Розробка багаторівневого веб-застосування на хмарній платформі «AWS»

Науковий керівник: ст. викл., к.т.н. Черкасов Дмитро Іванович

Виконала: Безрука Анастасія Володимирівна

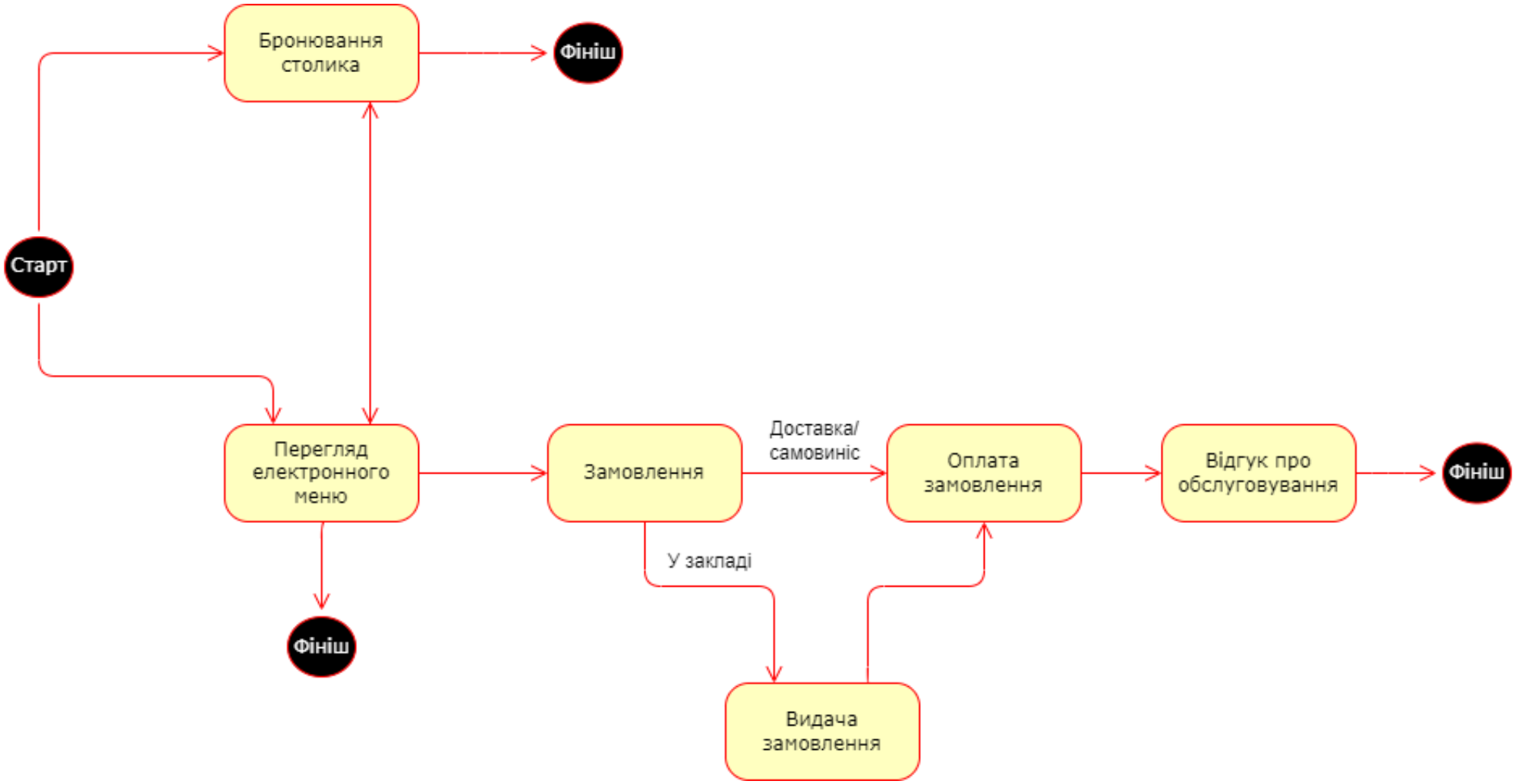
Мета та завдання роботи

- Метою даної роботи є розробити багаторівневе веб-застосування на хмарній платформі «AWS» для ресторанного бізнесу. Додаток містить електронне меню, керування замовленням, систему відгуків та збір статистичних даних про заклад.

Актуальність роботи

Автоматизація процесів	– >	швидше та більш якісне обслуговування
Електронне меню	– >	інтерактивність, простота його оновлення
Система відгуків	– >	вивчення потреб клієнтів

Функціонал застосунку



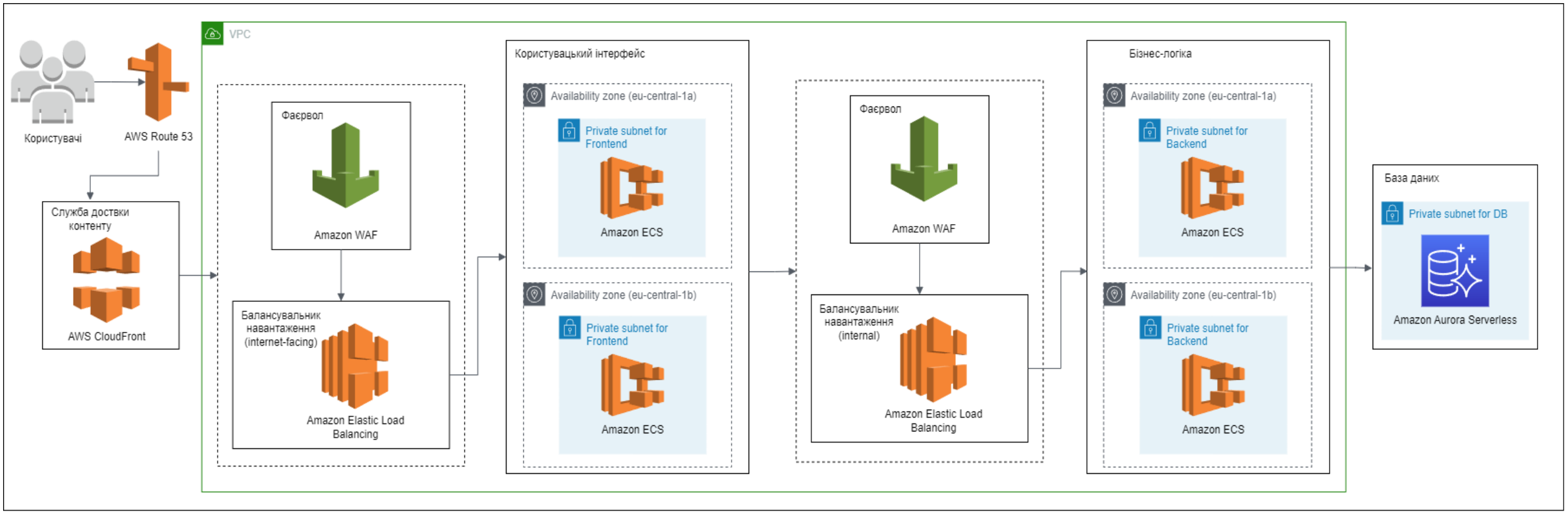
Існуючі рішення

- Choice
- OddMenu
- expirenza

Архітектура, базова технологія та платформа

Архітектура	– >	Багаторівнева
Базова технологія	– >	Контейнери, безсерверні обчислення
Платформа	– >	Хмарні обчислення

Структура системи



Віртуальна приватна хмара (VPC)

Мета: ізоляція ресурсів від несанкціонованого доступу

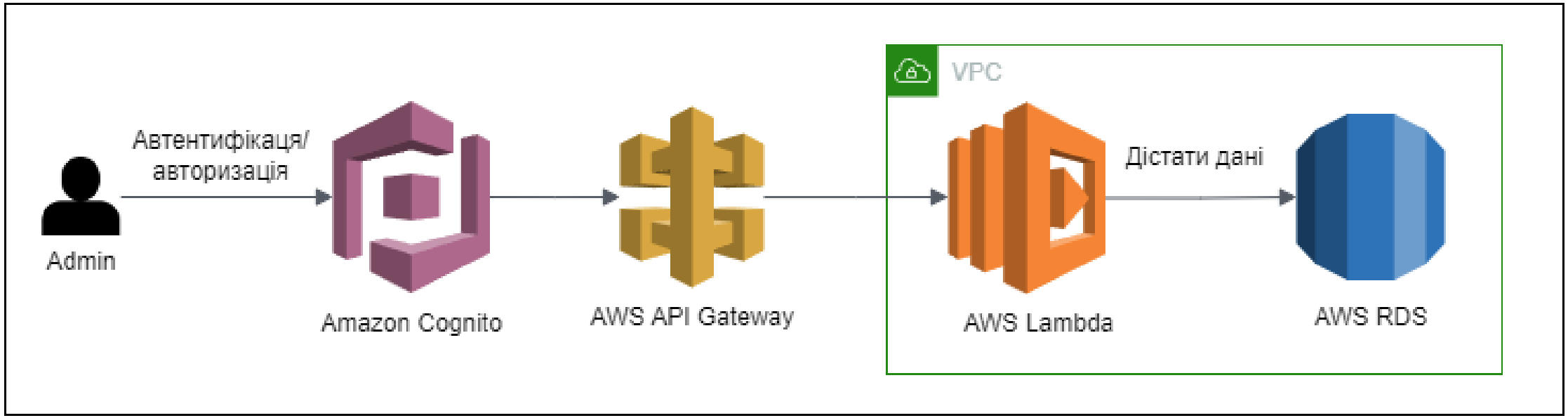
8 підмереж (по 2 в різних зонах):

- рівня бази даних – приватні, не мають доступу в Інтернет;
- бізнес логіки – приватні, з доступом в Інтернет;
- користувацького інтерфейсу – приватні, з доступом в Інтернет;
- публічні підмережі – для доступу до застосунку.

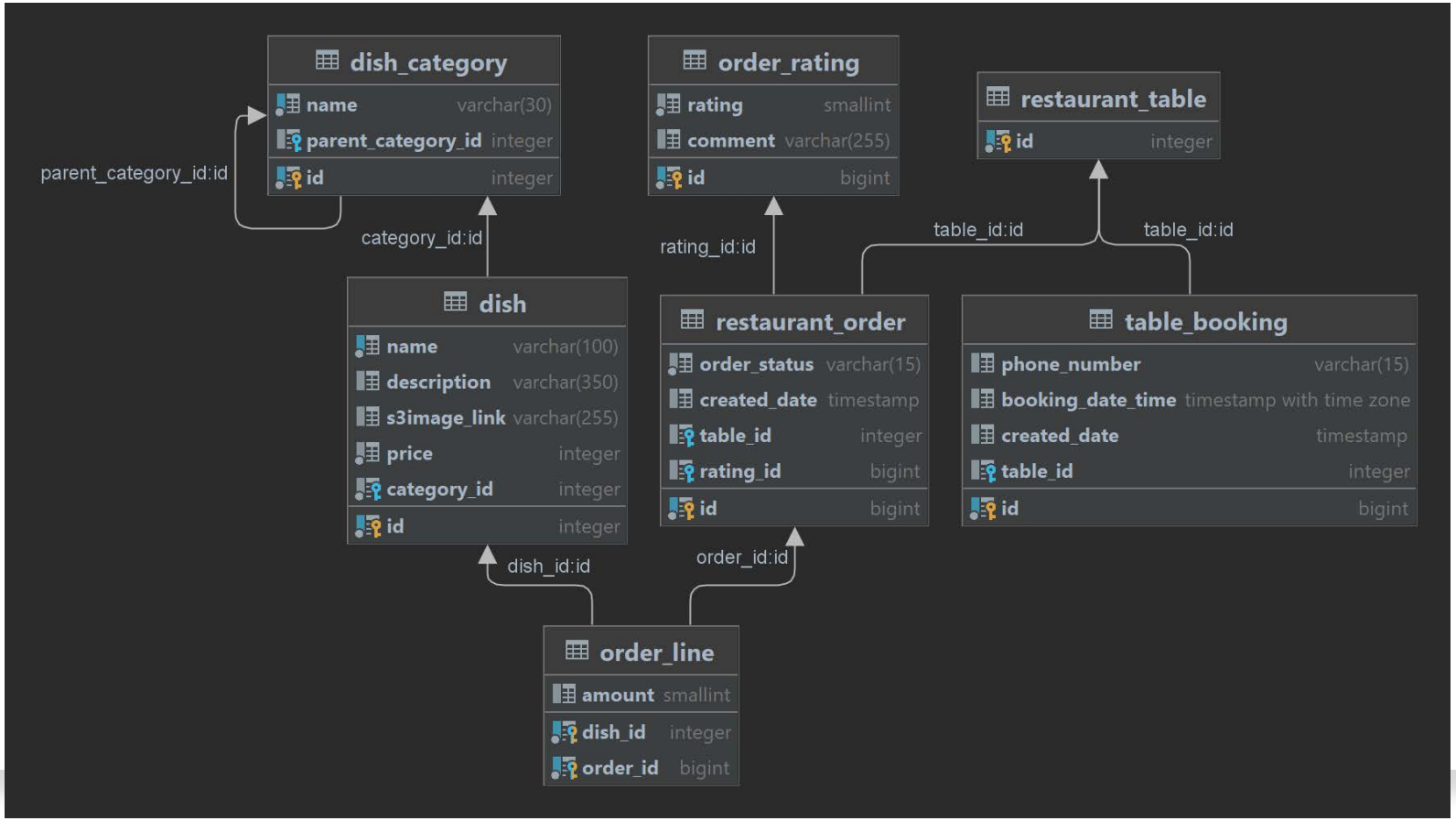
Компоненти системи

База даних	– >	«Amazon Aurora»
Бізнес-логіка	– >	«Elastic Container Service» («ECS»)
Користувацький інтерфейс	– >	«Elastic Container Service» («ECS»)
Балансувальники навантаження	– >	«Elastic Load Balancing»
Фаєрвол	– >	«WAF»
Служба доставки контенту	– >	«CloudFront»
Реєстрація доменного імені	– >	«Route 53»

Обробка даних для обрахунку статистики закладу

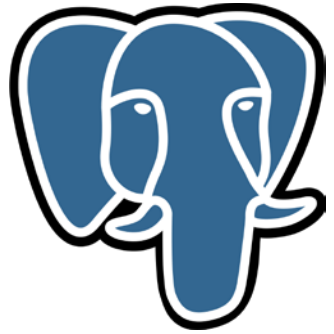


Детальна розробка: реляційна схема



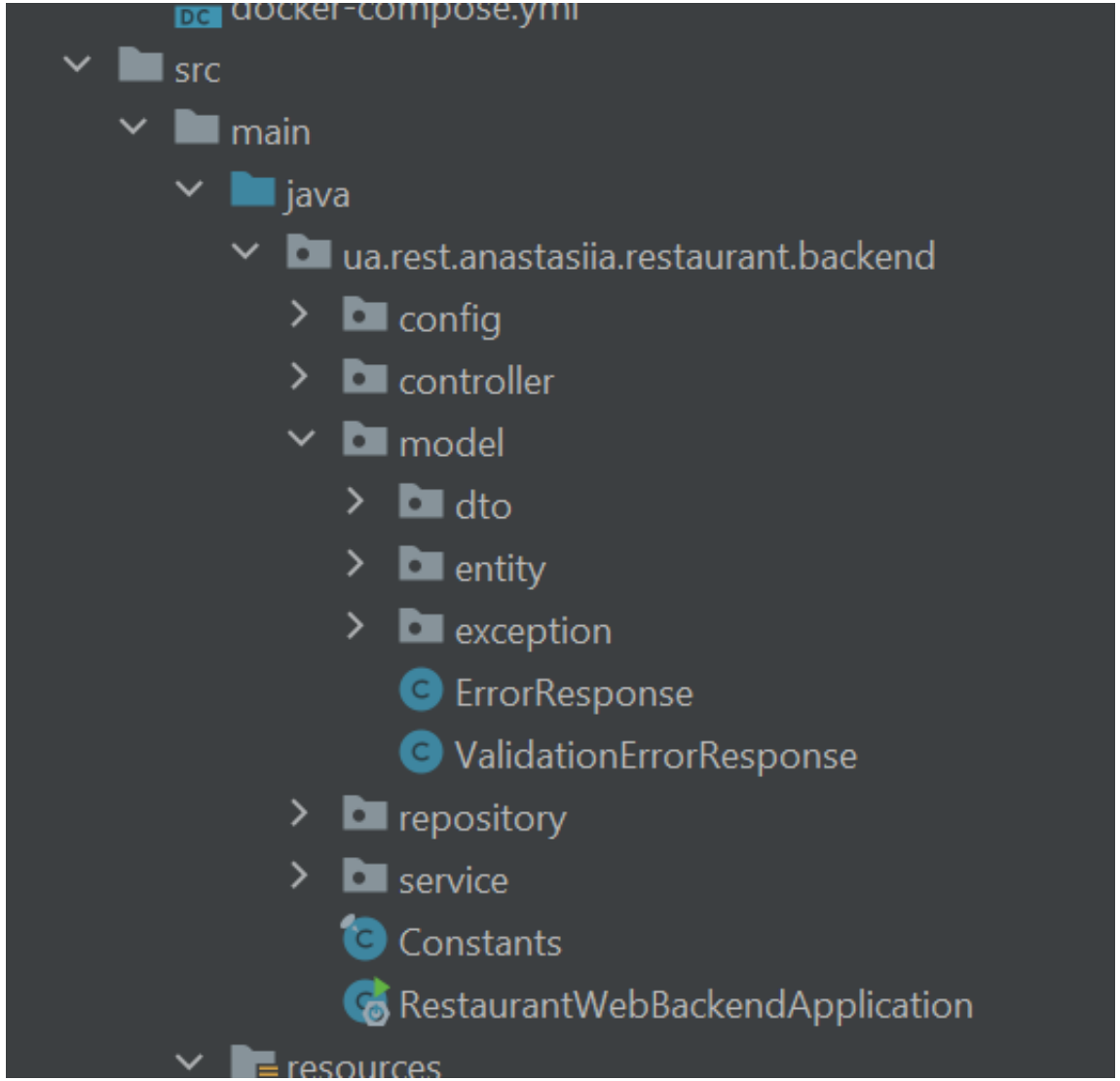
Детальна розробка: рівень бізнес-логіки

- Java 17
- Spring Framework
- Flyway
- PostgreSQL
- Docker



Детальна розробка: рівень бізнес-логіки

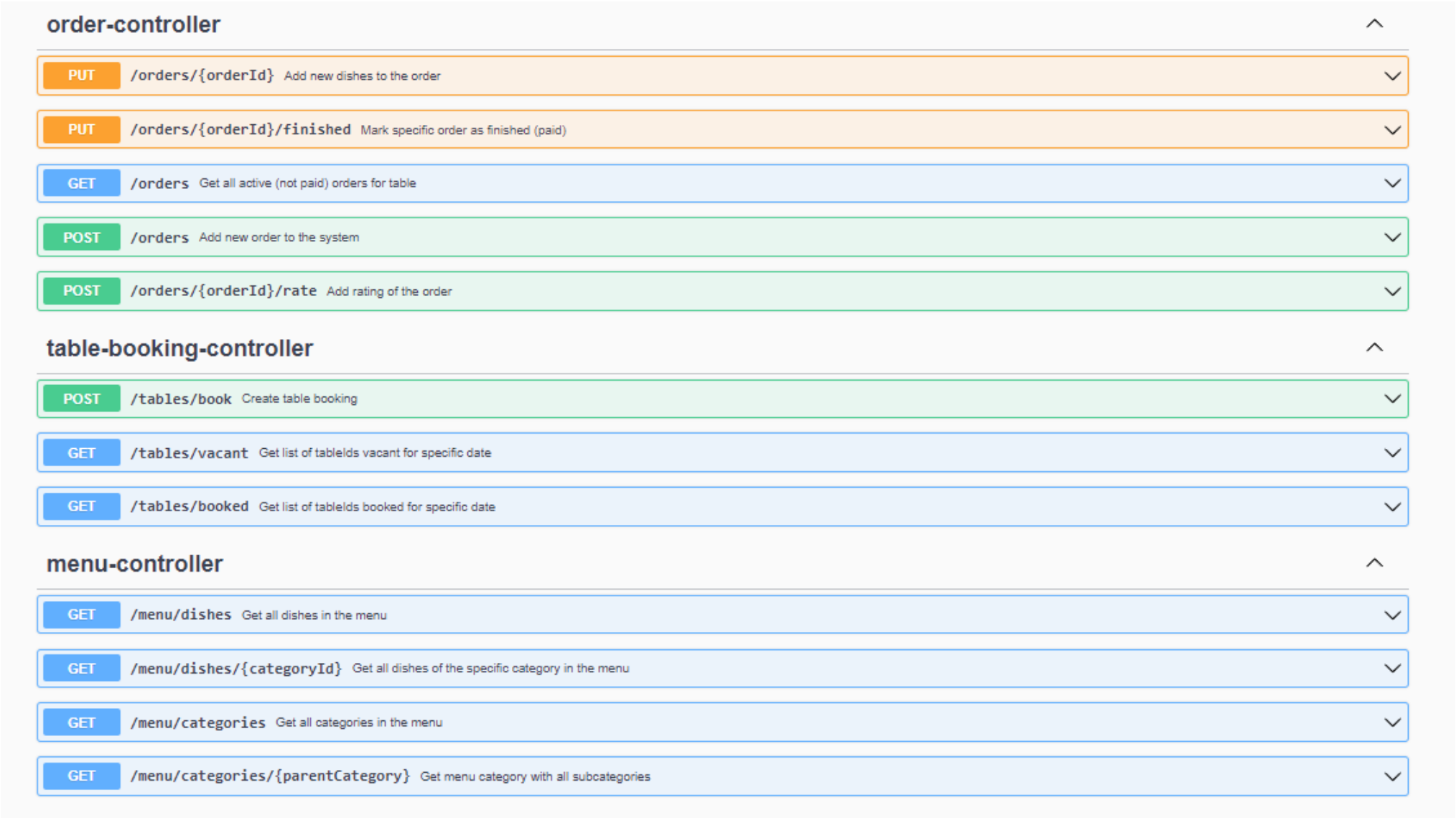
- Моделі - сутності бази даних, класи DTO та користувацькі виключення .
- Репозиторії - класи для доступу до бази даних.
- Сервіси - класи, що містять власне бізнес-логіку.
- Контролери - класи, що обробляють запити REST API.



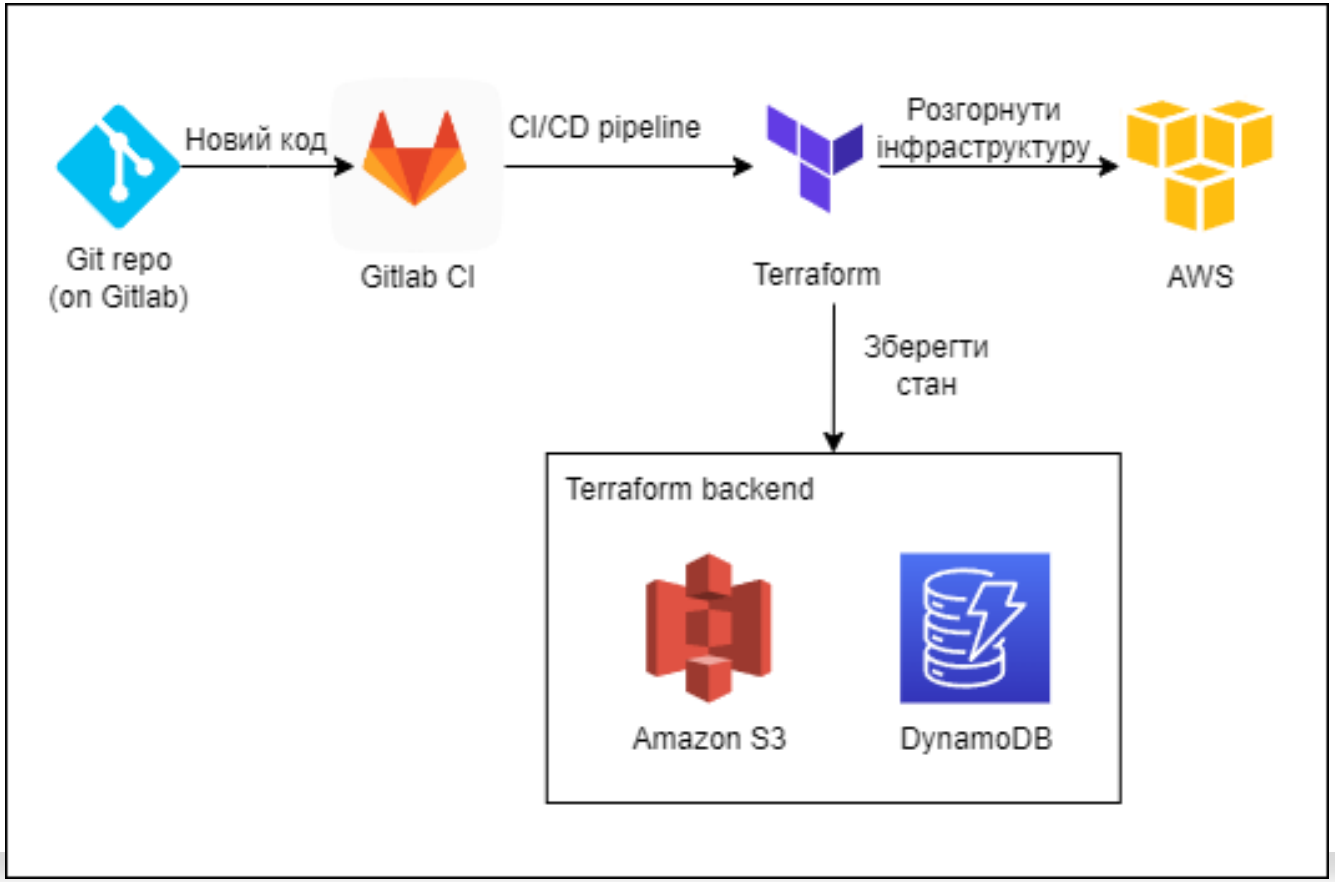
Вимоги до API

- перегляд категорій меню;
- перегляд страв та напоїв за відповідною категорією, включаючи усі підкатегорії;
- створення замовлення;
- перегляд активного замовлення за відповідним столиком;
- додавання нових позицій у замовлення, якщо воно активне;
- позначення замовлення як завершеного (оплаченого);
- додавання відгуку про замовлення, якщо воно завершене;
- перегляд столиків, що ще не зарезервовані для конкретної дати та часу;
- додавання бронювання столику.

Структура API



Автоматизоване розгортання системи



Конвеєр неперервної інтеграції та доставки для «shared-resources»

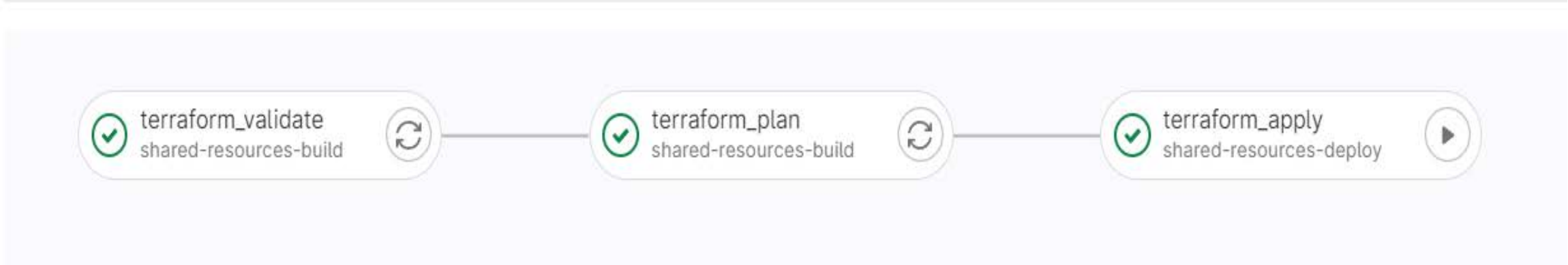
У ньому розгортаються:

- VPC
- ECS кластер
- S3 бакет для зображень страв
- Рівень бази даних

Конвеєр неперервної інтеграції та доставки для «shared-resources»

- етап «shared-resources-build»:
  - завдання «terraform_validate»
  - завдання «terraform_plan»
- етап «shared-resources-deploy»:
  - завдання «terraform_apply»

Відповідні команди інструменту «Terraform»



Конвеєр неперервної інтеграції та доставки для «restaurant_web_backend»

У ньому розгортаються:

- ECS сервіс з масштабуванням
- Балансувальник-навантаження

Конвеєр неперервної інтеграції та доставки для «restaurant_web_backend»

- етап «maven-verify-build»:
  завдання «maven_verify»
  завдання «maven_build

- етап «ecr-build-deploy»:
  завдання «docker_build_image»
  завдання «docker_push_ecr»

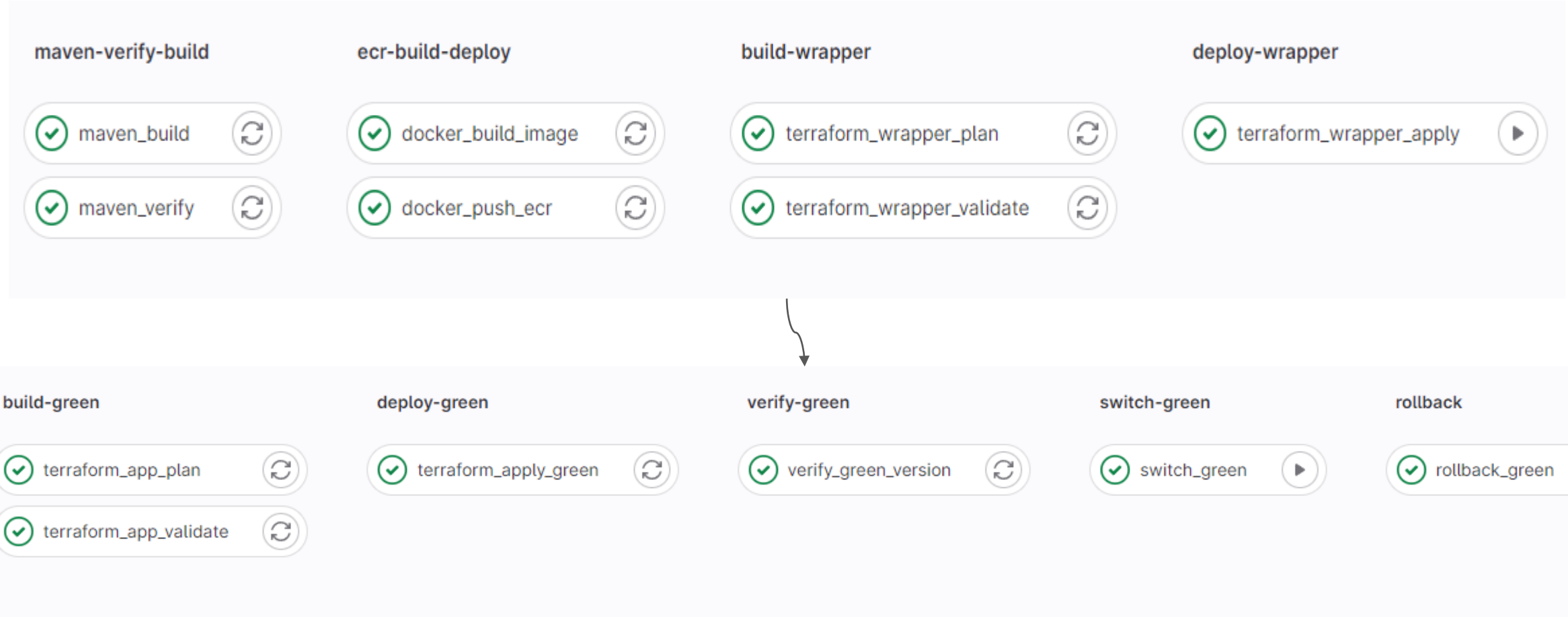
- етап «build-wrapper» такий же як «shared-resource-build»
- етап «deploy-wrapper» такий же як «shared-resource-deploy»

Конвеєр неперервної інтеграції та доставки для «restaurant_web_backend»

- етап «build-green»:
  завдання «terraform_app_validate»
  завдання «terraform_app_plan»
- етап «deploy-green»:
  завдання «terraform_apply_green
- етап «switch-green»:
  завдання «switch_green»
- етап «rollback»
  завдання «rollback_green»

«Terraform» + «AWS CLI»

Конвеєр неперервної інтеграції та доставки для «restaurant_web_backend»



Scan me*

Демонстрація готового API

* - QR-code перенаправляє на API

Висновки

- Оглянуто існуючі рішення з подібним функціоналом
- Проаналізовано різні архітектури та базові технології розгортання, проведено порівняння розміщення інфраструктури у хмарі й на власних ресурсах
- Створено структуру застосунку
- Розроблено детально модуль API
- Було розгорнуто рівень бази даних, бізнес-логіки та балансувальник навантаження. Як наслідок, REST API готове до використання
- Висвітлено автоматичне розгортання системи у хмарному провайдері «AWS», за допомогою «Gitlab CI» та «Terraform»
- Продемонстровано одну із можливих імплементацій стратегії розгортання «Blue/Green» на цій хмарній платформі

Висновки

- На прикладі даного застосунку продемонстровано переваги хмарних обчислень такі як: гнучкість, ефективне масштабування та простота розробки. Завдяки структурним рішенням, до прикладу розміщення у декількох зонах доступу, було досягнено відмовостійкості, а завдяки використанню приватної віртуальної хмари та фаєрволу – безпеки

Дякую за увагу!