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

BÁO CÁO BÀI TẬP LỚN MÔN HỌC CÔNG NGHỆ PHẦN MỀM LỚP L01

Đề tài: URBAN WASTE COLLECTION AID UWC 2.0

GVHD: thầy Phan Trung Hiếu
thầy Bùi Công Tuấn

SV thực hiện:

Huỳnh Văn Phúc	-	2014160
Trần Vĩnh Phúc	-	2014185
Nguyễn Hồng Phát	-	2014082
Hồ Trọng Phúc	-	2014159
Nguyễn Anh Tuấn	-	2014946
Tô Đại Thịnh	-	2012118
Phan Trần Minh Đạt	-	2111025

TP. Hồ Chí Minh, Tháng 2/2023

Mục lục

1 Architecture design **3**

1.1 Draw 01 architectural diagram for the overall design of UWC 2.0 system. Write 01 paragraph for your Presentation strategy, 01 paragraph for Data storage approach and 01 paragraph for API management. 3

1.2 Draw a component diagram for the Task Assignment module 6

2 TÀI LIỆU THAM KHẢO **8**

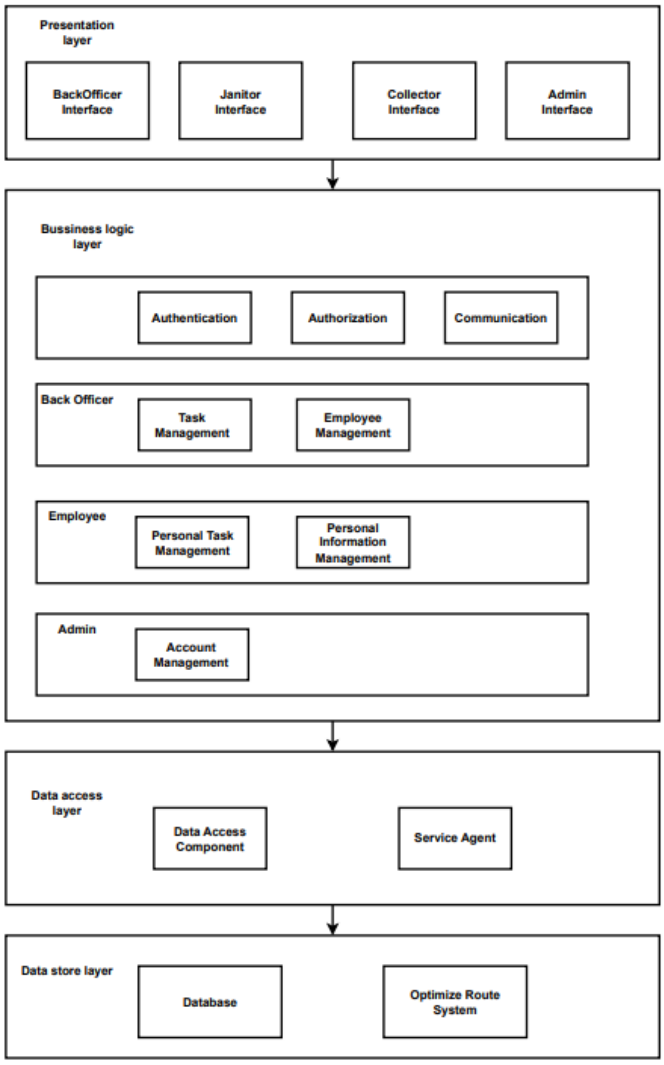
Danh sách thành viên và kết quả thực hiện

STT	Họ và tên	MSSV	Nhiệm vụ	% công việc
1	Huỳnh Văn Phúc	2014160	Task 1: 1.1, Task 2: 2.2	100%
2	Trần Vĩnh Phúc	2014185	Task 1: 1.2, Task 2: 2.4	100%
3	Nguyễn Hồng Phát	2014082	Task 1: 1.2, Task 2: 2.2	100%
4	Hồ Trọng Phúc	2014159	Task 1: 1.2, Task 2: 2.1	100%
5	Nguyễn Anh Tuấn	2014946	Task 1: 1.2, Task 2: 2.1	100%
6	Tô Đại Thịnh	2012118	Task 1: 1.3, Task 2: 2.4	100%
7	Phan Trần Minh Đạt	2111025	Task 1: 1.3, Task 2: 2.3	100%

1 Architecture design

1.1 Draw 01 architectural diagram for the overall design of UWC 2.0 system. Write 01 paragraph for your Presentation strategy, 01 paragraph for Data storage approach and 01 paragraph for API management.

1.1.1 Draw 01 architectural diagram for the overall design of UWC 2.0 system



Hình 1: Architectural Diagram

1.1.2 Layered Architecture

Sơ đồ Layer Architecture gồm 4 lớp: Presentation layer, Bussiness logic layer, Data access layer, Data store layer.

+ Lớp Presentation Layer chịu trách nhiệm hiển thị giao diện tương tác với người dùng và chịu trách nhiệm tương tác với lớp Bussiness logic layer.

+ Lớp Bussiness logic Layer sẽ nhận yêu cầu từ người dùng và thực hiện các thao tác vụ xử lí logic nghiệp vụ của ứng dụng.

+ Lớp Data access layer sẽ tương tác với hệ thống lưu trữ dữ liệu được lưu trữ trong lớp Data store layer và trả về các thông tin phù hợp.

+ Lớp Data store layer là lớp lưu trữ các thông tin cần thiết chẳng hạn như ghi nhật ký, bảo mật, giao thức truyền thông...

1.1.3 Write 01 paragraph for your Presentation strategy

Lớp Presentation Layer chịu trách nhiệm hiển thị giao diện tương tác người dùng và giao tiếp trực tiếp với lớp Bussiness logic layer. Trong lớp presentation, giao diện người dùng được chia thành 5 thành phần bao gồm các giao diện dùng chung tùy thuộc vào vai trò và chức năng của người dùng, và phần riêng cho Back Officer, Janitor, Collector và Admin.

- Giao diện cho Back Officer bao gồm các trang phục vụ cho chức năng quản lý nhân viên, quản lý task...
- Giao diện dành cho Janitor và Collector như trang quản lý thông tin cá nhân, quản lý task cá nhân, check-in check-out...
- Giao diện cho Admin như trang danh sách tài khoản, tạo mới, sửa xóa.
- Ngoài ra còn có 1 số giao diện dùng chung như đăng nhập, đăng xuất, chat.

1.1.4 Write 01 paragraph for your Data storage approach

Trong kiến trúc phân lớp, việc lưu trữ dữ liệu được thực hiện thông qua một lớp Data Layer riêng biệt và cho phép các lớp khác truy cập vào dữ liệu thông qua lớp này. Việc lớp Bussiness logic layer giao tiếp với lớp Data Layer để truy xuất và thao tác với dữ liệu chứ không có cơ chế lưu trữ dữ liệu vì trong kiến trúc phân lớp thì các lớp phải tách riêng biệt nhau. Cơ chế lưu trữ dữ liệu được chọn phải dựa trên nhiều yếu tố như khả năng mở rộng, hiệu suất,... Trong bài tập lớn lần này nhóm chúng em sẽ thực hiện việc lưu trữ dữ liệu qua các bước sau:

- Xác định các lớp: Bước đầu tiên là xác định các lớp của kiến trúc. Thông thường, kiến trúc phân lớp có lớp trình bày, lớp logic nghiệp vụ và lớp truy cập dữ liệu.

- Chọn một hệ thống quản lý cơ sở dữ liệu: Khi các lớp được xác định, bước tiếp theo là

chọn một hệ thống quản lý cơ sở dữ liệu để lưu trữ dữ liệu. Có nhiều tùy chọn có sẵn, bao gồm cơ sở dữ liệu quan hệ, cơ sở dữ liệu NoSQL và cơ sở dữ liệu hướng đối tượng.

- Xác định lược đồ dữ liệu: Bước tiếp theo là xác định lược đồ dữ liệu cho từng lớp. Lược đồ dữ liệu xác định cấu trúc của dữ liệu sẽ được lưu trữ trong cơ sở dữ liệu. Bước này rất quan trọng vì nó đảm bảo rằng dữ liệu nhất quán trên tất cả các lớp của kiến trúc.

- Triển khai tầng truy cập dữ liệu: Tầng truy cập dữ liệu chịu trách nhiệm tương tác với hệ quản trị cơ sở dữ liệu. Lớp này cung cấp các phương thức để đọc, ghi, cập nhật và xóa dữ liệu. Nó cũng đảm bảo rằng dữ liệu được truy xuất và lưu trữ ở định dạng được chỉ định trong lược đồ dữ liệu.

- Triển khai lớp logic nghiệp vụ: Lớp logic nghiệp vụ chịu trách nhiệm xử lý dữ liệu và đưa ra quyết định dựa trên dữ liệu đó. Lớp này tương tác với lớp truy cập dữ liệu để lấy dữ liệu cần thiết và sau đó thực hiện xử lý theo yêu cầu.

- Triển khai lớp trình bày: Lớp trình bày chịu trách nhiệm hiển thị dữ liệu cho người dùng. Lớp này tương tác với lớp logic nghiệp vụ để lấy dữ liệu cần thiết rồi trình bày cho người dùng ở định dạng mong muốn.

- Kiểm tra hệ thống: Sau khi triển khai xong, hệ thống phải được kiểm tra kỹ lưỡng để đảm bảo rằng hệ thống hoạt động chính xác. Điều này bao gồm kiểm tra độc lập từng lớp của kiến trúc và sau đó kiểm tra toàn bộ hệ thống.

1.1.5 Write 01 paragraph for your API management

Trong kiến trúc phân lớp thì việc truy cập các liên kết bên ngoài và API phải thông và một lớp dịch vụ. Lớp này giúp cho việc trừu tượng hóa cũng như cung cấp giao diện để lớp Bussiness Logic thực hiện tương tác. Trong bài tập lớn này, nhóm thực hiện quản lí API qua các bước:

- Tạo API: Bước đầu tiên trong quản lý API là tạo API. Điều này liên quan đến việc xác định các điểm cuối, phương thức, tham số và cấu trúc dữ liệu của API.

- Xuất API: Sau khi API được tạo, nó cần được xuất bản để các nhà phát triển có thể sử dụng. Xuất bản API liên quan đến việc làm cho API có thể truy cập được qua internet và cung cấp tài liệu cũng như các tài nguyên khác để giúp các nhà phát triển hiểu cách sử dụng nó.

- Bảo mật API: Bảo mật API là một khía cạnh quan trọng trong quản lý API. Nó liên quan đến việc triển khai các biện pháp bảo mật như xác thực và ủy quyền để đảm bảo rằng chỉ những người dùng được ủy quyền mới có thể truy cập API.

- Khả năng mở rộng API: API cần được thiết kế theo cách có thể xử lý khối lượng lớn lưu lượng truy cập và tăng hoặc giảm quy mô khi cần. Quản lý API liên quan đến việc triển khai các cơ chế mở rộng quy mô như cân bằng tải và lưu vào bộ nhớ đệm để đảm bảo rằng các API có thể xử lý khối lượng lớn yêu cầu.

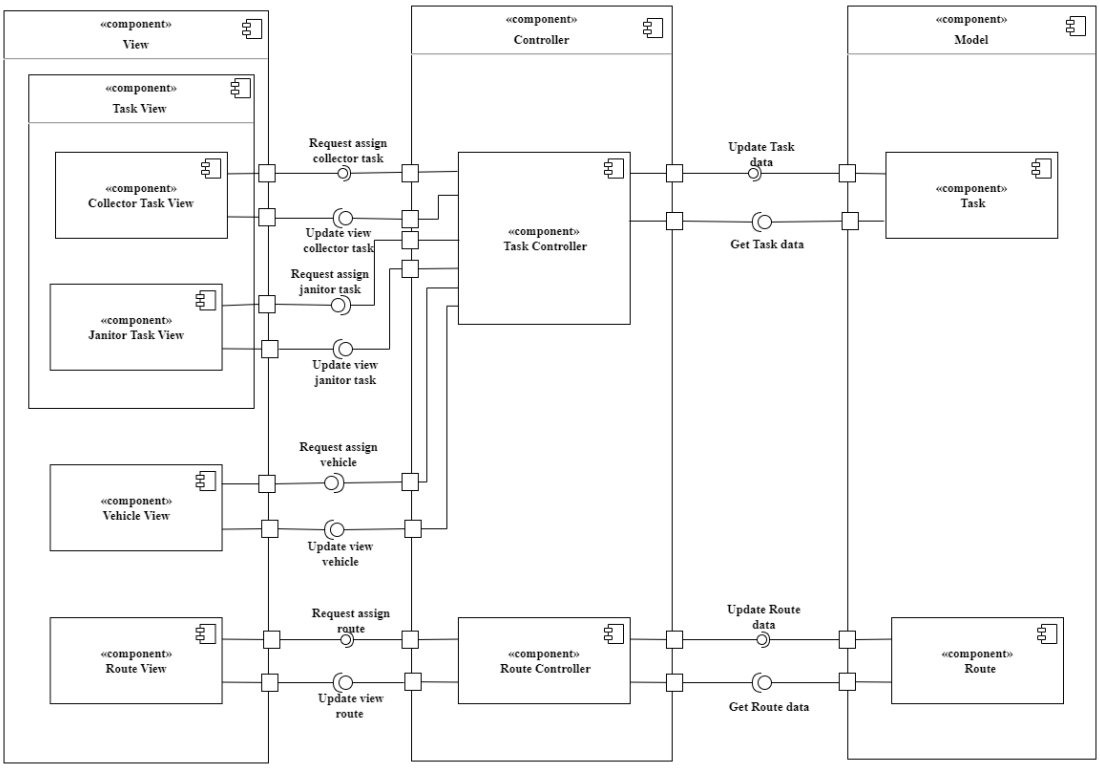
- Giám sát và phân tích API: Quản lý API liên quan đến việc giám sát các API để đảm

bảo rằng chúng đang hoạt động tốt và nhanh chóng xác định các vấn đề. Analytics giúp người quản lý API xác định các kiểu sử dụng, hiểu hành vi của người dùng và tối ưu hóa hiệu suất của API.

- Phiên bản API: Khi các API phát triển theo thời gian, điều cần thiết là duy trì khả năng tương thích ngược và hỗ trợ nhiều phiên bản API. Quản lý API liên quan đến việc lập phiên bản API và cung cấp các công cụ cũng như tài nguyên để giúp các nhà phát triển chuyển sang các phiên bản mới.

->Nhìn chung, quản lý API là rất quan trọng để đảm bảo rằng API an toàn, có thể mở rộng và hiệu quả, đồng thời đáp ứng nhu cầu của cả nhà phát triển và người dùng cuối.

1.2 Draw a component diagram for the Task Assignment module



Hình 2: Component Diagram for the Task Assignment module

Component Diagram cho Task Assignment Module được nhóm triển khai theo mô hình MVC, được chia thành 3 «component» lớn là View, Controller và Model, dưới đây là mô tả chi tiết về Component Diagram của nhóm:

- Với «component» "collector task view", BO có thể chọn thông tin cho một task mới bao

gồm thời gian, tuyến đường, xe và collector. Các thông tin sẽ được gửi đến «component» "task controller"khi nhận được «interface» "request assign collector task".

- Với «component» "janitor task view", BO có thể chọn thông tin cho một task mới bao gồm thời gian, tuyến đường, xe và janitor. Các thông tin sẽ được gửi đến «component» "task controller khi nhận được «interface» "request assign janitor task".
- Với «component» "vehicle view", BO có thể chọn thông tin Vehicle cho collector. Các thông tin sẽ được gửi đến «component» "task controller"khi nhận được «interface» "request assign vehicle".
- Với «component» "route view", BO có thể chọn thông tin Route cho Collector hoặc Janitor. Các thông tin sẽ được gửi đến «component» "route controller"khi nhận được «interface» "request assign route".
- Với «component» "task controller", xử lý các thông tin nhận được từ «component» "View"để tạo task mới hoặc lấy thông tin các task có sẵn từ «component» "Model". Dữ liệu task mới sẽ được gửi đến «component» "Task"khi nhận được «interface» "Update Task data"; khi nhận được «interface» "Update view collector task", gửi danh sách cập nhật collector task tới «component» "Collector task view"; khi nhận được «interface» "Update view janitor task", gửi danh sách cập nhật janitor task tới «component» "Janitor task view"; khi nhận được «interface» "Update view collector task", gửi danh sách cập nhật vehicle tới «component» "Vehicle view".
- Với «component» "route controller", xử lý thông tin về Route nhận được, tối ưu hóa Route cho Collector hoặc Janitor. Dữ liệu về Route tối ưu được gửi tới «component» "Route"khi nhận được «interface» "Update Route data"; gửi danh sách Route đã được cập nhật tới «component» "Route View"khi nhận được «interface» "Update view Route".
- Với «component» "Task", lưu trữ các thông tin liên quan đến Task. Khi nhận được «interface» "Get Task data", gửi thông tin các Task đến «Component» "Task controller".
- Với «component» "Route", lưu trữ các thông tin liên quan đến Route. Khi nhận được «interface» "Get Route data", gửi thông tin các Route đến «Component» "Route controller".

2 TÀI LIỆU THAM KHẢO

Tài liệu