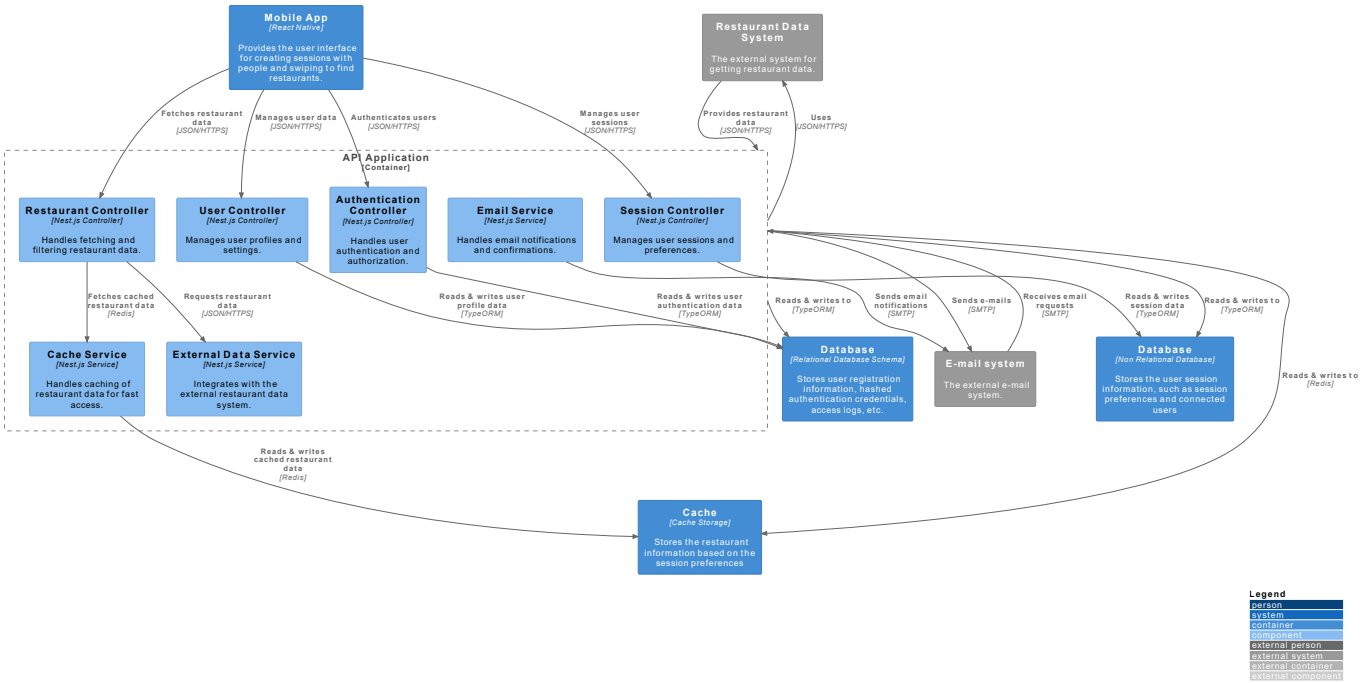
# API Application

\1 Fork Finder System\API Application




# Level 3 Component Diagram: API Application

## Overview

This component diagram decomposes the **API Application** container, illustrating its internal components, their responsibilities, technologies used, and interactions. The purpose of this diagram is to provide software architects and developers with an understanding of the internal architecture of the API container, and how it connects to other containers and external systems.

### Components in Scope:

1. **Authentication Controller**
2. **User Controller**
3. **Session Controller**
4. **Restaurant Controller**
5. **Email Service**
6. **External Data Service**
7. **Cache Service**

### Supporting Elements:

- **Relational Database (DB)**
- **Non-Relational Database (NRDB)**
- **Cache Storage (Cache)**
- **External E-mail System (E-mail)**

- **Restaurant Data System (RDS)**

---

## Components

### 1. **Authentication Controller**

- **Technology:** Nest.js Controller
- **Responsibilities:**
  - Manages user authentication and authorization.
  - Verifies user credentials and issues tokens for secure access to the system.
- **Interactions:**
  - **Reads and writes** authentication data from/to the **Relational Database** (DB) using **TypeORM**.
  - Interacts with **Mobile App** (ma) to authenticate users via **JSON/HTTPS**.
  - Sends authentication tokens to the **Mobile App**.

### 2. **User Controller**

- **Technology:** Nest.js Controller
- **Responsibilities:**
  - Manages user profile data, such as registration, profile updates, and settings.
- **Interactions:**
  - **Reads and writes** user profile data from/to the **Relational Database** (DB) using **TypeORM**.
  - Communicates with the **Mobile App** (ma) to manage user data via **JSON/HTTPS**.

### 3. **Session Controller**

- **Technology:** Nest.js Controller
- **Responsibilities:**
  - Manages user sessions, including session creation, session preferences, and user connections.
- **Interactions:**
  - **Reads and writes** session data from/to the **Non-Relational Database** (NRDB) using **TypeORM**.
  - Provides session information to the **Mobile App** (ma) through **JSON/HTTPS**.

### 4. **Restaurant Controller**

- **Technology:** Nest.js Controller
- **Responsibilities:**
  - Handles the fetching, filtering, and presentation of restaurant data.
- **Interactions:**
  - **Fetches cached restaurant data** from the **Cache Service** (Redis) for faster access.
  - **Requests new restaurant data** from the **External Data Service** (RDS) via **JSON/HTTPS**.
  - Communicates with the **Mobile App** (ma) to display restaurant information via **JSON/HTTPS**.

### 5. **Email Service**

- **Technology:** Nest.js Service
- **Responsibilities:**
  - Sends email notifications and confirmations to users, such as registration and session updates.

- **Interactions:**
  - **Sends email requests** to the external **E-mail System** (SMTP).
  - Sends email notifications to users as needed.

### 6. **External Data Service**

- **Technology:** Nest.js Service
- **Responsibilities:**
  - Integrates with external systems to fetch up-to-date restaurant data.
- **Interactions:**
  - **Requests restaurant data** from the **Restaurant Data System** (RDS) via **JSON/HTTPS**.
  - Sends the fetched data to the **Restaurant Controller** for further processing.

### 7. **Cache Service**

- **Technology:** Nest.js Service
- **Responsibilities:**
  - Manages caching of restaurant data for fast retrieval, ensuring that restaurant-related requests are handled quickly.
- **Interactions:**
  - **Reads and writes** cached restaurant data from/to the **Cache Storage** (Redis).
  - Communicates with the **Restaurant Controller** to provide cached restaurant data when requested.

---

## Relationships

### Internal Component Interactions:

- The **Authentication Controller** interacts with the **Relational Database** for user authentication data and communicates with the **Mobile App** for user authentication.
- The **User Controller** and **Session Controller** interact with the **Relational Database** and **Non-Relational Database** for profile and session management, respectively.
- The **Restaurant Controller** uses the **Cache Service** for fast data retrieval and fetches restaurant data from the **External Data Service** when needed.
- The **Email Service** sends email notifications to users via the **E-mail System** (SMTP).
- The **External Data Service** fetches data from the **Restaurant Data System** (RDS) and sends the data to the **Restaurant Controller** for further processing.

### External Interactions:

- The **API Application** communicates with the **E-mail System** and **Restaurant Data System** via **JSON/HTTPS** for restaurant data and email notifications.
- The **API** interacts with the **Cache Storage** (Redis) and **Databases** (Relational and Non-Relational) for data persistence and fast access.

---

## Conclusion

This **Level 3: Component Diagram** provides a detailed view of the **API Application** container's internal components, their responsibilities, and how they interact with each other and external systems. This breakdown aids software architects and developers in understanding the modular structure of the API container and its connection to the broader system architecture.