ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO THỰC TẬP

NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI: Phát triển ứng dụng quản trị tài chính bằng Golang và MongoDB

Cán bộ hướng dẫn: Tạ Việt Cường

Giảng viên đánh giá: TS. Tạ Việt Cường
Sinh viên: Bùi Thị Phương Thảo
Mã sinh viên: 19020445
Lớp: K64-CQ-C-F

Hà Nội, tháng 9 năm 2022

MỤC LỤC

LỜI CẢM ƠN

Trong thời gian thực tập tại công ty cổ phần VTI, nhờ có sự hướng dẫn, giúp đỡ tận tình từ giảng viên hướng dẫn và các anh/chị cố vấn trong công ty đã giúp em có một kỳ thực tập ý nghĩa.

Em xin chân thành cảm ơn trường Đại học Công Nghệ - Đại học Quốc gia Hà Nội và công ty cổ phần VTI đã tạo điều kiện cho em hoàn thành tốt đợt thực tập chuyên ngành. Nhờ đợt thực tập này đã giúp cho em rất nhiều kinh nghiệm và kiến thức quý báu. Những kinh nghiệm đó sẽ giúp em hoàn thiện hơn trong học tập, công việc và môi trường làm việc sau này.

Em cũng xin cảm ơn giảng viên Tạ Việt Cường đã hướng dẫn, hỗ trợ và góp ý cho em trong việc hoàn thành khóa thực tập chuyên ngành.

Em xin gửi lời cảm ơn đến Ban Lãnh đạo, Ban Đào tạo của công ty, cán bộ hướng dẫn Trần Tuấn An và nhóm thực tập đã giúp đỡ em trong quá trình tiếp cận những công nghệ mới cũng như trong quá trình xây dựng và hoàn thiện sản phẩm, giúp em được trải nghiệm trong môi trường làm việc chuyên nghiệp, năng động.

Nhờ có những sự giúp đỡ này, em đã hoàn thành được dự án thực tập chuyên ngành và học hỏi được thêm nhiều kiến thức cũng như các kỹ năng trong quá trình học và làm việc tại công ty.

Sau đây là báo cáo thực tập chuyên ngành của em. Nếu có những thiếu sót, rất mong nhận được sự thông cảm từ quý thầy cô và công ty để giúp em hoàn thành tốt bộ môn thực tập chuyên ngành.

Một lần nữa em xin chân thành cảm ơn!

I. Giới thiệu chung

1. Giới thiệu công ty

- Công ty cổ phần VTI là công ty công nghệ Việt Nam trẻ với sứ mệnh mang công nghệ Việt ra Thế giới. Đặc biệt, thị trường chính của chúng tôi là Nhật Bản.

- Được thành lập từ năm 2017, VTI là công ty phần mềm chuyên cung cấp các dịch vụ cho các đối tác lớn của Nhật Bản, Hàn quốc, Việt Nam,… trong lĩnh vực chứng khoán, tài chính, bảo hiểm và sản xuất lớn như: Hitachi, Toyota, SamSung,LG, Vingroup…, kết hợp với sự đa dạng về công nghệ mới như AI, ML, hay gần đây là Cloud Computing, Blockchain,… đã giúp những sản phẩm của VTI trong thực tiễn phù hợp với công nghệ số 4.0 và ứng dụng vào thực tiễn trong các nhà máy, khu công nghiệp, trường học để đáp ứng được nhu cầu của đối tác và người sử dụng. Minh chứng là VTI đang phát triển những sản phẩm do chính các kỹ sư và lập trình viên nhà VTI nghiên cứu để ứng dụng rộng rãi trong công việc và đời sống như: thiết bị nhận điện khuôn mặt, thiết bị check-in học sinh trên xe bus tại trường học...

- Bên cạnh đó , VTI có các công ty con như VTI Cloud - chuyên cung cấp các giải pháp điện toán đám mây, VTI Academy - chuyên về đào tạo nhân sự IT.

2. Giới thiệu công việc

2.1. Giới thiệu chung

- Phát triển ứng dụng Web cho khách hàng hoặc các tool nội bộ, tổ chức theo mô hình scrum agile;

- Đội ngũ phát triển có 15 thành viên gồm quản lý dự án, BA, backend developer, frontend developer, tester;

- Công nghệ sử dụng:

+ Frontend: ReactJS;
+ Backend: Golang;
+ Database: MongoDB / PostgreSQL (MongoDB được sử dụng trong dự án này).

- Các công cụ:
  + Quản lý mã nguồn: Gitlab;
  + Quản lý task, issue: Gitlab / Redmine (Gitlab được sử dụng trong dự án này);
  + Test API: Postman.

2.2. Công việc cá nhân

- Vị trí: Backend development.
- Nhiệm vụ:
  + Tham gia xây dựng cấu trúc source code và cơ sở dữ liệu;
  + Trao đổi với BA về các yêu cầu và giải pháp;
  + Viết các API, service lấy dữ liệu từ cơ sở dữ liệu và xử lý theo yêu cầu khách hàng;
  + Fix bugs;
  + Viết báo cáo hàng ngày;
  + Tham gia họp hàng tuần, trao đổi công việc, tiến độ, đề xuất giải pháp.

3. Giới thiệu qua dự án

Dự án VFS là dự án xây dựng tool nội bộ quản lý công việc, báo cáo tài chính, báo cáo quản trị kế toán và kế hoạch kinh doanh, tài chính cho đội ngũ kế toán của công ty. Dự án có sự tham gia của bộ phận kế toán công ty với vai trò khách hàng.

II. Yêu cầu dự án

1. Tổng quan ứng dụng

1.1. Mô tả

- Xây dựng phần mềm VFS hỗ trợ quản lý, quản trị kế toán và tài chính một cách thuận tiện, minh bạch và tiết kiệm thời gian, chi phí quản lý cho đội ngũ kế toán.
- Bộ phận kế toán sẽ cung cấp dữ liệu mẫu thô (các thông số và dữ liệu giả định), công thức tính toán và cấu trúc dữ liệu sau khi tính toán (giả định) cho đội ngũ phát triển. Căn cứ vào các dữ liệu kế toán cung cấp, cần tạo được cấu trúc database, từ đó xây dựng phần mềm.

1.2. Phạm vi

- Nội bộ công ty

1.3. Đối tượng sử dụng

- Đối tượng sử dụng: Toàn bộ nhân viên của bộ phận kế toán và một số quản lý của bộ phận khác.

2. Tổng quan tính năng

2.1. Quản lý người dùng

- Người quản trị có quyền thêm sửa xóa tài khoản và phân quyền;
- Người dùng có thể đăng nhập, đăng xuất theo thông tin đã được cấp bởi người quản trị.

2.2. Báo cáo tài chính

Quản lý tài chính theo các đơn vị trực thuộc của công ty:

- Cho phép người dùng import, export dữ liệu báo cáo tài chính;

- Cho phép người dùng thêm, sửa, xóa dữ liệu tài chính. Những dữ liệu liên quan đến các dữ liệu này cũng sẽ tự động thay đổi. Các dữ liệu thay đổi sẽ được đánh dấu để đối chiếu với dữ liệu ban đầu;

- Cho phép người dùng thống kê các dữ liệu tài chính thông qua các bảng, biểu đồ. Các bảng biểu này có đính kèm phần nhận xét.

2.3. Báo cáo quản trị

Quản lý theo các bộ phận của công ty:

- Cho phép người dùng import, export dữ liệu báo cáo quản trị;

- Cho phép người dùng thêm, sửa, xóa dữ liệu báo cáo. Những dữ liệu liên quan đến các dữ liệu này cũng sẽ tự động thay đổi. Các dữ liệu thay đổi sẽ được đánh dấu để đối chiếu với dữ liệu ban đầu;

- Cho phép người dùng thống kê các dữ liệu báo cáo quản trị thông qua các bảng, biểu đồ. Các bảng biểu này có đính kèm phần nhận xét.

2.4. Quản lý kế hoạch

Quản lý kế hoạch kinh doanh theo các đơn vị trực thuộc của công ty:

- Cho phép người dùng import dữ liệu;

- Cho phép người dùng thêm, sửa, xóa dữ liệu. Những dữ liệu liên quan đến các dữ liệu này cũng sẽ tự động thay đổi. Các dữ liệu thay đổi sẽ được đánh dấu để đối chiếu với dữ liệu ban đầu;

- Cho phép người dùng quản lý dữ liệu thông qua các bảng.

3. Phân công

- Team Backend: đảm nhiệm việc thiết kế hệ thống phía Backend, thiết kế cơ sở dữ liệu, lập trình phía Backend cho ứng dụng để cung cấp các

API cho phía Frontend, kết hợp Business Analytic và Front-end để đảm bảo các yêu cầu đề ra của ứng dụng.

- Team Business Analyst: Thu thập yêu cầu từ khách hàng.
- Team Frontend: Thiết kế giao diện và tương tác người dùng của ứng dụng và hiển thị dữ liệu của hệ thống cho người dùng.
- Team DevOps: triển khai ứng dụng lên cho khách hàng.

Trong đó, em được phân công vị trí Backend phần quản lý kế hoạch, cụ thể như sau:

- Thiết kế source code;
- Thiết kế cơ sở dữ liệu theo tài liệu đặc tả của BA;
- Viết các API, service lấy dữ liệu từ cơ sở dữ liệu và xử lý theo đặc tả yêu cầu.

III. Tóm tắt lý thuyết, giải pháp, thuật toán

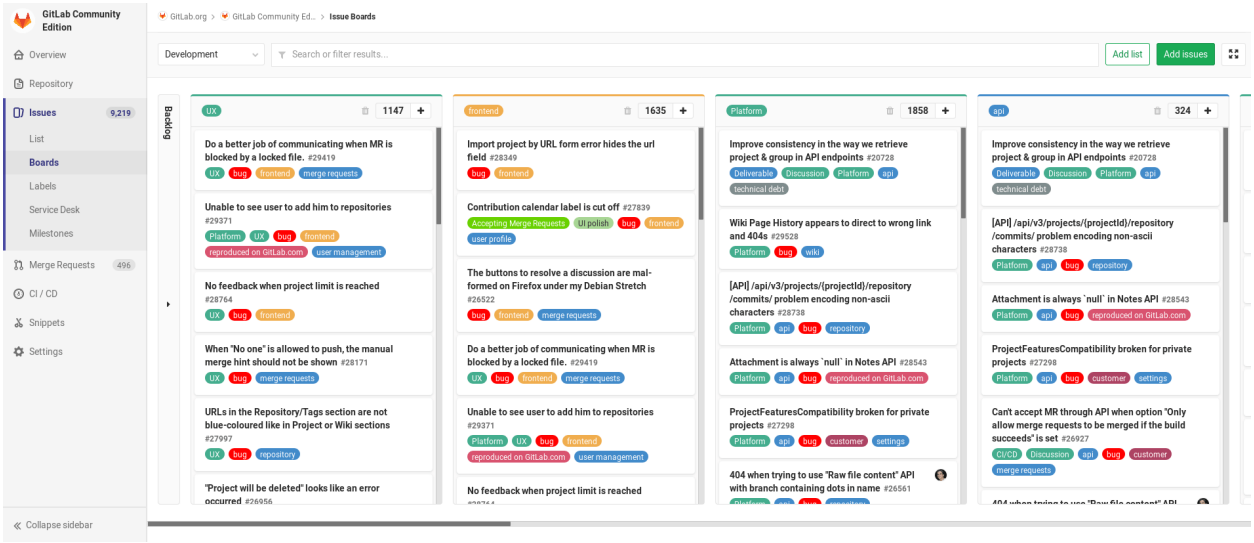
1. Mô hình Client - Server

Mô hình Client Server là mô hình mạng máy tính trong đó các máy tính con được đóng vai trò như một máy khách, chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ. Để máy chủ xử lý yêu cầu và trả kết quả về cho máy khách đó. Trong mô hình Client Server, server chấp nhận tất cả các yêu cầu hợp lệ từ mọi nơi khác nhau trên Internet, sau đó trả kết quả về máy tính đã gửi yêu cầu đó. Mô hình mạng Client Server sẽ cho phép mạng tập trung các ứng dụng có cùng chức năng tại một hoặc nhiều dịch vụ file chuyên dụng. Chúng sẽ trở thành trung tâm của hệ thống. Hệ điều hành của mô hình Client server sẽ cho phép người dùng chia sẻ đồng thời cùng một loại tài nguyên mà không giới hạn vị trí địa lý.

2. Gitlab Issue Board

Gitlab được biết đến rộng rãi, được nhiều cá nhân và tổ chức sử dụng để quản lý mã nguồn an toàn, dễ dàng và nhanh chóng. Bên cạnh khả năng quản lý mã nguồn, gitlab còn có thể được sử dụng để quản lý issue. GitLab

Issue Board là một công cụ quản lý dự án phần mềm được sử dụng để lập kế hoạch, tổ chức và trực quan hóa quy trình làm việc cho một bản phát hành tính năng hoặc sản phẩm.



Hình minh họa Gitlab Issue Board

Công cụ này cung cấp sẵn 2 cột Open và Closed. Các Issue mới hình thành sẽ được đưa vào Open, các Issue đã hoàn thành và được Product Owner nghiệm thu thì sẽ được Closed. Có thể thêm các stage trung gian giữa Open và Closed để phản ánh quy trình phát triển của nhóm phát triển. Ngoài ra, người quản lý có thể dễ dàng theo dõi tiến độ của dự án nói chung và các task của cá nhân nói riêng thông qua các dữ liệu duedate, estimate và spend time. Việc sử dụng kết hợp cả hai chức năng quản lý mã nguồn và quản lý issue sẽ giúp người quản lý cũng như các thành viên xây dựng, theo dõi và hoàn thiện dự án hiệu quả hơn.

3. Kiến trúc Clean

- **Đặt vấn đề**

Trong quá trình phát triển ứng dụng, nhóm phát triển đã đánh giá nghiệp vụ và đưa ra các vấn đề sau:

- Rất khó để thêm các tính năng mới.
- Quá khó để fix bug.

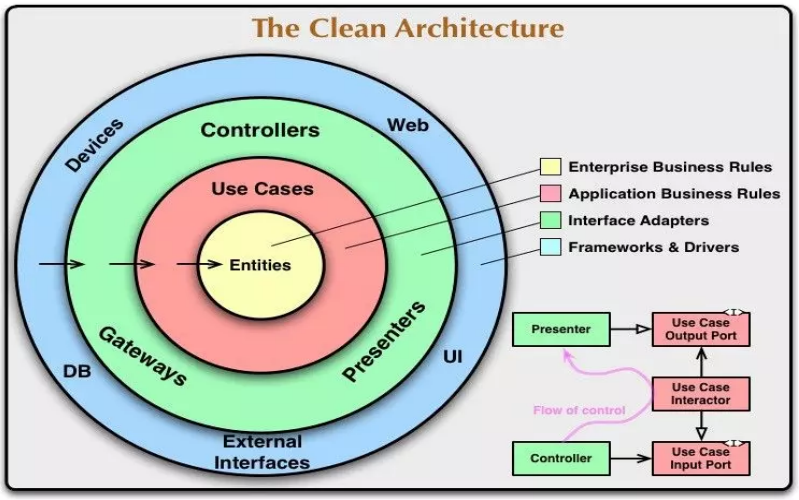
- Khó hoặc không thể test nếu không có các phụ thuộc như web server hay database.
- Có được một cái nhìn và logic nghiệp vụ trộn lẫn với nhau một cách rộng rãi, thậm chí không thể tách rời.
- Khó hiểu mục đích của nó là gì.
- Có rất nhiều công việc trong một chức năng và quá dài.

Để giải quyết những vấn đề này, nhóm phát triển đã quyết định ứng dụng kiến trúc Clean vào dự án này.

- **Tổng quan kiến trúc**

Clean Architecture được xây dựng dựa trên tư tưởng "độc lập" kết hợp với các nguyên lý thiết kế hướng đối tượng (đại diện tiêu biểu là Dependency Inversion). Độc lập ở đây nghĩa là việc project không bị phụ thuộc vào framework và các công cụ sử dụng trong quá trình kiểm thử.

Kiến trúc của Clean Architecture chia thành 4 layer với một quy tắc phụ thuộc. Các layer bên trong không nên biết bất kỳ điều gì về các layer bên ngoài. Điều này có nghĩa là nó có quan hệ phụ thuộc nên "hướng" vào bên trong.



Hình minh họa kiến trúc Clean

Trong đó,

- Entities là layer trong cùng, cũng là layer quan trọng nhất. Entity chính là các thực thể hay từng đối tượng cụ thể và các quy tắc logic nghiệp vụ của nó. Trong OOP, đây chính là Object cùng với các method và properties tuân thủ nguyên tắc Encapsulation - chỉ bên trong Object mới có thể thay đổi trạng thái của chính nó. Các logic nghiệp vụ của layer Entities sẽ không quan tâm hay lệ thuộc vào các logic nghiệp vụ ở các layer bên ngoài như Use Cases.
- Use Cases là layer chứa các logic nghiệp vụ ở cấp độ cụ thể từng Use Case. Các logic nghiệp vụ của Use Case đương nhiên cũng sẽ không quan tâm và lệ thuộc vào việc dữ liệu đến từ đâu, dùng các thư viện nào làm adapter, dữ liệu thể hiện thế nào,...
- Interface Adapters chính là layer phụ trách việc chuyển đổi các format dữ liệu để phù hợp với từng Use Case và Entities. Như vậy dữ liệu đầu vào và ra ở tầng Interface Adapter chỉ cần đủ và hợp lý. Nó sẽ không quan tâm việc dữ liệu sẽ được hiển thị cụ thể như thế nào cũng như được thu thập như thế nào.
- Framework & Drivers là tầng ngoài cùng, tổ hợp các công cụ cụ thể phục vụ cho từng nhu cầu của end user như: thiết bị (devices), web, application, databases,...

Để các layer trong Clean Architecture có thể làm việc được nhưng lại độc lập với nhau thì chúng sẽ dùng các Interfaces.

- **Ưu điểm**
  - Được tác rời khỏi bất kỳ UI, web framework hoặc DB.
  - Tập trung nhiều hơn vào logic nghiệp vụ.
  - Dễ dàng kiểm tra.
  - Có thể bảo trì, hiển thị và dễ hiểu hơn.

- **Nhược điểm**
  - Cồng kềnh và phức tạp: Điều dễ thấy nhất là Clean Architecture không hề dễ sử dụng, phải viết nhiều lớp (class/object) hơn. Trong trường hợp ứng dụng của bạn quá đơn giản, ít tính năng, vòng đời ngắn thì chọn lựa kiến trúc này có thể mang lại những rắc rối không cần thiết.

  - Tính trừu tượng cao: Vấn đề này gọi là indirect. Trừu tượng càng cao thì tiện cho các developers nhưng sẽ gây ảnh hưởng không nhỏ tới tốc độ thực thi (performance). Ngoài ra cũng không thể code nhanh, vội vã được mà phải tạo đủ các Interfaces.

  - Khó tuyển người: Sử dụng Clean Architecture sẽ cần tuyển dụng developer thấu hiểu về kiến trúc này. Nguyên tắc Dependency Inversion rất dễ bị xâm phạm vì sự hạn chế kiến thức, sự bất cẩn hoặc vì thời gian cần triển khai tính năng quá ít.

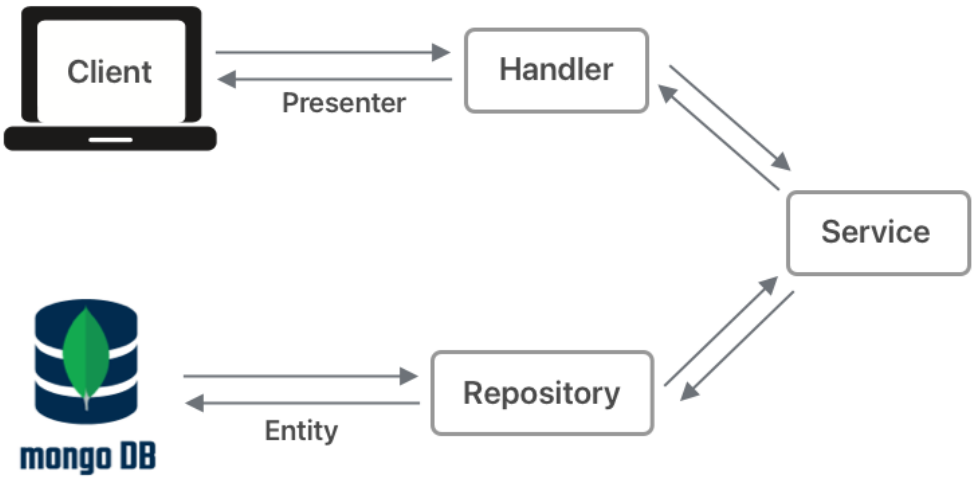
- **Ứng dụng**

Nhóm phát triển đã ứng dụng và xây dựng source code dựa trên kiến trúc này:

```
├── api
│   ├── handler
│   ├── middleware
│   └── presenter
├── config
├── entity
├── infrastructure
│   └── repository
│       ├── define
│       └── 'file chuyen doi du lieu'.go
├── main.go
├── usecase
│   ├── usecase folders
│   |   ├── interface.go
│   |   └── service.go
│   └── common.go
├── util
```

Cấu trúc source code

Mô tả luồng dữ liệu:



Minh họa luồng dữ liệu

4. Framework Iris

Iris là framework được viết bằng ngôn ngữ lập trình Golang. Tác giả iris mong muốn chọn lựa những tính năng tốt nhất, tối ưu nhất tích hợp luôn vào Iris. Do đó framework Iris gọi là full battery, full option (món đồ chơi có đầy đủ pin và phụ kiện). Iris được thiết kế và lập trình theo chủ ý của tác giả khá nhiều. Framework này hỗ trợ 8 loại view template rendering engine và cho phép sử dụng nhiều View Template Engine trong một ứng dụng. Mỗi nhóm Party có thể sử dụng View Template Engine. Đây là framework có tốc độ cao và phù hợp để phát triển ứng dụng Web và REST.

IV. Mô tả phần mềm cài đặt

Vì yêu cầu bảo mật thông tin của công ty nên mô tả phần mềm cài đặt không được công khai.

V. Kết quả đạt được, hướng phát triển

1. Kỹ năng & kiến thức thu thập được

1.1. Kỹ năng mềm

- Kỹ năng làm việc nhóm và theo dõi công việc theo quy trình Agile/Scrum;
- Kỹ năng quản lý và phân bố thời gian để xử lý các công việc được hiệu quả;
- Biết thêm về một số công cụ quản lý công việc doanh nghiệp thường sử dụng;
- Kỹ năng tự học và tự tìm hiểu;
- Kỹ năng phản biện.

1.2. Kỹ năng chuyên môn

- Trau dồi các kỹ năng về OOP, DSA, thiết kế cơ sở dữ liệu, thiết kế hệ thống phần mềm;
- Tiếp xúc và sử dụng ngôn ngữ, framework mới;
- Trau dồi và sử dụng các kỹ thuật, thuật toán xử lý;
- Trau dồi thêm kỹ năng sử dụng các công cụ quản lý phiên bản (git);
- Trau dồi thêm kỹ năng về coding convention, đảm bảo code sạch đúng tiêu chuẩn;
- Hiểu về kiến trúc Clean và ứng dụng trong dự án;
- Hiểu thêm về quy trình làm phần mềm, làm rõ các khái niệm về công nghệ phần mềm đã được học trên trường như Agile-Scrum, Sprint,..
- Hiểu thêm về một số nghiệp vụ khác…

2. Hướng phát triển tiếp theo để hoàn thiện giải pháp

- Cải thiện hiệu năng bằng cách sử dụng concurrency của Golang;

- Bổ sung unit test;
- Tối ưu truy vấn (ví dụ sử dụng BulkWrite trong mongoDB…);
- Cải thiện giao diện thêm trực quan và thân thiện với người dùng.

Tài liệu tham khảo

[1] Clean Architecture:
https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

Ý kiến đánh giá:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Hà Nội, ngày tháng năm 20 .

Người hướng dẫn

(Ký, ghi rõ họ tên & dấu công ty)

Ý kiến đánh giá:

.....

.....

.....

.....

.....

.....

.....

.....

Điểm số: ……. Điểm chữ: ……

Hà Nội, ngày tháng năm 20 .

Giảng viên đánh giá

(Ký, ghi rõ họ tên)