

**Phát Triển Ứng Dụng**

**Báo cáo bài tập Thực Hành**

# Thông tin sinh viên

* Họ tên: Nguyễn Văn Anh Tuấn
* Mã sinh viên: 19497581
* Link github repository: https://github.com/tuanio/19497581\_NguyenVanAnhTuan\_AD\_ToDoApp.git

# Quá trình thực hiện

1. Trình bày ý tưởng thực hiện

* Đây là ứng dụng lên danh sách những việc cần làm. Có thể chỉnh sửa, xóa, them deadline cho từng việc (gọi là task) và đánh dấu task nào đã hoàn thành.
* Em xây dựng dựa trên mẫu mà đề bài đưa cho, sử dụng Adapter trong android làm danh sách các task. Lưu trữ dữ liệu trên Firebase Realtime Database.

1. Giao diện của ứng dụng

Graphical user interface, text, application, Teams

Description automatically generatedText

Description automatically generated with medium confidence

Figure : Giao diện chính, gồm các task tương ứng và một nút tròn để hiển thị cửa sổ nhập liệu

Figure : Đây là giao diện của cửa sổ nhập liệu

1. Các lớp chức năng chính trong ứng dụng

* Các thành phần chính trong ứng dụng của em gồm: MainActivity, DialogAddItem, TaskAdapter, và 2 data class là TaskItemModel và Task Model.
* MainActivity:
  + Đây là activity chính trong ứng dụng, quản lí, tập hợp lại và hiển thị các thao tác của ứng dụng, lớp này sẽ quản lí biến realtime database, view binding, phương thức hiển thị khung nhập liệu và hàm xử lý khi người dùng nhấn sửa hoặc thêm. Các thuộc tính và phương thức trong lớp này sẽ được truyền đến cho DialogAddItem và TaskAdapter để tận dụng lại sau.
  + Hình dưới minh họa cho một số đoạn code đầu của MainActivity.

Text

Description automatically generated

* DialogAddItem:
  + Đây là lớp quản lý khung nhập liệu (thêm và sửa), ở class này, em sử dụng các code được lấy từ <https://developer.android.com/guide/topics/ui/dialogs>, để tùy biến lại cho giao diện của khung nhập liệu.
  + Ở trong lớp này, có một interface tên là DialogAddItemListener, interface này có hai nguyên mẫu hàm là onDialogPositiveClick và onDialogNegativeClick để biểu thị cho hai thao tác nhấn của user tương ứng với “save” và “cancel”. MainActivity sẽ thừa kế interface này và implement lại hai phương thức kia.
  + Dưới đây là hình minh họa cho đoạn code đầu của lớp

Graphical user interface, text

Description automatically generated

* TaskAdapter
  + Đây là lớp nhận dữ liệu từ firebase realtime database trong MainActivity và hiển thị lên màn hình các task tương ứng. Trong lớp này, các thao tác như sửa, check hoàn thành, và xóa được thực hiện trong hàm onClickItem.
  + Khi thêm, xóa sửa, để adapter được cập nhật thì em sẽ gọi các hàm như notifyItemRemoved, notifyItemRangeChanged để cập nhật lại adapter hiển thị của màn hình.
  + Dưới đây là hình code minh họa

Graphical user interface, text, application

Description automatically generated

* TaskItemModel, TaskModel
  + Hai lớp dữ liệu này mục đích chỉ để lưu cấu trúc dữ liệu để đưa vào firebase realtime database và nhận về tương ứng.
  + Dưới đây là hai hình minh họa

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Kết nối dữ liệu với firebase

* Em lưu trên firebase một Key tên là “task”, bên trong key này sẽ chứa các key nhỏ hơn, các key này chính là unique id của từng task, được lấy bằng cách nối “ngày giờ deadline của task tính theo giây” với “ngày giờ hiện tại tính theo giây” (nếu người dùng không chỉ định, mặc định ngày giờ deadline sẽ là “31/12/2030 - 23:59”), mục đích của việc này là vì realtime database sẽ tự động sắp xếp dữ liệu theo key, em làm thế để sau khi người dùng đóng app và mở lại, hệ thống sẽ tự động sắp xếp lại các task theo thời gian deadline tương ứng.
* Bên trong mỗi task, em lưu 3 trường, “dateString” chính là ngày giờ theo dạng chuỗi để hiển thị, “desc” là mô tả về task của người dùng nhập vào, và “status” là trạng thái người dùng đã tick vào hoàn thành task này hay chưa, kiểu bool.
* Dưới đây là hình minh họa cây dữ liệu có 3 task

Graphical user interface, text, application, email

Description automatically generated

1. Hiển thị kết quả chạy chương trình

* Các chức năng chính của ứng dụng bao gồm:
  + Thêm một task, trong một task có thể có
    - Thông tin mô tả
    - Ngày đến hạn (deadline), nếu không chỉnh thì giá trị mặc định sẽ là “31/12/2030 - 23:59”
  + Khi click vào một task đã tạo, có thể lựa chọn giữa:
    - Chỉnh sửa
      * Sửa thông tin mô tả
      * Sửa ngày đến hạn
    - Xóa
    - Cập nhật trạng thái của task hoàn thành hay chưa
  + Tự động sắp xếp các task theo ngày đến hạn
    - Muốn thấy được kết quả sắp xếp thì phải tắt ứng dụng đi rồi mở lại. Điều này cũng giống với cách sử dụng ứng dụng thường ngày của mọi người. Khi thêm task vào ứng dụng sẽ không cập nhật ngay. Vào lần tiếp theo người dùng mở ứng dụng lên thì các task đã được sắp xếp.
* Dưới đây là các hình ảnh minh họa:

|  |  |
| --- | --- |
| Figure 3 Khi vừa mở ứng dụng lên và đã có sẵn vài task | Figure 4 Khi đã nhấn vào một task |
| Figure 5 Khi nhấn vào thêm task mới | Figure 6 Chọn ngày |
| Figure 7 Chọn giờ | Figure 8 Nhập thông tin |
| Figure 9 Sau khi đã thêm task, task sẽ luôn nằm dưới cùng | Figure 10 Nhấn nút sửa |
| Figure 11 Sửa thông tin bao gồm mô tả và deadline | Figure 12 Sau khi thoát ứng dụng và mở lại, task "mua hoa mai" đã được sắp xếp đúng theo thứ tự |
| Figure 13 Xóa task | Figure 14 Check vào task để cập nhật thay đổi |

# Kết quả thực hiện

1. Làm được

* Thiết kế giao diện
* Thực hiện chức năng thêm, xóa, cập nhật
* Kết nối được với firebase
* Thêm ngày đến hạn

1. Chưa làm được

* Không có

1. Khó khăn gặp phải trong quá trình làm

* Kiến thức để thực hiện rộng, liên quan đến việc liên kết các thuộc tính và các phương thức giữa các lớp với nhau.
* Ý tưởng được đề cập của đề hơi mơ hồ