختصات جغرافیایی (عرض و طول جغرافیایی) به صورت GEOGRAPHY (با SRID 4326 برای استفاده از سیستم مختصات جغرافیایی جهانی).

ملاحظات: از GIST Index برای بهینه‌سازی جستجو بر اساس موقعیت جغرافیایی استفاده شده است

(CREATE INDEX idx_places_location ON places USING GIST (location)).

داده نمونه(Sample data):

location	Address	city	type	place_id
POINT(51.3890 35.6895)	Tehran, Main Street	Tehran	Restaurant	1
POINT(49.5892 37.2876)	Rasht, Hospital Road	Rasht	Hospital	2

جدول Places tranlation :

توضیح کلی:

این جدول ترجمه‌های مختلف برای مکان‌ها را ذخیره می‌کند. هر مکان می‌تواند چندین ترجمه داشته باشد، برای مثال به زبان فارسی و انگلیسی.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
place_id	شناسه مکان که به UUID اشاره دارد و به جدول places متصل است.
lang	زبان ترجمه (برای مثال، fa برای فارسی و en برای انگلیسی).
name	نام ترجمه شده مکان.
description	توضیحات ترجمه شده مکان.

ملاحظات: PRIMARY KEY از ترکیب place_id و lang تشکیل شده است تا هر مکان در هر زبان فقط یک ترجمه داشته باشد.

داده نمونه(Sample data):

place_id	lang	name	description
1	fa	رستوران تهران	رستوران معروف در تهران
1	en	Tehran Restaurant	Famous restaurant in Tehran

جدول Events :

توضیح کلی:

این جدول اطلاعات مربوط به رویدادها را ذخیره می‌کند. هر رویداد شامل تاریخ و زمان شروع و پایان، توضیحات، آدرس و مختصات جغرافیایی است.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
event_id	شناسه یکتای رویداد که توسط دیتابیس تولید می‌شود.
start_at	تاریخ و زمان شروع رویداد به صورت TIMESTAMPTZ.
end_at	تاریخ و زمان پایان رویداد به صورت TIMESTAMPTZ.
city	نام شهر محل رویداد.
address	آدرس محل رویداد.
location	مختصات جغرافیایی محل رویداد به صورت GEOGRAPHY.

داده نمونه(Sample data):

location	address	city	end_at	start_at	event_id
POINT(51.3890 35.6895)	Tehran, Music Hall	Tehran	2024-12-01 12:00:00	2024-12-01 10:00:00	1
POINT(49.5892 37.2876)	Rasht, Art Gallery	Rasht	2024-12-10 16:00:00	2024-12-10 14:00:00	2

جدول Events :

توضیح کلی:

این جدول اطلاعات مربوط به رویدادها را ذخیره می‌کند. هر رویداد شامل تاریخ و زمان شروع و پایان، توضیحات، آدرس و مختصات جغرافیایی است.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
event_id	شناسه یکتای رویداد که توسط دیتابیس تولید می‌شود.
lang	زبان ترجمه (برای مثال، **fa**برای فارسی و **en**برای انگلیسی.(
title	عنوان ترجمه شده رویداد.
description	توضیحات ترجمه شده رویداد.

داده نمونه(Sample data):

event_id	lang	title	description
1	fa	کنسرت موسیقی تهران	کنسرت موسیقی در سالن تهران
1	en	Tehran Music Concert	Music concert in Tehran Hall

جدول Images :

توضیح کلی:

این جدول اطلاعات مربوط به تصاویر مکان‌ها یا رویدادها را ذخیره می‌کند.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
image_id	شناسه یکتای تصویر که به صورت UUID ذخیره می‌شود.
target_type	نوع تصویر (place برای مکان، event برای رویداد).
target_id	شناسه هدف تصویر (می‌تواند شناسه مکان یا رویداد باشد).
image_url	لینک تصویر در فضای ذخیره‌سازی.

داده نمونه(Sample data):

image_id	target_type	target_id	image_url
1	place	1	/images/tehran.jpg
2	event	1	/images/concert.jpg

جدول Comments :

توضیح کلی:

این جدول نظرات کاربران برای مکان‌ها و رویدادها را ذخیره می‌کند.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
comment_id	شناسه یکتای نظر که توسط دیتابیس تولید می‌شود.
target_type	نوع هدف نظر (place یا event).
target_id	شناسه مکان یا رویدادی که نظر به آن تعلق دارد.
rating	امتیاز داده شده (بین 1 تا 5).
created_at	تاریخ و زمان ایجاد نظر.

داده نمونه(Sample data):

comment_id	target_type	target_id	rating	created_at
1	place	1	5	2024-11-25 09:00:00
2	event	1	4	2024-11-26 10:00:00

جدول Hotel Details :

توضیح کلی:

این جدول اطلاعات هتل‌ها را ذخیره می‌کند.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
place_id	شناسه مکان (که به جدول places اشاره دارد).
stars	تعداد ستارگان هتل.
price_range	بیشترین مقدار ممکن برای قیمت(به تومان).

داده نمونه(Sample data):

place_id	stars	price_range
1	5	20,000,000
2	3	7,000,000

جدول Restaurant Details :

توضیح کلی:

این جدول اطلاعات رستوران‌ها را ذخیره می‌کند.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
place_id	شناسه مکان (که به جدول places اشاره دارد).
cuisine	نوع غذای رستوران.
avg_price	قیمت متوسط هر وعده غذایی.

داده نمونه(Sample data):

place_id	cuisine	avg_price
1	Iranian	50
2	Italian	70

جدول Museum Details :

توضیح کلی:

این جدول اطلاعات موزه‌ها را ذخیره می‌کند.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
place_id	شناسه مکان (که به جدول places اشاره دارد).
open_at	زمان باز شدن موزه.
close_at	زمان بسته شدن موزه.
ticket_price	قیمت بلیط موزه.

داده نمونه(Sample data):

place_id	open_at	close_at	ticket_price
1	09:00	17:00	20
2	10:00	18:00	25

جدول Place Amenities :

توضیح کلی:

جدول place_amenities اطلاعات مربوط به امکانات موجود در مکان‌ها (مانند هتل‌ها، رستوران‌ها، بیمارستان‌ها و غیره) را ذخیره می‌کند. برای هر مکان، می‌توان چندین امکانات مختلف (مانند پارکینگ، استخر، اینترنت بی‌سیم) را ذخیره کرد.

قیدها و توضیحات هر ویژگی:

ویژگی	توضیح
place_id	شناسه مکان که به UUID ذخیره می‌شود. این فیلد به جدول places اشاره دارد و از کلید خارجی برای ارتباط با آن استفاده می‌شود.
amenity_name	نام ویژگی یا امکانات (برای مثال: پارکینگ، استخر، اینترنت).

داده نمونه(Sample data):

place_id	amenity_name
1	Parking
1	Pool

جدول Route Logs :

توضیح کلی:

جدول route_logs اطلاعات مربوط به سفرهای انجام شده توسط کاربران را ذخیره می‌کند. هر سفر شامل اطلاعاتی مانند مکان مبدا، مقصد، نوع حمل‌ونقل و تاریخ و زمان سفر است.

قیدها و توضیحات هر ویژگی:

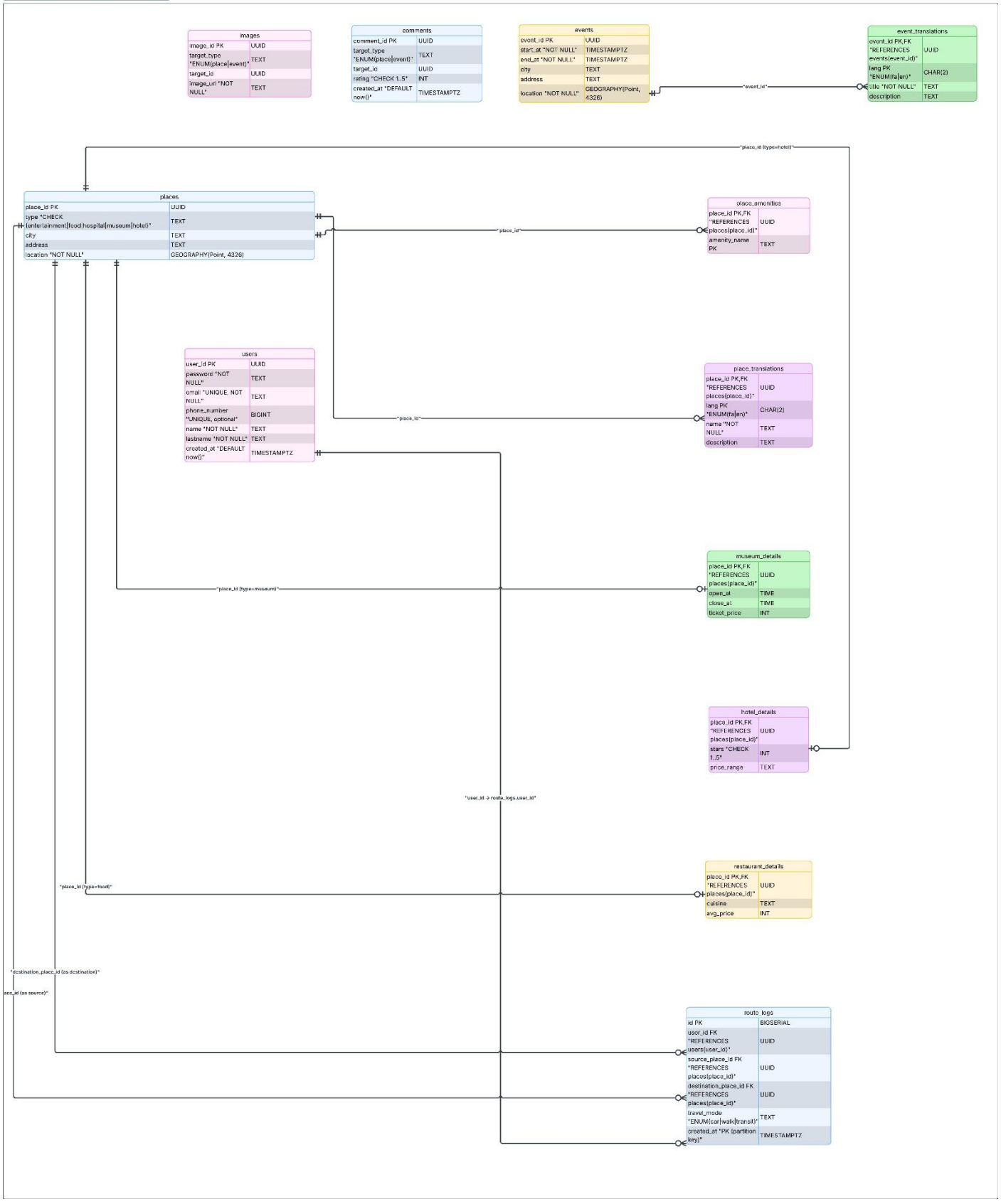
ویژگی	توضیح
id	شناسه یکتای سفر که به صورت BIGSERIAL تولید می‌شود.
user_id	شناسه کاربر که به UUID اشاره دارد و به جدول users متصل است.
source_place_id	شناسه مکان مبدا که به UUID اشاره دارد و به جدول places متصل است.
destination_place_id	شناسه مکان مقصد که به UUID اشاره دارد و به جدول places متصل است.
travel_mode	نوع حمل‌ونقل که باید یکی از مقادیر ''car، walk'' یا 'transit' باشد.
created_at	تاریخ و زمان ایجاد سفر به صورت TIMESTAMPTZ.

داده نمونه(Sample data):

created_at	travel_mode	destination_place_id	source_place_id	user_id	id
2024-11-20 10:00:00	car	5	2	1	1
2024-11-22 08:00:00	transit	52	30	5	2

ER Diagram

Location-Based City Guide ERD



کد جدول‌ها

```
CREATE EXTENSION IF NOT EXISTS postgis;
CREATE EXTENSION IF NOT EXISTS pgcrypto; -- برای UUID

CREATE TABLE users (
  user_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  -- شناسه یکتای کاربر که به صورت UUID تولید می‌شود.
  name TEXT NOT NULL,
  lastname TEXT NOT NULL,
  username TEXT NOT NULL UNIQUE,
  password TEXT NOT NULL,
  email TEXT NOT NULL UNIQUE,
  phone_number TEXT UNIQUE,
  created_at TIMESTAMPTZ DEFAULT now()
);

CREATE TABLE places (
  place_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  type TEXT NOT NULL CHECK (type IN ('entertainment', 'food', 'hospital', 'museum', 'hotel')),
  city TEXT,
  address TEXT,
  location GEOGRAPHY(Point, 4326) NOT NULL
);

CREATE INDEX idx_places_location ON places USING GIST (location);
CREATE INDEX idx_places_type ON places(type);
```

```
CREATE TABLE place_translations (
  place_id UUID REFERENCES places(place_id) ON DELETE CASCADE,
  lang CHAR(2) CHECK (lang IN ('fa', 'en')),
  name TEXT NOT NULL,
  description TEXT,
  PRIMARY KEY (place_id, lang)
);

CREATE TABLE events (
  event_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  start_at TIMESTAMPTZ NOT NULL,
  end_at TIMESTAMPTZ NOT NULL,
  city TEXT,
  address TEXT,
  location GEOGRAPHY(Point, 4326) NOT NULL
);

CREATE TABLE event_translations (
  event_id UUID REFERENCES events(event_id) ON DELETE CASCADE,
  lang CHAR(2) CHECK (lang IN ('fa', 'en')),
  title TEXT NOT NULL,
  description TEXT,
  PRIMARY KEY (event_id, lang)
);

CREATE TABLE images (
  image_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
  target_type TEXT NOT NULL CHECK (target_type IN ('place', 'event')),
  target_id UUID NOT NULL,
  image_url TEXT NOT NULL
);

CREATE TABLE comments (
  comment_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  target_type TEXT NOT NULL CHECK (target_type IN ('place', 'event')),
  target_id UUID NOT NULL,
  rating INT CHECK (rating BETWEEN 1 AND 5),
  created_at TIMESTAMPTZ DEFAULT now()
);

CREATE TABLE hotel_details (
  place_id UUID PRIMARY KEY REFERENCES places(place_id) ON DELETE CASCADE,
  stars INT CHECK (stars BETWEEN 1 AND 5),
  price_range TEXT
);

CREATE TABLE restaurant_details (
  place_id UUID PRIMARY KEY REFERENCES places(place_id) ON DELETE CASCADE,
  cuisine TEXT,
  avg_price INT
);

CREATE TABLE museum_details (
  place_id UUID PRIMARY KEY REFERENCES places(place_id) ON DELETE CASCADE,
  open_at TIME,
```

```
  close_at TIME,
  ticket_price INT
);

CREATE TABLE place_amenities (
  place_id UUID REFERENCES places(place_id) ON DELETE CASCADE,
  amenity_name TEXT,
  PRIMARY KEY (place_id, amenity_name)
);

CREATE TABLE route_logs (
  id BIGSERIAL,
  user_id UUID REFERENCES users(user_id) ON DELETE SET NULL,
  source_place_id UUID REFERENCES places(place_id) ON DELETE CASCADE,
  destination_place_id UUID REFERENCES places(place_id) ON DELETE CASCADE,
  travel_mode TEXT CHECK (travel_mode IN ('car', 'walk', 'transit')),
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  PRIMARY KEY (id, created_at)
) PARTITION BY RANGE (created_at);

//مثالی برای پارتیشن بندی
CREATE TABLE route_logs_2025_01
PARTITION OF route_logs
FOR VALUES FROM ('2025-01-01') TO ('2025-02-01');

CREATE INDEX idx_route_src_dest ON route_logs (source_place_id, destination_place_id);
CREATE INDEX idx_route_created_at ON route_logs (created_at);
```

معماری پروژه

معماری کلی پروژه :

معماری میکروسرویس‌ها (Microservices Architecture) :

- پروژه از معماری میکروسرویس‌ها برای تقسیم سیستم به سرویس‌های کوچک و مستقل استفاده می‌کند. هر میکروسرویس مسئول یک وظیفه خاص است و از طریق APIها با سایر سرویس‌ها ارتباط برقرار می‌کند.
- این معماری به دلیل مزایای مقیاس‌پذیری، استقلال سرویس‌ها و انعطاف‌پذیری برای پروژه انتخاب شده است.

دلایل انتخاب معماری میکروسرویس‌ها:

- مقیاس‌پذیری: هر میکروسرویس به صورت مستقل مقیاس‌پذیر است.
- استقلال سرویس‌ها: تغییرات در یک میکروسرویس بر سایر سرویس‌ها تأثیری نخواهد داشت.
- ساده‌سازی توسعه: تقسیم سیستم به سرویس‌های کوچک، سرعت توسعه و نگهداری را افزایش می‌دهد.

معماری (Model–View–Presenter) MVP در داخل هر میکروسرویس :

برای هر میکروسرویس از معماری MVP برای مدیریت داده‌ها و منطق تجاری استفاده می‌شود. این معماری به تفکیک بهتر وظایف کمک می‌کند و View (نمایش داده‌ها) را از Model (داده‌ها و منطق تجاری) جدا می‌کند.

اجزای معماری MVP در میکروسرویس‌ها:

- Model: مسئول نگهداری داده‌ها و منطق تجاری است. در این معماری، Model مستقیماً به پایگاه داده دسترسی دارد و داده‌ها را ذخیره یا بازیابی می‌کند.
- View: مسئول نمایش داده‌ها به کاربر است. در معماری میکروسرویس‌ها، View ممکن است به UI مربوط باشد که داده‌های پردازش‌شده را نمایش می‌دهد.
- Presenter:
  - مسئول دریافت ورودی‌های کاربر از View و ارسال آن‌ها به Model برای پردازش است.
  - پس از پردازش، Presenter داده‌ها را از Model گرفته و آن‌ها را به View ارسال می‌کند تا نمایش داده شوند.

مزایای استفاده از MVP:

- تفکیک بهتر وظایف: View تنها مسئول نمایش داده‌ها و Presenter تمام منطق تجاری را مدیریت می‌کند.
- آسان‌تر شدن تست‌ها: چون View از منطق تجاری جدا شده است، می‌توان آن را راحت‌تر تست کرد.
- کاهش پیچیدگی: با جدا کردن Presenter از Model و View، منطق تجاری پیچیده به طور مستقل از رابط کاربری مدیریت می‌شود.

معماری داخلی هر میکروسرویس

در داخل هر میکروسرویس، از معماری لایه‌ای استفاده می‌شود. این معماری به تفکیک مسئولیت‌ها و مدیریت داده‌ها کمک می‌کند. لایه‌های مختلف میکروسرویس شامل لایه داده‌ها، لایه منطق تجاری و لایه ارائه هستند.

لایه‌ها در هر میکروسرویس:

- **لایه داده‌ها (Data Layer):**
  - مسئول مدیریت داده‌ها است. این لایه با پایگاه داده ارتباط دارد و داده‌ها را ذخیره یا بازیابی می‌کند.
  - از PostGIS برای داده‌های جغرافیایی و PostgreSQL برای داده‌های ساختاریافته استفاده می‌شود.
- **لایه منطق تجاری (Business Logic Layer):**
  - مسئول پردازش داده‌ها و اجرای الگوریتم‌های پیچیده است.
  - به عنوان مثال، در میکروسرویس مسیریابی، این لایه مسئول محاسبه مسیریابی و زمان تخمینی رسیدن (ETA) است.
- **لایه ارائه (Presentation Layer):**
  - مسئول ارسال داده‌ها به View یا API است.
  - این لایه داده‌های پردازش‌شده را به کلاینت یا سایر میکروسرویس‌ها ارسال می‌کند.

نحوه ارتباط میکروسرویس‌ها و کلاینت-سرور

معماری کلاینت-سرور (Client-Server Architecture)

برای برقراری ارتباط میان کلاینت‌ها (واسط کاربری) و سرور (که درخواست‌ها را به میکروسرویس‌ها ارسال می‌کند)، از معماری کلاینت-سرور استفاده می‌شود.

نحوه عملکرد:

- کلاینت‌ها درخواست‌هایی به سرور ارسال می‌کنند.
- سرور این درخواست‌ها را به میکروسرویس‌های مختلف ارسال می‌کند.
- میکروسرویس‌ها پس از پردازش، پاسخ‌ها را به سرور برمی‌گردانند و سرور آن‌ها را به کلاینت‌ها ارسال می‌کند.

ارتباط بین میکروسرویس‌ها

- میکروسرویس‌ها با استفاده از APIها به یکدیگر متصل می‌شوند.
- هر میکروسرویس پایگاه داده مستقل خود را دارد و از پایگاه داده‌های دیگر استفاده نمی‌کند.

امنیت و مقیاس‌پذیری

امنیت

- احراز هویت و دسترسی: برای امنیت داده‌ها از OAuth 2.0 و JWT برای احراز هویت و مدیریت دسترسی‌ها استفاده می‌شود.
- رمزنگاری داده‌ها: داده‌های حساس مانند گذرواژه‌ها و اطلاعات شخصی کاربران به صورت hash ذخیره می‌شوند.

مقیاس‌پذیری

- برای مقیاس‌پذیری سیستم، از Kubernetes برای مدیریت بار و Load Balancer برای توزیع درخواست‌ها استفاده می‌شود.
- از کشینگ (Caching) برای ذخیره‌سازی داده‌های پر استفاده و افزایش سرعت دسترسی به داده‌ها استفاده می‌شود.

انتخاب معماری برای هر میکروسرویس

در این پروژه، برای هر میکروسرویس از معماری MVP به همراه معماری لایه‌ای داخلی استفاده می‌شود. هر میکروسرویس به صورت مستقل عمل کرده و از پایگاه داده مستقل خود استفاده می‌کند.

- ❖ **میکروسرویس مسیریابی (Routing Service):**
  - **Model**: مسئول داده‌های مسیریابی و الگوریتم‌های مسیریابی.
  - **View**: ارسال نتایج به UI یا API.
  - **Presenter**: دریافت درخواست‌ها از UI و ارسال آن‌ها به Model برای پردازش.

- ❖ **میکروسرویس جستجو و مدیریت مکان‌ها (POI Service):**
  - **Model**: مسئول داده‌های مکان‌ها و فیلتر کردن مکان‌ها.
  - **View**: نمایش مکان‌ها به UI یا API.
  - **Presenter**: مدیریت فیلترها و پردازش جستجوهای ورودی.

- ❖ **میکروسرویس مدیریت رویدادها (Event Management Service):**
  - **Model**: ذخیره‌سازی و پردازش داده‌های رویدادها.
  - **View**: نمایش اطلاعات رویدادها به UI.
  - **Presenter**: مدیریت ورودی‌های مربوط به جستجو و فیلتر کردن رویدادها.

- ❖ **میکروسرویس نظرات و امتیازدهی (Review & Rating Service):**
  - **Model**: پردازش نظرات و امتیازها.
  - **View**: نمایش نظرات و امتیازها به UI.
  - **Presenter**: مدیریت ورودی‌های کاربر برای نوشتن نظر و امتیازدهی.

اعضای گروه

دلارام روحانی 40231026

امیرعباس انتظاری 40231006

بهرام امامی 40231404

ایلیا اسدی زیدآبادی 40231401