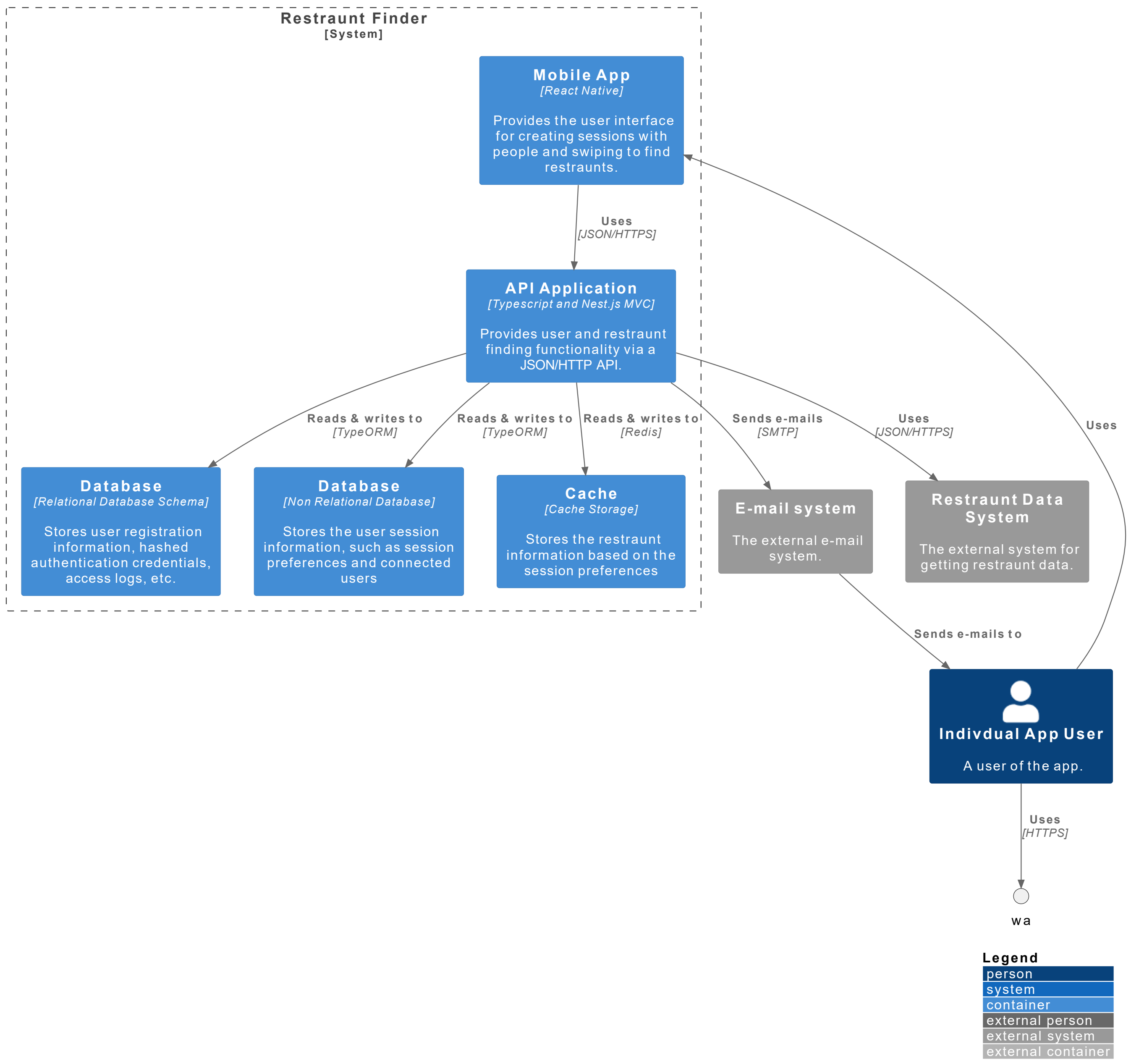
1 Fork Finder System

\1 Fork Finder System



Restraunt Finder
[System]
Mobile App
[React Native]
Provides the user interface for creating sessions with people and swiping to find restraunts.
Uses
[JSON/HTTPS]
API Application
[Typescript and Nest.js MVC]
Provides user and restraunt finding functionality via a JSON/HTTP API.
Reads & writes to
[TypeORM]
Reads & writes to
[TypeORM]
Reads & writes to
[Redis]
Sends e-mails
[SMTP]
Uses
[JSON/HTTPS]
Uses
Database
[Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Database
[Non Relational Database]
Stores the user session information, such as session preferences and connected users
Cache
[Cache Storage]
Stores the restraunt information based on the session preferences
E-mail system
The external e-mail system.
Restraunt Data System
The external system for getting restraunt data.
Sends e-mails to
Indivdual App User
A user of the app.
Uses
[HTTPS]
wa
Legend
person
system
container
external person
external system
external container

Level 2: Container Diagram

Context Scope

This diagram outlines the structure and major components of the **Restaurant Finder App**. It provides a high-level view of the system's containers (frontend, backend, database, etc.), external systems it interacts with (email, restaurant data), and how these components communicate with one another.

Containers and Relationships

People

- **Individual App User (IAU)**: A person who uses the app to find restaurants and manage sessions.

External Systems

- **E-mail System (ES)**: An external system that sends notifications and emails to users (e.g., for registration confirmation, session updates).
- **Restaurant Data System (RDS)**: An external system used to fetch restaurant data based on user preferences and location (e.g., Yelp API, Google Places).

Internal Containers

- **Mobile App (MA)**:
    - **Technology**: React Native
    - **Description**: The mobile app provides the user interface for creating sessions, inviting friends, and swiping to find restaurants.
- **Database (Relational) (DB)**:
    - **Technology**: Relational Database Schema
    - **Description**: Stores persistent user information, such as registration data, hashed authentication credentials, and access logs.
- **Non-Relational Database (NRDB)**:
    - **Technology**: NoSQL (Non-Relational) Database
    - **Description**: Stores user session data, including session preferences, active sessions, and connected users.
- **Cache Storage (CACHE)**:
    - **Technology**: Redis
    - **Description**: Temporary storage for restaurant data based on session preferences to reduce repeated API calls to external systems.
- **API Application (API)**:
    - **Technology**: Typescript, Nest.js MVC
    - **Description**: Handles core functionality including user authentication, restaurant data retrieval, and session management. Provides this functionality via a JSON/HTTP API.

Container Communication

- **API ↔ E-mail System**: Sends emails through the external email system via SMTP.
- **API ↔ Restaurant Data System**: Fetches restaurant data from the external system using JSON/HTTPS requests.
- **API ↔ Relational Database**: Reads and writes user and session data to the relational database using TypeORM.

- **API ↔ Non-Relational Database**: Reads and writes session-specific data to the NoSQL database using TypeORM.
- **API ↔ Cache Storage**: Interacts with Redis to cache restaurant data for faster access.
- **Mobile App ↔ API**: Communicates with the API to authenticate users, fetch restaurant data, and manage sessions via JSON/HTTPS requests.

People Communication

- **Individual App User ↔ Mobile App**: Users interact with the mobile app over HTTPS to perform various actions.
- **E-mail System ↔ Individual App User**: The external email system sends emails to the user (e.g., for registration confirmation or session updates).

Relationships

- **MA → API**: The mobile app communicates with the API using JSON/HTTPS to retrieve restaurant data and manage sessions.
- **API → ES**: The API communicates with the email system over SMTP to send emails to users.
- **API → RDS**: The API fetches restaurant data from external sources via JSON/HTTPS.
- **API → DB**: The API reads from and writes to the relational database using TypeORM.
- **API → NRDB**: The API interacts with the NoSQL database to store and retrieve session information.
- **API → CACHE**: The API caches restaurant data in Redis based on session preferences to avoid redundant external API calls.

Intended Audience

This diagram is intended for **technical staff**, including software architects, developers, and operations/support personnel. It provides an overview of how the containers within the system are structured and communicate with each other.