20241 - BÀI TẬP LỚN
IT4788 – Phát triển ứng dụng đa nền tảng

Thiết kế phần Backend cho ứng dụng di động đa nền tảng "Đi chợ tiện lợi"

Bước 1: Chọn nền tảng công nghệ backend

- Cơ sở để lựa chọn công nghệ
  - *Yêu cầu chức năng*: Bao gồm các tính năng cụ thể mà ứng dụng cần hỗ trợ, ngoài ra còn có các chức năng riêng biệt của ứng dụng di động đa nền tảng như: xử lý dữ liệu, thông báo đẩy, tương tác thời gian thực,…
  - *Yêu cầu phi chức năng*: Khả năng mở rộng, hiệu suất, bảo mật, và độ tin cậy. Những yếu tố này ảnh hưởng lớn đến cách thiết kế kiến trúc và lựa chọn công nghệ.
- Các công nghệ cần xác định
  - Ngôn ngữ lập trình cho backend (có thể là framework để phát triển backend)
  - Cơ sở dữ liệu (loại mô hình dữ liệu phù hợp với yêu cầu)
    - SQL (quan hệ): MySQL, PostgreSQL,… cho việc quản lý dữ liệu có cấu trúc.
    - NoSQL (phi quan hệ): MongoDB, Firebase,… cho dữ liệu không có cấu trúc hoặc yêu cầu mở rộng tốt.
  - Cơ sở hạ tầng cho triển khai (Infrastructure): máy chủ và các dịch vụ hỗ trợ cho triển khai backend
    - Ví dụ: AWS, Google Cloud, hoặc Heroku,…

Bước 2: Thiết kế kiến trúc backend

- Xác định kiểu kiến trúc: quyết định tính linh hoạt, khả năng bảo trì, và khả năng mở rộng
  - Ví dụ Kiến trúc phân lớp
    - Lớp API (Giao diện lập trình ứng dụng): Cung cấp giao diện cho ứng dụng di động giao tiếp với backend
    - Lớp dịch vụ: Xử lý logic nghiệp vụ chính của ứng dụng, bao gồm xác thực, xử lý thanh toán, hoặc các quy trình xử lý dữ liệu
    - Lớp dữ liệu: Quản lý việc lưu trữ và truy vấn dữ liệu từ cơ sở dữ liệu
    - …

- Ví dụ Kiến trúc microservices: phát triển và triển khai nhanh chóng, dễ dàng scale từng service độc lập, linh hoạt trong việc chọn công nghệ cho từng service. Nhược điểm là phức tạp.
    - API Gateway làm điểm vào duy nhất, Các service độc lập, có thể scale riêng biệt, Giao tiếp không đồng bộ giữa các service
    - Các thành phần chính (không bắt buộc phải có đầy đủ, tùy theo từng bài toán với yêu cầu cụ thể)
        1. API Gateway
        2. Authentication Service
        3. Core Business Services
        4. Message Queue
        5. Database Services
        6. Cache Layer
        7. Storage Service
        8. Monitoring & Logging
        9. …
- Biểu diễn kiến trúc Backend: xây dựng biểu đồ phù hợp cho việc thể hiện kiến trúc backend
    - Component Diagram (Biểu đồ thành phần)
    - Deployment Diagram (Biểu đồ triển khai)
    - UML Package Diagram (Biểu đồ gói UML)
    - Architecture Diagram (Biểu đồ kiến trúc)
        - Ví dụ biểu đồ kiến trúc phân lớp, biểu đồ kiến trúc microservices,…
    - …
    - Có thể kết hợp thêm một số biểu đồ để diễn giải sự tương tác frontend / client với backend hoặc sự tương tác giữa các thành phần của backend
        - Sequence Diagram (Biểu đồ tuần tự)
        - Data Flow Diagram (Biểu đồ luồng dữ liệu)
        - Event-Driven Architecture Diagram (Biểu đồ luồng sự kiện)

Bước 3: Thiết kế API

- Lựa chọn API:
    - RESTful

- GraphQL
- SOAP (Simple Object Access Protocol)
- …

RESTful API là lựa chọn hàng đầu do dễ phát triển và sử dụng rộng rãi cho hầu hết các ứng dụng di động, đảm bảo tính đơn giản và khả năng mở rộng.

- Thiết kế API: một số lưu ý
  - Tuân thủ các nguyên tắc của RESTful
  - Đánh phiên bản cho API (API versioning) để thuận tiện cho việc mở rộng sau này
  - Kiểm soát lỗi nhất quán (định nghĩa các mã lỗi và thông điệp tương ứng để sử dụng nhất quán trong ứng dụng)
  - Xác thực dữ liệu hợp lệ cho Request/Response
  - Xây dựng tài liệu API (API documentation)

Bước 4: Thiết kế cơ sở dữ liệu

- Xây dựng sơ đồ biểu diễn cho Cơ sở dữ liệu theo mô hình dữ liệu đã lựa chọn (mô hình quan hệ hoặc phi quan hệ)
- Xây dựng đặc tả dữ liệu
  - Đặc tả cho các bảng dữ liệu
  - Đặc tả cho JSON documents
  - Các ràng buộc đối với dữ liệu
- Các vấn đề thiết kế CSDL khác
  - Tạo UUIDs nhất quán và tự động cho các thực thể dữ liệu
  - Các giới hạn về kích thước của thực thể dữ liệu
  - Lưu trữ dữ liệu đa phương tiện
  - …

Bước 5: Thiết kế giải pháp Push notification

- *5.1 Xác định mục tiêu và loại thông báo đẩy*
  - Việc gửi thông báo đẩy nhằm mục tiêu gì? → ví dụ: cập nhật thông tin quan trọng (tin tức, khuyến mãi), nhắc nhở người dùng về sự kiện hoặc hoạt động (lịch hẹn, deadline), gửi thông tin tương tác, ví dụ, nhắc nhở về các tin nhắn mới, Cá nhân hóa thông báo dựa trên hành vi của người dùng,…
  - Phân loại thông báo đẩy:

        - Transactional notifications: Thông báo sự kiện xảy ra (như tin nhắn, giao dịch).
        - Promotional notifications: Thông báo quảng cáo và khuyến mãi.
        - System notifications: Cập nhật hệ thống hoặc yêu cầu người dùng thực hiện hành động.
        - …
- *5.2 Lựa chọn dịch vụ Push Notification*
    - Lưu ý lựa chọn dịch vụ phù hợp để tích hợp vào ứng dụng di động đa nền tảng
        - Firebase Cloud Messaging (FCM)
        - Apple Push Notification Service (APNs)
        - OneSignal
        - Pusher Beams
        - …
- *5.3 Thiết lập backend để quản lý Push Notification*
    - Tạo API để gửi Push Notification: gửi thông báo đến dịch vụ FCM, APNs hoặc OneSignal. Backend cũng có thể quản lý logic khi nào cần gửi thông báo, cá nhân hóa thông báo cho từng người dùng, và lưu lịch sử thông báo đã gửi.
    - Xây dựng cơ sở dữ liệu quản lý tokens: quản lý và lưu các token thiết bị của người dùng. Mỗi thiết bị sẽ có một token, backend cần lưu trữ và quản lý danh sách token này để gửi thông báo. Ngoài ra cần xử lý các token hết hạn hoặc không còn sử dụng.
- *5.4 Tích hợp trên frontend (ứng dụng di động đa nền tảng)*
    - Đăng ký nhận thông báo từ dịch vụ push: gửi yêu cầu đăng ký và nhận token cho từng thiết bị, ngoài ra có thể cần yêu cầu người dùng cấp quyền nhận thông báo.
    - Xử lý thông báo trong ứng dụng:
        - Foreground notifications: Xử lý thông báo khi ứng dụng đang mở.
        - Background notifications: Quản lý thông báo khi ứng dụng không hoạt động.
    - Cá nhân hóa thông báo cho người dùng **(nâng cao)**
        - Thông báo dựa trên hành vi: Gửi thông báo nhắc nhở dựa trên hành vi trước đó của người dùng (ví dụ, nếu họ đã thêm sản phẩm vào giỏ hàng nhưng chưa mua).
        - Phân loại người dùng: Phân chia người dùng theo nhóm và gửi thông báo phù hợp với từng nhóm đối tượng.

- Gửi thông báo đẩy theo lịch (Scheduled Push Notifications)
- Thông báo dựa trên địa điểm (Geolocation-based Push Notifications): được kích hoạt dựa trên vị trí của người dùng. Ví dụ, khi người dùng đến gần một cửa hàng cụ thể, họ sẽ nhận được thông báo về khuyến mãi.
- …

Bước 6: Thiết kế giải pháp quản lý chế độ offline và chế độ chuyển đổi "Back online"

- *6.1 Phân tích các chức năng cần hỗ trợ offline*
  - Lập danh sách các tính năng mà người dùng vẫn có thể thực hiện khi không có kết nối internet.
  - Định nghĩa các trường hợp "Back online": Xác định các hành vi khi người dùng chuyển từ chế độ offline sang online.
    - Đồng bộ hóa dữ liệu: Khi có kết nối internet, các thay đổi được thực hiện offline cần được đồng bộ với server.
    - Giải quyết xung đột dữ liệu: Nếu có dữ liệu trên server thay đổi trong lúc người dùng offline, cần có cơ chế giải quyết xung đột.
- *6.2 Thiết kế cơ chế lưu trữ dữ liệu cục bộ (Offline storage)*
  - Tùy chọn lưu trữ cục bộ để lưu trữ tạm thời hoặc dài hạn dữ liệu của người dùng khi không có kết nối internet:
    - LocalStorage (trong web app)
    - SQLite (trong mobile app)
    - NoSQL (Realm, PouchDB)
    - File system
    - …
  - Tổ chức cấu trúc lưu trữ dữ liệu:
    - Id duy nhất cho mỗi bản ghi: Mỗi dữ liệu hoặc thực thể cần có một ID duy nhất để đảm bảo không có xung đột khi đồng bộ hóa.
    - Dấu thời gian (timestamps): Ghi nhận thời gian thay đổi hoặc chỉnh sửa dữ liệu để dễ dàng xử lý xung đột.
    - Phiên bản dữ liệu (data versioning): Đánh dấu phiên bản của dữ liệu để xác định đâu là dữ liệu mới nhất trong trường hợp có xung đột.

        - Cached data: Dữ liệu tạm thời được lưu cục bộ để người dùng có thể truy cập khi offline.
        - Pending operations: Các thay đổi của người dùng (thêm, sửa, xóa) khi offline được lưu lại và gửi lên server khi có kết nối.
- *6.3 Phát hiện trạng thái mạng (Network status detection)*
    - Theo dõi trạng thái mạng trong thời gian thực của thiết bị để biết khi nào chuyển sang chế độ offline và khi nào quay lại online → pakage hoặc thư viện có sẵn
    - Quản lý trạng thái chuyển đổi mạng: khi trạng thái mạng thay đổi, cần thực hiện các hành động phù hợp như thông báo cho người dùng và chuyển chế độ hoạt động.
- *6.4 Xử lý đồng bộ dữ liệu (Data synchronization)*
    - Quản lý hàng đợi đồng bộ (Sync Queue): Khi người dùng thực hiện các thao tác trong chế độ offline, các thay đổi cần được lưu trong một hàng đợi đồng bộ (sync queue. Khi có kết nối trở lại, hàng đợi này sẽ được thực thi.
    - Đồng bộ hai chiều (Two-way synchronization):
        - **Client → Server:** Gửi tất cả thay đổi được thực hiện offline lên server
        - **Server → Client:** Cập nhật lại dữ liệu từ server nếu có thay đổi trong thời gian offline
    - Giải quyết xung đột dữ liệu: Trong một số trường hợp, dữ liệu trên server có thể thay đổi trong lúc người dùng offline, dẫn đến xung đột khi đồng bộ trở lại. Có thể chọn các giải pháp như:
        - Client overwrites server: Dữ liệu từ client sẽ ghi đè dữ liệu trên server.
        - Server overwrites client: Dữ liệu trên server sẽ ghi đè dữ liệu cục bộ trên client.
        - Giải pháp xung đột có thông báo: Hiển thị thông báo yêu cầu người dùng chọn dữ liệu nào sẽ được giữ lại.
        - …

Bước 7: Các vấn đề khác

- *Bảo mật backend*
    - Bảo mật API: Triển khai OAuth hoặc JWT cho việc bảo mật các endpoint của API.

    - Mã hóa dữ liệu: Sử dụng HTTPS cho giao tiếp giữa backend và ứng dụng di động. Dữ liệu nhạy cảm phải được mã hóa trong cơ sở dữ liệu và trong quá trình truyền tải.
    - Kiểm tra và xác thực dữ liệu: Đảm bảo tất cả các dữ liệu nhập vào đều được kiểm tra và xác thực để tránh các lỗ hổng bảo mật như SQL Injection hoặc XSS.
- *Tối ưu hóa hiệu suất*
    - Caching: Sử dụng Redis hoặc Memcached để giảm tải cho cơ sở dữ liệu bằng cách lưu trữ tạm các truy vấn dữ liệu.
    - Load Balancing: Phân phối lưu lượng truy cập giữa nhiều máy chủ backend để đảm bảo tính khả dụng cao.
    - Tối ưu hóa cơ sở dữ liệu: Thiết kế cơ sở dữ liệu và các truy vấn một cách hiệu quả để giảm thời gian phản hồi.
- *Giám sát và quản lý backend*
    - Giám sát: Sử dụng các công cụ như Prometheus, Grafana hoặc New Relic để theo dõi hiệu suất và phát hiện các vấn đề tiềm ẩn.
    - Logging: Tích hợp hệ thống log như ELK Stack (Elasticsearch, Logstash, Kibana) để lưu trữ và phân tích log từ backend.
- *Tích hợp CI/CD cho phát triển và triển khai backend*
    - CI/CD (Continuous Integration/Continuous Deployment): Sử dụng các công cụ như Jenkins, GitLab CI, hoặc CircleCI để tự động hóa quá trình kiểm tra, xây dựng, và triển khai backend.
    - Containerization: Sử dụng Docker để đóng gói ứng dụng, giúp quá trình triển khai dễ dàng hơn và giảm thiểu sự phụ thuộc môi trường.

./Hết/.