ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Đồ án đa ngành - CO3109

Báo cáo

PHÁT TRIỂN HỆ THỐNG NHÀ THÔNG MINH DREAM HOME

Giảng viên hướng dẫn: ThS. Mai Đức Trung

Sinh viên thực hiện: 2011364 - Nguyễn Duy Khang
2010206 - Du Thành Đạt
2011128 - Hoàng Nhật Hà
2012945 - Nguyễn Tiến Đạt

TP. Hồ Chí Minh, 06/2023

Mục lục

Danh mục hình vẽ

Danh mục bảng biểu

1 Mở đầu

Nhà thông minh là một khái niệm ngày càng phổ biến trong thời đại công nghệ số hiện nay. Nhà thông minh là nhà có khả năng tự động hóa và điều khiển các thiết bị gia dụng thông qua Internet vạn vật (IOT) và các cảm biến. Nhà thông minh mang lại nhiều lợi ích cho người sử dụng, như tiết kiệm năng lượng, tăng cường an ninh, nâng cao sức khỏe và thoải mái sống.

Để tìm hiểu thêm thêm về công nghệ IOT và các cảm biến thông minh, cũng như ứng dụng kiến thức học được về công nghệ phần mềm, nhóm đã chọn đề tài **Dream Home**.

Qua dự án, nhóm hy vọng đạt được các mục tiêu sau.

Thứ nhất, nhận và hiển thị dữ liệu từ thiết bị.

Thứ hai, kiểm tra dữ liệu nhận được vượt quá ngưỡng cho phép.

Thứ ba, điều khiển thiết bị.

Thứ tư, ghi nhận hoạt động thiết bị.

Thứ năm, sử dụng design pattern trong hiện thực.

2 Phân tích yêu cầu

Nhóm chia làm các yêu cầu chức năng và yêu cầu phi chức năng.

2.1 Yêu cầu chức năng

Nhóm đề xuất các yêu cầu phi chức năng như sau:

- Đo độ sáng, độ ẩm, nhiệt độ trong phòng.
- Hiển thị các giá trị đo được trên màn hình LCD.
- Điều khiển thiết bị trong phòng.
- Hiển thị các giá trị đo được trên app.
- Thêm, bớt các thiết bị.
- Điều chỉnh ngưỡng cảm biến thông báo.
- Kích hoạt các thiết bị khác khi vượt ngưỡng.
- Lưu lịch sử hoạt động của thiết bị.

2.2 Yêu cầu phi chức năng

Nhóm đề xuất các yêu cầu phi chức năng như sau:

- Số liệu update liên tục trong ít nhất 2 phút.
- Số liệu lưu trên server ít nhất 1 năm.
- Ứng dụng chạy được trên Android.
- Người dùng có thể self-host hệ thống.

3 Use-case scenarios

Danh sách use case:

- UC001 · Xem giá trị cảm biến trên LCD (Bảng 1)
- UC002 · Xem giá trị cảm biến trên app (Bảng 2)
- UC003 · Điều khiển thiết bị (Bảng 3)
- UC004 · Thêm cảm biến, thiết bị mới (Bảng 4)
- UC005 · Thiết lập cảm biến, thiết bị (Bảng 5)
- UC006 · Thiết lập ngưỡng cảm biến (Bảng 6)
- UC007 · Thiết lập lịch trình thiết bị (Bảng 7)
- UC008 · Xem lịch sử hoạt động thiết bị (Bảng 8)

3.1 UC001 · Xem giá trị cảm biến trên LCD

Author	Du Thành Đạt
Date	16/02/2023
Description	Xem giá trị cảm biến mới nhất trên màn hình LCD.
Actor	User
Trigger	User xem màn hình LCD.
Pre-condition	Không có
Post-condition	User biết giá trị mới nhất của cảm biến.
Normal flow	1. User xem giá trị mới nhất của cảm biến trên màn hình LCD.
Alternative flow	Không có
Exception flow	Không có

Bảng 1: UC001 · Xem giá trị cảm biến trên LCD

3.2 UC002 · Xem giá trị cảm biến trên app

Author	Hoàng Nhật Hà
Date	16/02/2023
Description	User muốn xem các giá trị độ ẩm, độ sáng, nhiệt độ mà các sensor đang đo được ở thời điểm hiện tại bằng app.
Actor	User
Trigger	User vào màn hình Dashboard
Pre-condition	Không có
Post-condition	User biết được các giá trị hiện tại sensor đo được.
Normal flow	1. App cập nhật các giá trị mà sensor đo được và hiển thị số liệu lên màn hình 2. User đọc các số liệu và biết được thông tin về độ đo mong muốn
Alternative flow	Không có
Exception flow	Tại bước 1, App không kết nối được với server để lấy dữ liệu của sensor. App crash.

Bảng 2: UC002 · Xem giá trị cảm biến trên app

3.3 UC003 · Điều khiển thiết bị

Author	Hoàng Nhật Hà
Date	16/02/2023
Description	User muốn điều khiển, tắt bật, và cài đặt chế độ tự động cho các thiết bị đèn, quạt đã được kết nối với hệ thống thông qua app
Actor	User
Trigger	User vào màn hình Dashboard
Pre-condition	Không có
Post-condition	User tắt bật thiết bị thành công
Normal flow	1. User bật tắt công tắc của thiết bị 2. App nhận thông tin và thực hiện các kết quả điều khiển tương ứng
Alternative flow	Không có
Exception flow	Tại bước 2, App không kết nối được với server để thay đổi trạng thái thiết bị. Thay đổi không được thực hiện.

Bảng 3: UC003 · Điều khiển thiết bị

3.4 UC004 · Thêm cảm biến, thiết bị mới

Author	Nguyễn Tiến Đạt
Date	10/03/2023
Description	User muốn thêm thiết bị mới vào nhà thông minh.
Actor	User
Trigger	User nhấn chọn nút thêm thiết bị trên màn hình Settings
Pre-condition	Không có
Post-condition	User thêm thiết bị thành công
Normal flow	1. User điền tên thiết bị, chọn loại thiết bị. 2. User nhấn nút xác nhận.
Alternative flow	Không có
Exception flow	• Tại bước 3, App không kết nối được với server để thêm thiết bị. Thêm thiết bị không được thực hiện. • Tại bước 3, tên thiết bị bị trùng với thiết bị đã có trong hệ thống. App hiển thị thông báo lỗi và yêu cầu User nhập lại tên thiết bị. Quay lại bước 2. • Tại bước 3, User không nhập tên thiết bị. App hiển thị thông báo lỗi và yêu cầu User nhập lại tên thiết bị. Quay lại bước 2.

Bảng 4: UC004 · Thêm cảm biến, thiết bị mới

3.5 UC005 · Thiết lập cảm biến, thiết bị

Author	Du Thành Đạt
Date	10/03/2023
Description	User muốn thiết lập ngưỡng thông báo cảm biến, tắt mở tự động của thiết bị.
Actor	User
Trigger	User bấm vào thiết bị trên màn hình Settings hoặc Dashboard
Pre-condition	Không có
Post-condition	User thiết lập thiết bị thành công
Normal flow	Tùy loại thiết bị, chuyển đến use-case 006 hoặc 007.
Alternative flow	Không có
Exception flow	• Tại bước 3, App không kết nối được với server để thêm thiết bị. Thêm thiết bị không được thực hiện. • Tại bước 3, tên thiết bị bị trùng với thiết bị đã có trong hệ thống. App hiển thị thông báo lỗi và yêu cầu User nhập lại tên thiết bị. Quay lại bước 2. • Tại bước 3, User không nhập tên thiết bị. App hiển thị thông báo lỗi và yêu cầu User nhập lại tên thiết bị. Quay lại bước 2.

Bảng 5: UC005 · Thiết lập cảm biến, thiết bị

3.6 UC006 · Thiết lập ngưỡng cảm biến

Author	Nguyễn Duy Khang
Date	10/03/2023
Description	User muốn thiết lập ngưỡng thông báo cảm biến.
Actor	User
Trigger	User ở màn hình thiết lập của cảm biến
Pre-condition	Không có
Post-condition	User thiết lập cảm biến thành công
Normal flow	1. User chọn switch bật, tắt thông báo. 2. User chọn ngưỡng thông báo. 3. User chọn các hành động khi cảm biến chạm ngưỡng thông báo. 4. User nhấn nút lưu thiết lập.
Alternative flow	Không có
Exception flow	• Tại bước 4, App không kết nối được với server để lưu thiết lập. Thiết lập không được lưu.

Bảng 6: UC006 · Thiết lập ngưỡng cảm biến

3.7 UC007 · Thiết lập lịch trình thiết bị

Author	Du Thành Đạt
Date	10/03/2023
Description	User muốn thiết lập lịch trình thiết bị.
Actor	User
Trigger	User ở màn hình thiết lập của thiết bị
Pre-condition	Không có
Post-condition	User thiết lập thiết bị thành công
Normal flow	1. User chọn switch bật, tắt chế độ tự động. 2. User chọn lịch trình bật tắt tự động. 3. User nhấn nút lưu thiết lập.
Alternative flow	Không có
Exception flow	• Tại bước 3, App không kết nối được với server để lưu thiết lập. Thiết lập không được lưu.

Bảng 7: UC007 · Thiết lập lịch trình thiết bị

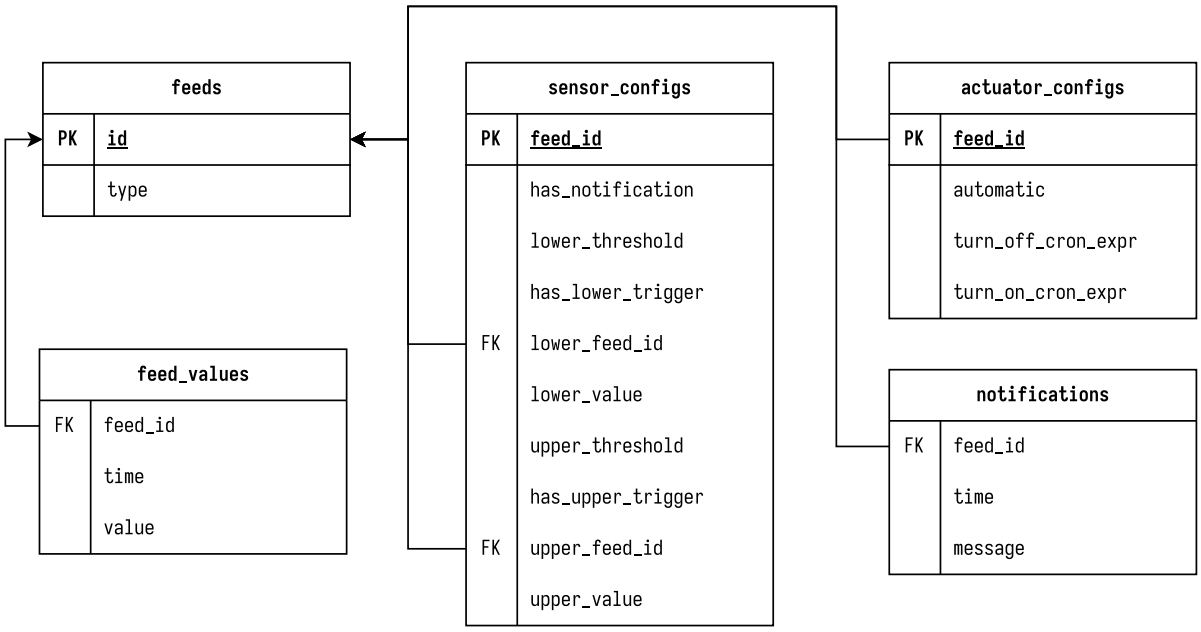
3.8 UC008 · Xem lịch sử hoạt động thiết bị

Author	Du Thành Đạt
Date	10/03/2023
Description	User muốn xem lịch sử hoạt động thiết bị.
Actor	User
Trigger	User vào màn hình Activity từ thanh điều hướng
Pre-condition	Không có
Post-condition	User xem được lịch sử hoạt động
Normal flow	1. User xem lịch sử hoạt động của thiết bị.
Alternative flow	Không có
Exception flow	• Tại bước 1, App không kết nối được với server. Lịch sử hoạt động không được hiển thị.

Bảng 8: UC008 · Xem lịch sử hoạt động thiết bị

4 Thiết kế database

Nhóm thiết kế database với lược đồ ở Hình 1.



Hình 1: Lược đồ schema của database

Lược đồ này gồm có các bảng:

feeds lưu các cảm biến và thiết bị được người dùng thêm vào nhà thông minh. `type` ở đây sẽ là `TEMPERATURE` hoặc `HUMIDITY` tương ứng với cảm biến nhiệt độ và độ ẩm, hoặc `LIGHT` cho thiết bị đèn.

feed_values lưu các giá trị mà cảm biến hoặc thiết bị ghi nhận được thông qua MQTT.

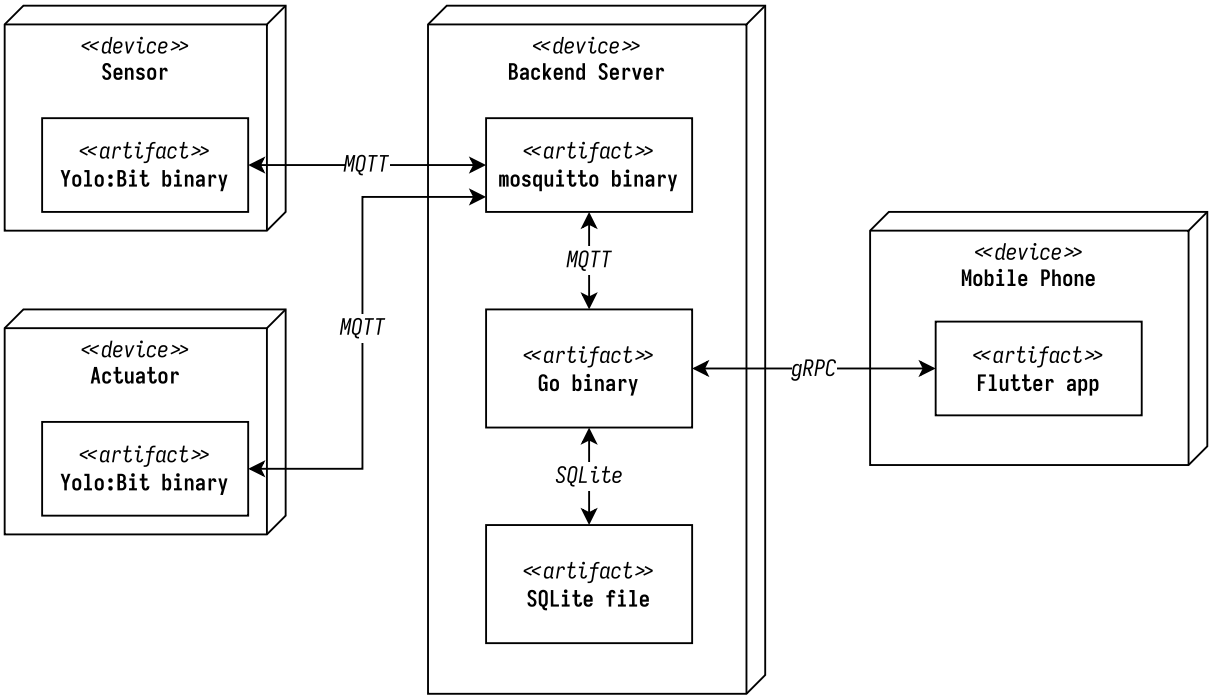
sensor_configs lưu các cấu hình cho cảm biến, bao gồm cấu hình thông báo, ngưỡng và kích hoạt thiết bị khác.

actuator_configs lưu các cấu hình cho thiết bị, bao gồm cấu hình kích hoạt tự động và lịch trình kích hoạt ở định dạng cron.

notifications lưu các thông báo được gửi đến người dùng.

5 Lược đồ deployment

Nhóm đề xuất lược đồ deployment ở Hình 2.



Hình 2: Lược đồ deployment

Lược đồ này gồm các thành phần chính:

Sensor là các cảm biến, được lập trình bởi Yolo:Bit để gửi các giá trị cảm biến đo được về cho máy chủ.

Actuator là các thiết bị, tương tự như cảm biến.

Backend Server là máy chủ, chịu trách nhiệm nhận dữ liệu và cung cấp API cho ứng dụng di động. Dữ liệu được nhận từ các cảm biến, thiết bị thông qua API MQTT, cung cấp bởi phần mềm `mosquitto`[1]. Các API được cung cấp bởi thư viện `gRPC`[2]. Các API này được sử dụng bởi ứng dụng di động để hiển thị dữ liệu.

Mobile Phone là thiết bị của người dùng. Ứng dụng di động sử dụng các API để hiển thị dữ liệu cũng như điều khiển thiết bị.

Nhóm dự kiến nếu đưa vào hoạt động, Backend Server sẽ được chạy trên một máy tính Raspberry Pi, song không có thời gian để thử nghiệm. Ngoài ra, trong đồ án này, để đảm bảo tính ổn định, các cảm biến và thiết bị sẽ gửi dữ liệu thông qua một gateway khác trước khi đến được máy chủ.

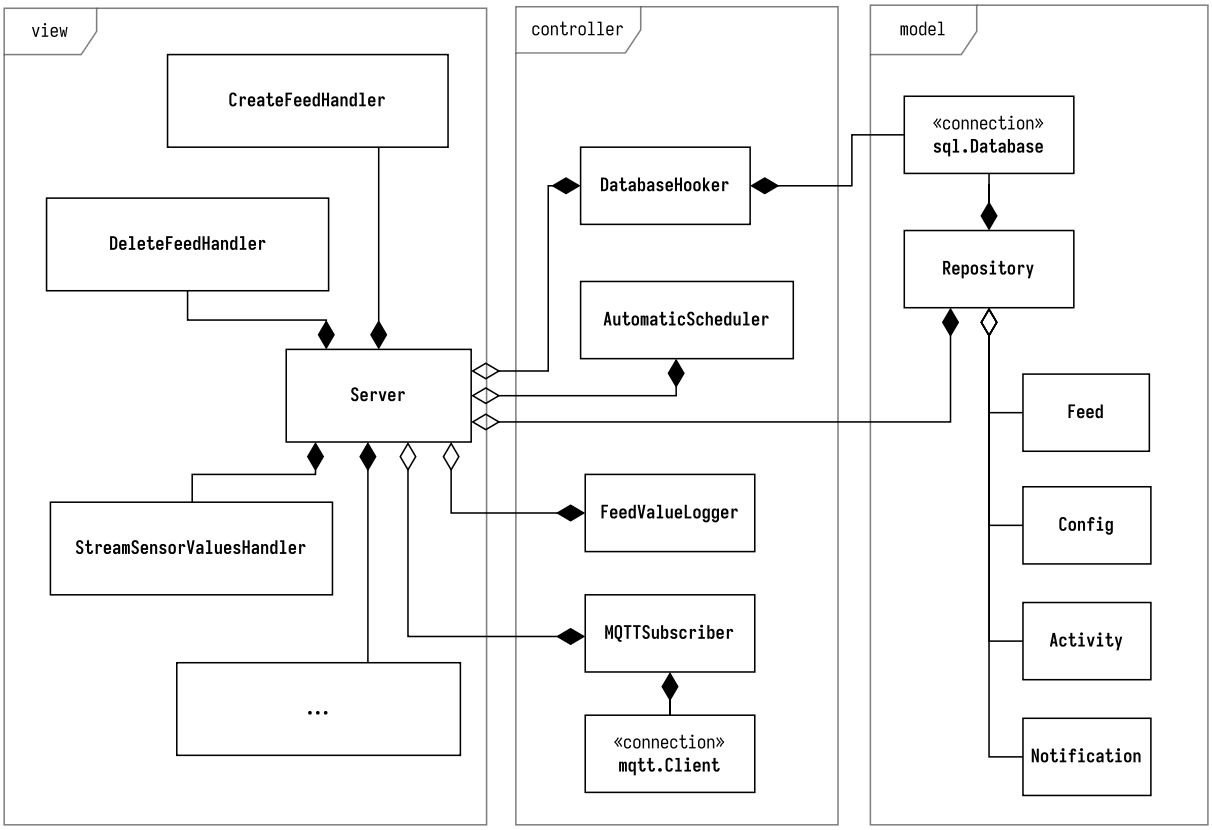
6 Thiết kế kiến trúc

Nhóm đề xuất mẫu thiết kế ở backend và frontend như sau.

6.1 Mô hình MVC ở Backend

Lược đồ các class ở backend được biểu diễn ở Hình 3.

[1]https://mosquitto.org
[2]https://grpc.io



Hình 3: Lược đồ các class ở backend

Các class này tạo thành mô hình MVC thành phần:

`VerbNounHandler` là các class chứa các method tiếp nhận các request từ client. Các method này được sinh tự động từ các file `*.proto` do gRPC. Các class đóng vai trò là view. Các class này sẽ gọi các controller khác để xử lý và trả về kết quả cho client.

`Server` là class chứa các method để khởi tạo server và điều phối các controller. Class này đóng vai trò là view.

Class này sử dụng design pattern **Mediator**, là đầu mối để tất cả các handler và controller giao tiếp với nhau.

`Repository` là class có nhiệm vụ truy xuất dữ liệu từ database. Class này đóng vai trò là model.

`Feed, Config, Activity, Notification` là các class model, được tạo bởi file `*.proto` do gRPC sinh ra.

`DatabaseHooker` là class controller, có nhiệm vụ lắng nghe các thay đổi ngay từ database và cập nhật cho những class cần.

Class này sử dụng design pattern **Observer**, để gửi các cập nhật cho các class cần thiết khác.

`AutomaticScheduler` là class controller, có nhiệm vụ chạy các task định kỳ là bật tắt thiết bị theo lịch.

`FeedValueLogger`[3] là class controller, có nhiệm vụ lắng nghe các giá trị cảm biến để sinh ra thông báo.

`MQTTSubscriber` là class controller, có nhiệm vụ lắng nghe các giá trị từ MQTT và lưu vào database, cũng như nhận các yêu cầu publish MQTT.

Class này sử dụng design pattern **Adaptor**, cung cấp API dễ sử dụng hơn class `mqtt.Client` do thư viện cung cấp.

6.2 Mô hình MVC ở frontend

Do ở frontend có *quá nhiều* class nên nhóm xin phép không thêm vào báo cáo. Thay vào đó, nhóm xin phép trình bày sơ lược hướng tiếp cận.

Nhóm sử dụng **Dart / Flutter** để xây dựng ứng dụng di động. Trong đó, nhóm sử dụng mô hình MVC để xây dựng ứng dụng, sử dụng hướng dẫn từ [1, 2, 3].

- Các class **View** sẽ là các Widget từ Flutter, có trách nhiệm nhận dữ liệu và hiển thị, gọi các callback được truyền vào khi có tương tác người dùng.
- Các class **Controller** sẽ cung cấp các callback cho **View**, và gọi các hàm từ các controller khác để thực hiện các logic.
- Các class **Model** sẽ là các class đại diện cho các đối tượng trong ứng dụng, có thể là các đối tượng từ backend, hoặc các đối tượng được tạo ra để phục vụ cho việc hiển thị. Hầu hết các class này được tạo ra bởi `gRPC`.

Tài liệu tham khảo

[1] Andrea Bizzotto. https://codewithandrea.com/

[2] "Flutter project structure: Feature-first or layer-first?." Accessed: Jun. 10, 2023. [Online]. Available: https://codewithandrea.com/articles/flutter-project-structure/

[3] "Flutter riverpod 2.0: The ultimate guide." Accessed: Jun. 10, 2023. [Online]. Available: https://codewithandrea.com/articles/flutter-state-management-riverpod/

[3] Tên class *đáng lẽ* nên là `*Notifier`, tên class hiện tại mang tính tàn tích lịch sử.