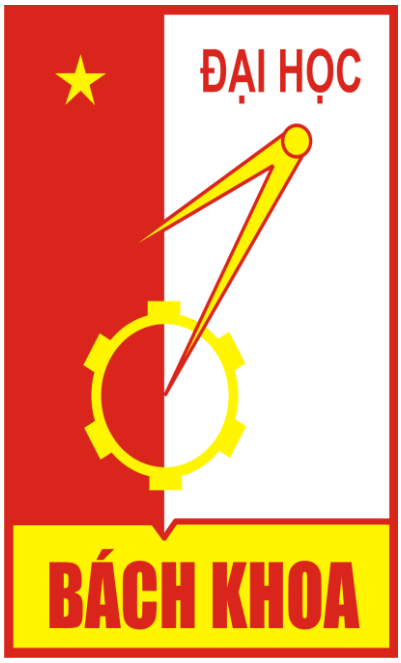
ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO CUỐI KỲ
HỌC PHẦN: NGHIÊN CỨU TỐT NGHIỆP 1
MÃ HỌC PHẦN: IT5021

Giảng viên hướng dẫn: ThS. Nguyễn Đức Tiến

Đề tài: Dự án triển khai kiến trúc multi - tenant vào phần mềm quản lý chung cư BlueMoon

Họ và tên	Mã số sinh viên	Lớp chuyên ngành
Bùi Quang Hưng	20225849	Việt Nhật 07 – K67

Hà Nội, tháng 7 năm 2025

MỤC LỤC

1 Giới thiệu chung

1.1 Giới thiệu đề tài

Trong bối cảnh chuyển đổi số ngày càng mạnh mẽ, các tổ chức và doanh nghiệp không chỉ tập trung vào số hóa quy trình, mà còn hướng đến việc tối ưu chi phí đầu tư vào hạ tầng công nghệ thông tin. Một trong những xu hướng nổi bật hiện nay là xây dựng các hệ thống phần mềm dùng chung, giúp nhiều bộ phận, chi nhánh hoặc thậm chí nhiều khách hàng bên ngoài có thể sử dụng chung một nền tảng, nhưng vẫn đảm bảo sự độc lập về dữ liệu và cấu hình.

Để đạt được điều đó, kiến trúc Multi-Tenant (đa thuê) đã trở thành một giải pháp hiệu quả và phổ biến. Mô hình này cho phép một hệ thống phần mềm duy nhất phục vụ đồng thời nhiều tenant – có thể là các phòng ban, chi nhánh, công ty con hoặc khách hàng khác nhau mà không cần triển khai nhiều bản cài đặt riêng biệt. Nhờ vậy, doanh nghiệp có thể giảm thiểu chi phí đầu tư phần cứng, chi phí triển khai, bảo trì và nâng cấp hệ thống.

1.2 Mục tiêu và phạm vi đề tài

Mục tiêu của đề tài này là tìm hiểu về kiến trúc **multi-tenant** trong ứng dụng **Accountify** và triển khai thiết kế, phát triển một hệ thống quản lý chung cư có áp dụng kiến trúc **multi-tenant** đơn giản, cho phép nhiều tòa nhà, khu dân cư hoặc ban quản lý sử dụng cùng một hệ thống nhưng với dữ liệu tách biệt, giao diện và chức năng tùy chỉnh. Hệ thống sẽ bao gồm các tính năng chính như: quản lý hộ gia đình, quản lý cư dân, quản lý việc thu phí hay quản lý xe cộ,... Đồng thời, hệ thống cần đảm bảo khả năng mở rộng, hiệu năng ổn định, và bảo mật dữ liệu giữa các tenant.

Với việc áp dụng kiến trúc **multi-tenant**, đề tài hướng tới tạo ra một nền tảng quản lý chung cư linh hoạt, tiết kiệm chi phí triển khai, dễ dàng bảo trì và có khả năng nhân rộng trên diện rộng – góp phần hiện đại hóa công tác quản lý chung cư trong kỷ nguyên số.

2 Kiến trúc Multi – tenant

2.1 Giới thiệu về Multi-Tenant

Multi-Tenant là một kiến trúc phần mềm cho phép nhiều khách hàng (tenant) sử dụng chung một ứng dụng hoặc dịch vụ trên cùng một nền tảng. Mỗi khách hàng có dữ liệu và cấu hình riêng biệt, đảm bảo tính bảo mật và riêng tư.

Trong mô hình này, một phiên bản phần mềm duy nhất và cơ sở hạ tầng của nó được chia sẻ đồng thời giữa nhiều khách hàng. Mặc dù dữ liệu của mỗi khách hàng vẫn tách biệt với dữ liệu khác, tất cả họ đều chia sẻ cùng một cơ sở dữ liệu, ứng dụng phần mềm và máy chủ SaaS.

Việc áp dụng kiến trúc Multi-Tenant mang lại nhiều lợi ích, bao gồm:

- **Tiết kiệm chi phí:** Chia sẻ tài nguyên giúp giảm chi phí phát triển, triển khai và bảo trì ứng dụng.
- **Tăng khả năng mở rộng:** Dễ dàng thêm khách hàng mới mà không cần tạo cơ sở hạ tầng riêng.
- **Nâng cao hiệu quả:** Quản lý và cập nhật ứng dụng trở nên đơn giản hơn với một phiên bản duy nhất.

2.2 Các mô hình triển khai và giải pháp thiết kế cơ sở dữ liệu cho Multi-Tenant

Trong mô hình Multi-Tenant, việc lựa chọn kiến trúc triển khai và thiết kế cơ sở dữ liệu là yếu tố quan trọng để đảm bảo hiệu suất, bảo mật và khả năng mở rộng của hệ thống. Dưới đây là các mô hình phổ biến:

- **Mô hình chia sẻ ứng dụng:** Tất cả khách hàng (tenant) sử dụng chung một phiên bản ứng dụng và dùng chung cơ sở dữ liệu. Một số phương pháp thiết kế dữ liệu đi kèm gồm:
  - **Cơ sở dữ liệu chung, chia sẻ bảng (Shared Database, Shared Table):** Tất cả tenant sử dụng chung một cơ sở dữ liệu và một bảng, nhưng dữ liệu của từng tenant được phân biệt thông qua một trường TenantID. Phương pháp này giúp tối ưu hóa chi phí, nhưng lại khó khăn trong việc bảo mật và phân tách dữ liệu.
  - **Cơ sở dữ liệu chung cho tất cả tenant (Shared Database-per-Tenant):** Tất cả tenant sử dụng chung một cơ sở dữ liệu, nhưng mỗi tenant sẽ có một bảng riêng biệt trong cơ sở dữ liệu. Phương pháp này giúp tiết kiệm tài nguyên, nhưng đòi hỏi các biện pháp bảo mật và phân quyền chặt chẽ.
- **Mô hình ứng dụng riêng biệt:** Mỗi tenant được cung cấp một phiên bản ứng dụng riêng hoặc một môi trường riêng biệt. Phù hợp với khách hàng có yêu cầu tùy chỉnh sâu hoặc bảo mật cao. Phương pháp thiết kế dữ liệu đi kèm:

- **Cơ sở dữ liệu riêng lẻ (Database-per-Tenant):** Mỗi tenant sẽ có một cơ sở dữ liệu riêng biệt. Phương pháp này giúp tối ưu hóa sự phân tách dữ liệu và bảo mật. Tuy nhiên, phương pháp này đòi hỏi chi phí quản lý cao và khó khăn trong việc mở rộng.
- **Cơ sở dữ liệu có thể mở rộng (Scalable Databases):** Sử dụng các phương pháp như sharding (phân mảnh cơ sở dữ liệu) và replication (sao chép dữ liệu) để đảm bảo hiệu suất khi số lượng tenant tăng lên, mở rộng cơ sở dữ liệu, phân tách và tối ưu hóa tài nguyên cho từng tenant.

Tùy vào quy mô hệ thống, mức độ bảo mật yêu cầu, và khả năng tài chính, doanh nghiệp có thể lựa chọn mô hình triển khai và thiết kế cơ sở dữ liệu phù hợp. Mỗi mô hình đều có ưu nhược điểm riêng và ảnh hưởng trực tiếp đến khả năng vận hành lâu dài của hệ thống Multi-Tenant.

2.3 Ưu điểm và nhược điểm của Multi-Tenant

Multi-Tenant là một mô hình triển khai công nghệ được sử dụng rộng rãi, đặc biệt trong các giải pháp phần mềm như SaaS. Tuy nhiên, mô hình này có những ưu điểm và nhược điểm cần được xem xét kỹ lưỡng.

Ưu điểm

- **Tiết kiệm chi phí:**
  Việc chia sẻ tài nguyên giúp giảm chi phí hạ tầng và vận hành, tạo điều kiện cho các doanh nghiệp nhỏ tiếp cận công nghệ hiện đại.
- **Dễ dàng mở rộng:**
  Hệ thống được thiết kế để phục vụ nhiều khách hàng đồng thời, giúp việc mở rộng quy mô trở nên đơn giản và hiệu quả hơn.
- **Cập nhật nhanh chóng:**
  Mô hình Multi-Tenant cho phép nhà cung cấp triển khai các bản cập nhật hoặc nâng cấp phần mềm trên toàn hệ thống một cách đồng bộ.
- **Quản lý tập trung:**
  Nhà cung cấp có thể quản lý và giám sát hệ thống từ một nguồn trung tâm, giảm thiểu thời gian và công sức quản lý.

Nhược điểm

- **Rủi ro bảo mật:**

Vì nhiều khách hàng chia sẻ tài nguyên, việc bảo vệ dữ liệu cá nhân và ngăn chặn truy cập trái phép là một thách thức lớn.

- **Giới hạn tùy chỉnh:**
  Một số mô hình Multi-Tenant có thể hạn chế khả năng tùy chỉnh theo nhu cầu cụ thể của từng khách hàng.
- **Ảnh hưởng hiệu suất:**
  Hiệu suất của hệ thống có thể bị ảnh hưởng nếu một khách hàng sử dụng quá nhiều tài nguyên, dẫn đến việc giảm chất lượng dịch vụ cho các khách hàng khác.
- **Phụ thuộc vào nhà cung cấp:**
  Các khách hàng phụ thuộc vào nhà cung cấp trong việc vận hành, bảo trì, và nâng cấp hệ thống.

2.4 Ứng dụng của Multi-Tenant trong thực tế

Multi-Tenant là mô hình phổ biến trong các dịch vụ điện toán đám mây, cho phép khách hàng dung một hệ thống mà vẫn đảm bảo dữ liệu tách biệt. Mô hình này được ứng dụng rộng rãi trong:

- **Phần mềm SaaS (Software as a Service):** Salesforce, Google Workspace, Microsoft 365 – giúp phục vụ nhiều khách hàng từ một hệ thống duy nhất, tiết kiệm chi phí và dễ quản lý.
- **Dịch vụ lưu trữ đám mây:** AWS, Google Cloud, Azure sử dụng Multi-Tenant để tối ưu hạ tầng và chi phí.
- **Hệ thống quản lý doanh nghiệp (ERP):** SAP, Oracle NetSuite – hỗ trợ nhiều doanh nghiệp dùng chung hệ thống mà không ảnh hưởng đến dữ liệu nhau.
- **Ứng dụng chia sẻ tài nguyên:** Dropbox, Slack sử dụng để chia sẻ không gian lưu trữ, tăng hiệu quả sử dụng.

Với ưu điểm về chi phí và khả năng mở rộng, Multi-Tenant ngày càng phổ biến trong các mô hình kinh doanh dựa trên đám mây.

Triển khai mô hình Multi-Tenant có thể mang lại nhiều lợi ích, nhưng cũng đi kèm với một số thách thức cần được giải quyết. Dưới đây là các thách thức chính khi triển khai Multi-Tenant:

- **Bảo mật dữ liệu:** Đảm bảo rằng dữ liệu của từng khách hàng (tenant) không bị lộ hoặc bị truy cập trái phép từ các tenant khác.

- **Khả năng mở rộng:** Hệ thống cần có khả năng mở rộng để duy trì hiệu suất khi số lượng tenant tăng lên và đảm bảo không có tenant nào bị ảnh hưởng bởi tài nguyên của tenant khác.
- **Quản lý hiệu suất:** Khi có nhiều tenant chia sẻ hạ tầng chung, cần phân bổ tài nguyên hợp lý và tránh tình trạng một tenant chiếm dụng quá nhiều tài nguyên có thể ảnh hưởng đến các tenant khác.
- **Khả năng tùy chỉnh:** Đáp ứng yêu cầu khác nhau về tính năng, giao diện người dùng, và cấu hình hệ thống tùy theo nhu cầu của mỗi tenant.
- **Quản lý cập nhật và bảo trì:** Việc cập nhật và bảo trì hệ thống đối với tất cả các tenant có thể phức tạp, vì không thể áp dụng thay đổi cho một tenant mà không ảnh hưởng đến các tenant khác. Điều này yêu cầu kế hoạch bảo trì cẩn thận và có sự kiểm soát chặt chẽ.

Việc triển khai Multi-Tenant đòi hỏi sự chuẩn bị kỹ lưỡng về cơ sở hạ tầng, bảo mật, và quản lý tài nguyên để đảm bảo hiệu quả và an toàn cho tất cả các tenant sử dụng dịch vụ.

3 Tìm hiểu kiến trúc Multi-tenant trong ứng dụng Accountify

3.1 Giới thiệu

Accountify – website quản lý tài chính có quy đổi ngoại hối theo kiến trúc đa khách hàng, là sản phẩm của đồ án tốt nghiệp của anh **Nguyễn Hải Dương** – **sinh viên K64 ngành công nghệ thông tin Việt Nhật, Đại học Bách Khoa Hà Nội**. **Accountify** là một phần mềm quản lý chi tiêu cho các tổ chức phi chính phủ (NGO). Dự án ra đời từ thực tế rằng nhiều tổ chức phi chính phủ (NGO) hiện nay vẫn sử dụng các phương pháp quản lý thủ công hoặc các công cụ phần mềm như Excel, Word,... không được thiết kế riêng để phù hợp với nhu cầu quản lý tài chính của họ. Điều này có thể dẫn đến việc quản lý chi tiêu thiếu hiệu quả và minh bạch, đồng thời không phải lúc nào cũng đảm bảo được việc tuân thủ các quy định pháp luật. Vấn đề quản lý chi tiêu không hiệu quả có thể gây ra nhiều hiệu quả nghiệm trọng, từ việc lãng phí nguồn lực, mất uy tín đến nguy cơ bị mất tài trợ.

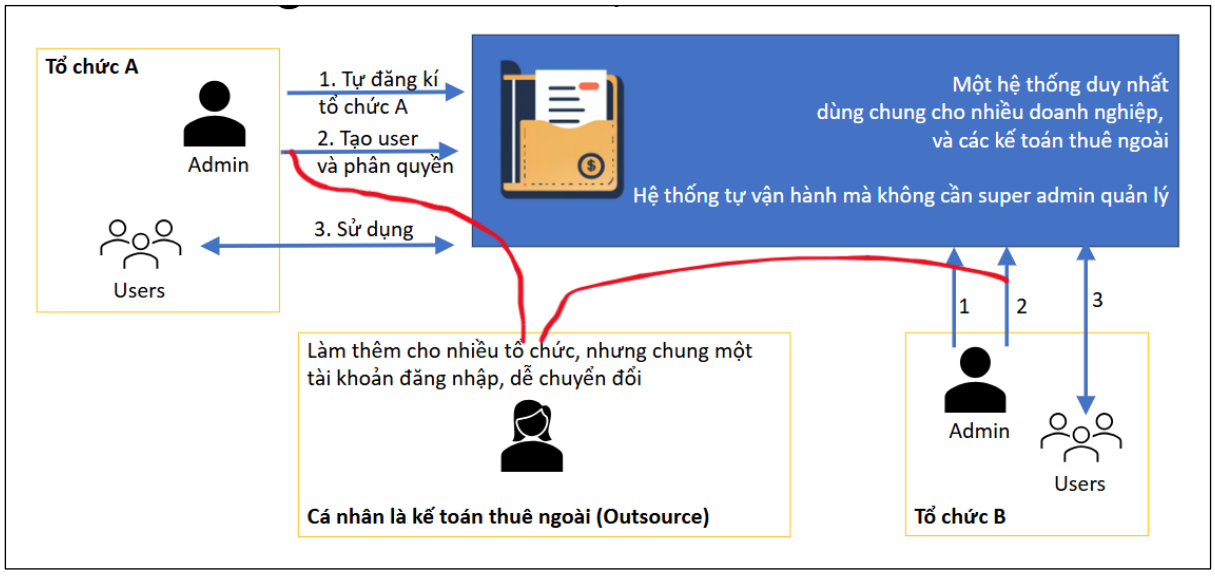
Acountify là phần mềm giúp cho các NGO khắc phục các biện pháp hiện tại bằng cách cung cấp một giải pháp toàn diện và chuyên biệt. Phần mềm này sẽ bao gồm các chức năng chính như quản lý ngân sách theo từng dự án, theo dõi

hoá đơn chi tiêu, thống kê báo cáo tài chính, hạch toán hoá đơn, kiểm tra tỷ giá ngoại tệ trong ngày và cảnh báo chi tiêu vượt mức. Điều đặc biệt là phần mềm sử dụng **kiến trúc đa khách hàng (Multi – tenant)** và hệ thống phân quyền để mỗi NGO có thể sử dụng hệ thống một cách độc lập, tùy chỉnh theo nhu cầu riêng và đảm bảo tính bảo mật dữ liệu cao.

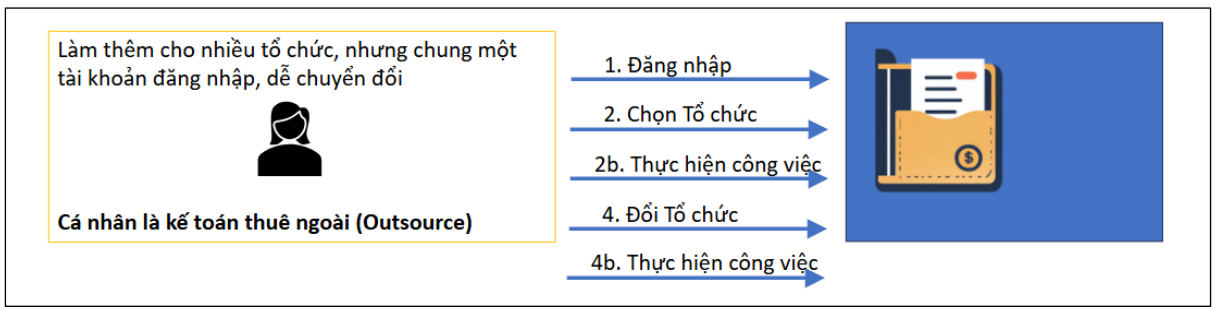
3.2 Kiến trúc Multi – tenant trong ứng dụng

Hệ thống sử dụng kiến trúc đa khách hàng cho phép mỗi NGO có một không gian làm việc riêng biệt nhưng vẫn chia sẻ cùng một cơ sở hạ tầng công nghệ.

Ngoài ra, kiến trúc đa khách hàng của ứng dụng Accountify còn đáp ứng cả trường hợp nhiều tổ chức NGO sẽ thuê nhân viên kế toàn từ bên ngoài (đối với các công ty có ít giao dịch). Ứng dụng cho phép một tài khoản cá nhân có thể thuộc nhiều tổ chức/doanh nghiệp khác nhau, họ có thể chọn tổ chức/doanh nghiệp mà người kế toán đó đang được thuê và bắt đầu làm việc. Quy trình xử lý nghiệp vụ cá nhân là kế toán thuê ngoài được thể hiện ở hai ảnh dưới.



Quy trình nghiệp vụ “thiết lập kiến trúc đa khách hàng”

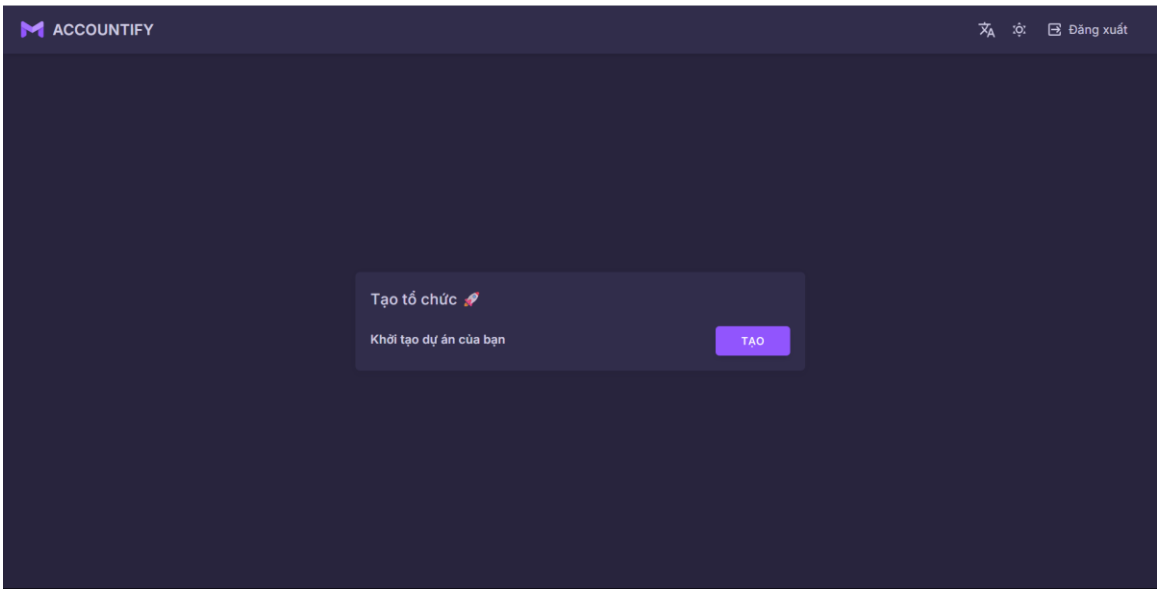


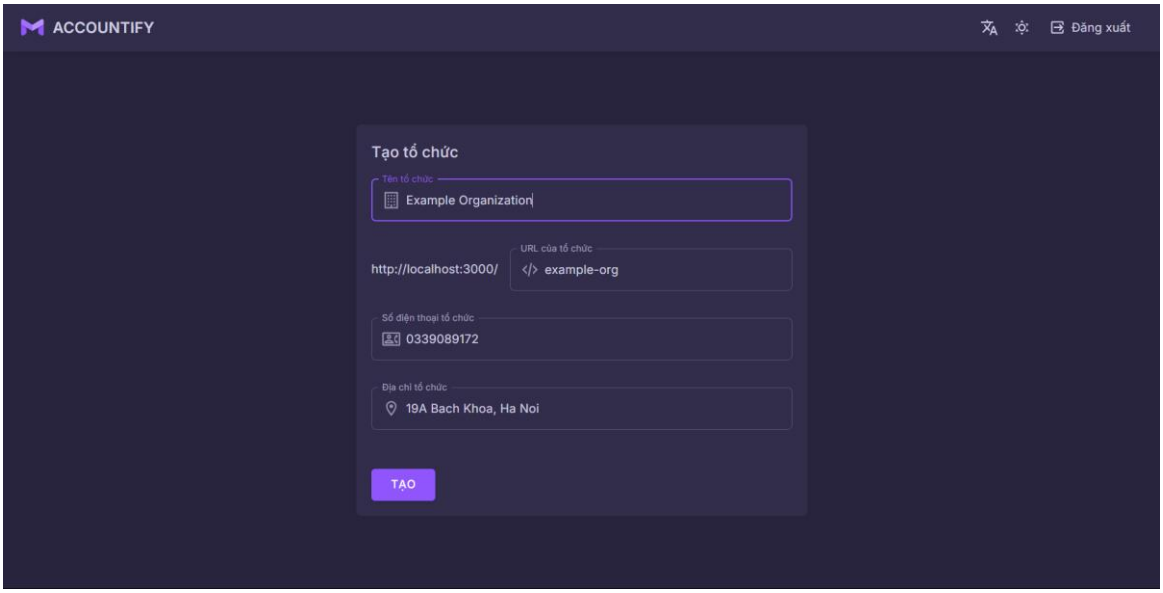
Quy trình nghiệp vụ “thuê ngoài kế toán”

3.3 Các chức năng liên quan đến kiến trúc Multi-tenant

a. Khởi tạo một **tenant**

Sau khi đăng ký tài khoản và đăng nhập vào hệ thống, nếu người dùng chưa thuộc tổ chức nào thì hệ thống sẽ yêu cầu khởi tạo một không gian làm việc dành cho tổ chức/doanh nghiệp của người dùng đó. Người khởi tạo chính là admin của tổ chức đó.

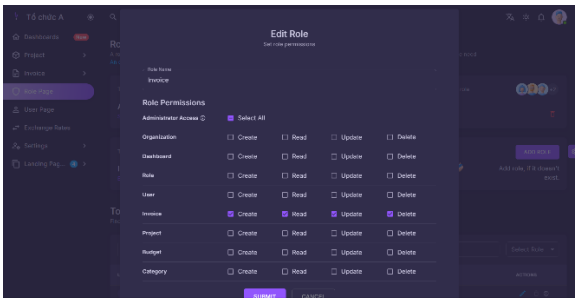




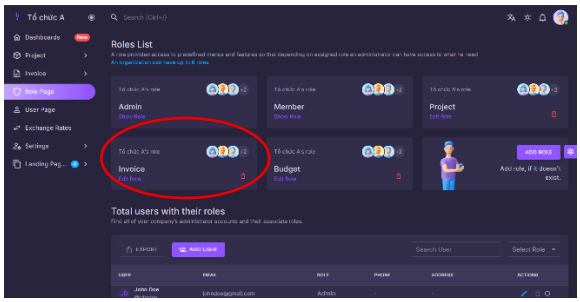
b. Chức năng phân quyền người dùng

Chức năng này đáp ứng đặc thù của các tổ chức NGO khi tổ chức đó thường có dự án với nhiều nhân viên và tình nguyện viên tham gia, mỗi người sẽ có quyền truy cập khác nhau vào các phần của hệ thống. Mỗi người sẽ chỉ được truy cập vào những tài nguyên và chức năng mà họ được cấp quyền.

Hệ thống có sẵn một bộ các đối tượng quyền hạn, quản trị viên của tổ chức (admin) sẽ là người có quyền hạn cao nhất, được phép truy cập tất cả các không gian cũng như chức năng của hệ thống. Ví dụ, quản trị viên sẽ tạo ra vai trò(role) mới dựa trên các đối tượng quyền hạn liên quan đến việc "**Quản lý hóa đơn**" với các quyền thêm/bớt/sửa/xóa hóa đơn nhưng chỉ có quyền xem đối với các phần khác.



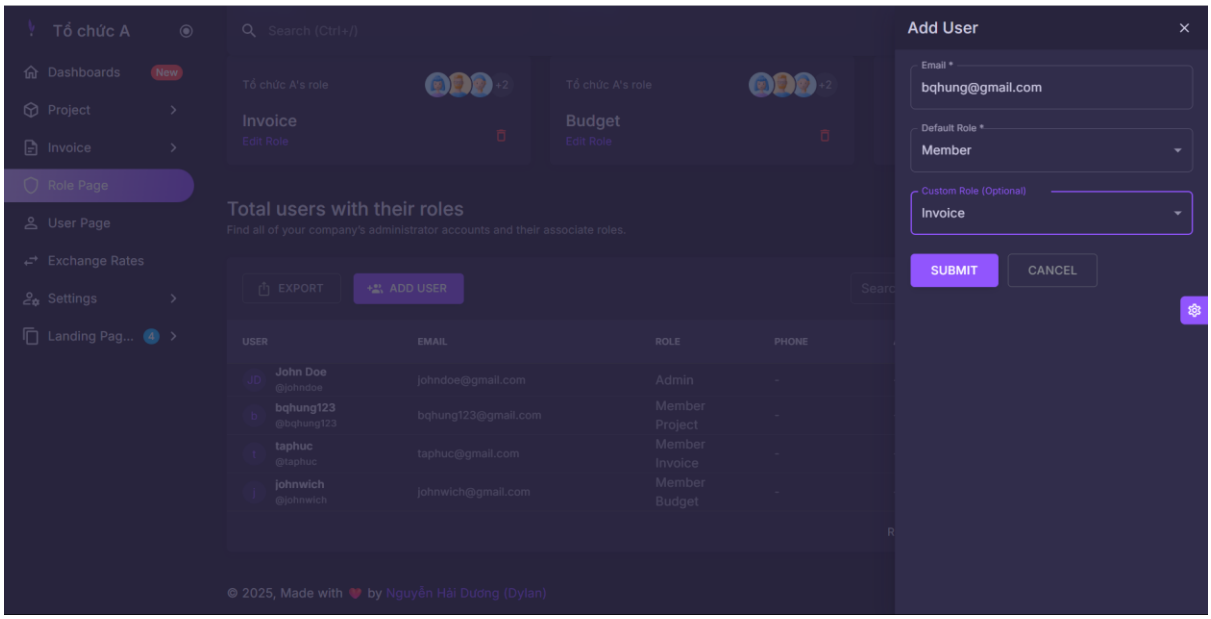
Màn hình tạo role mới

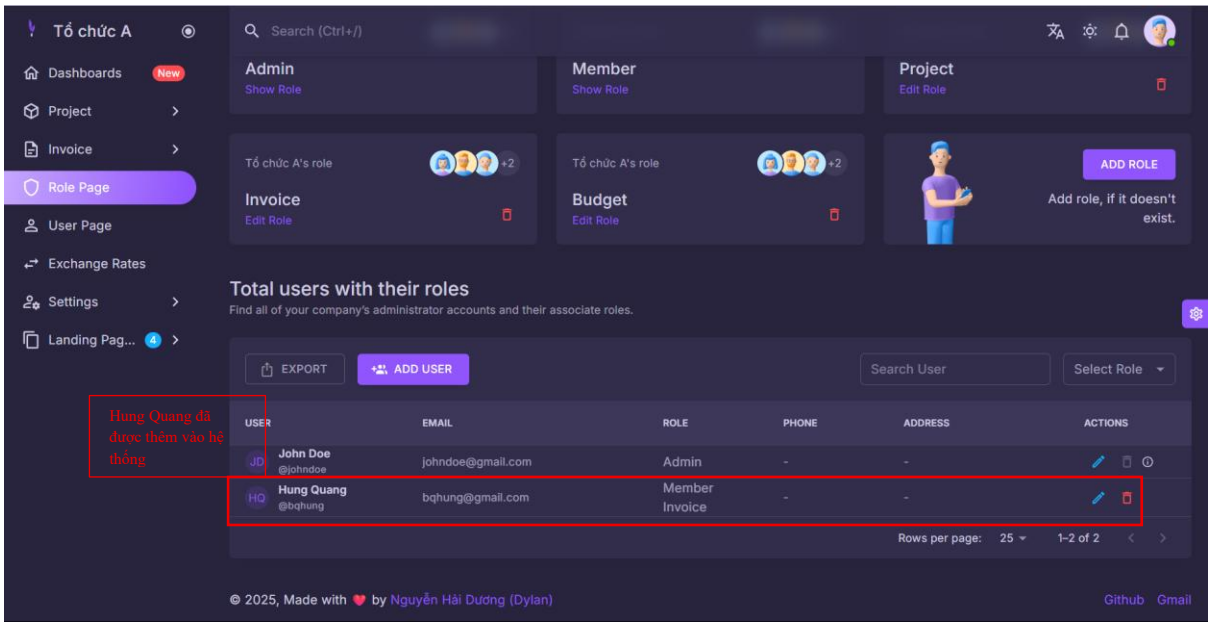


Màn hình trang Role page

Ví dụ vai trò quản lý hóa đơn

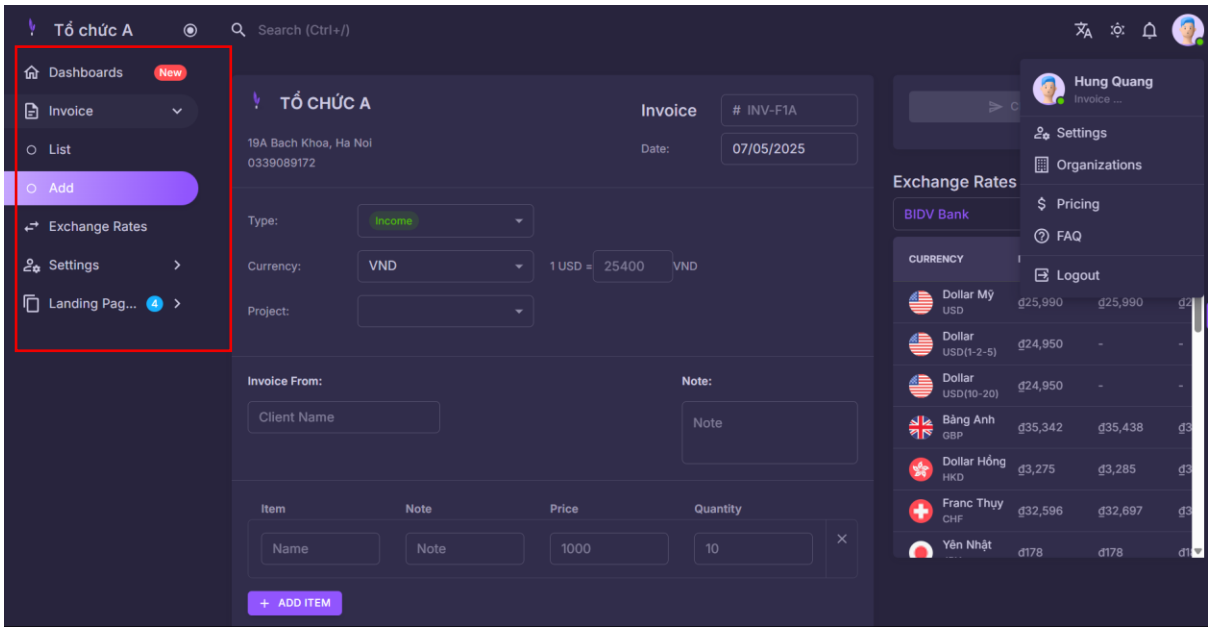
Sau khi có danh sách quyền hạn với vai trò đó, quản trị viên sẽ thêm người dùng vào hệ thống bằng cách nhập email của người đó (giả sử người dùng đó ở đây có email là **bqhung@gmail.com**) và gán cho người dùng đó vai trò liên quan đến việc "Quản lý hóa đơn".





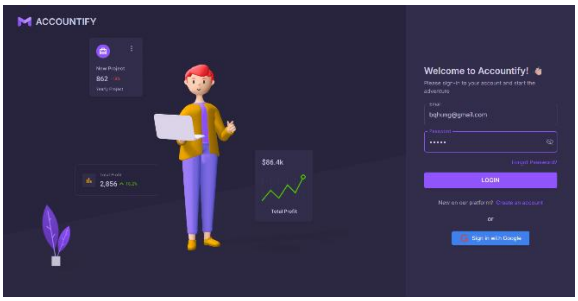
- Kết quả:

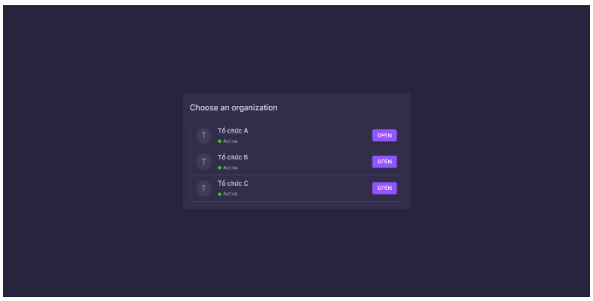
Người dùng với email **bqhung@gmail.com** truy cập vào không gian làm việc của tổ chức nhưng chỉ có quyền hạn thêm/bớt/sửa/xóa các hóa đơn nhưng giao diện không hiện các chức năng khác.



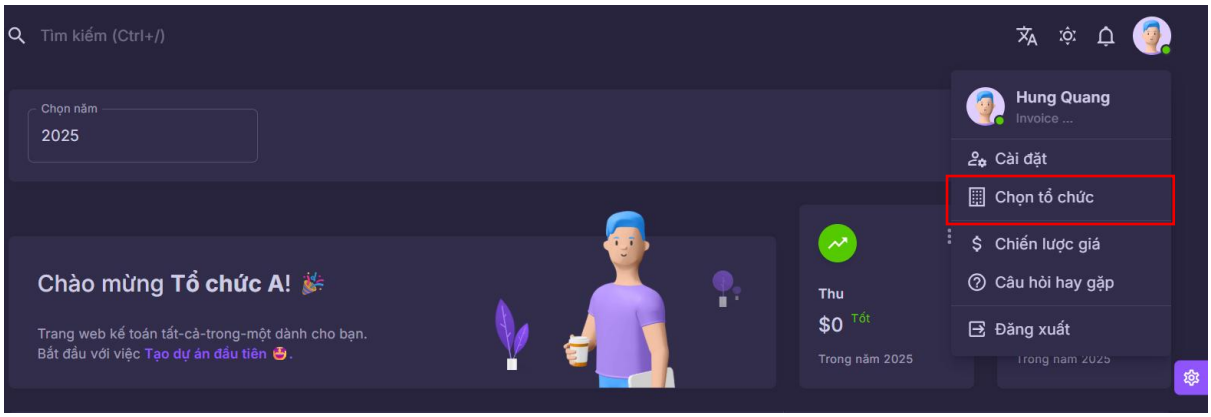
c. Chọn tổ chức/doanh nghiệp muốn làm việc

Một người dùng có thể thuộc nhiều tổ chức/doanh nghiệp khác nhau. Khi đăng nhập vào hệ thống sẽ hiển thị các tổ chức mà người dùng đó đang làm việc. Người dùng sẽ chọn tổ chức/doanh nghiệp mình muốn truy cập và bắt đầu phiên làm việc.





Hai hình ảnh trên là ví dụ về người dùng với tài khoản **bqhung@gmail.com** sau khi đăng nhập sẽ phải chọn một trong ba tổ chức để bắt đầu phiên làm việc. Người dùng đó cũng có thể chọn vào chức năng “chọn tổ chức” để có thể đổi sang tổ chức khác.



4 Triển khai kiến trúc Multi-tenant đơn giản

4.1 Bối cảnh

Hệ thống phần mềm quản lý chung cư Bluemoon đã được em và các bạn xây dựng để phục vụ yêu cầu của bài tập lớn học phần **Kỹ thuật phần mềm – IT4082.** Nhưng ban đầu phần mềm được triển khai với mô hình đơn lẻ, chỉ đáp ứng cho một chung cư duy nhất.

Để phục vụ cho đề tài của học phần Nghiên cứu đồ án tốt nghiệp 1, sau khi tìm hiểu về kiến trúc Multi-tenant, em đã áp dụng và triển khai kiến trúc này một cách đơn giản vào trong phần mềm quản lý chung cư.

4.2 Yêu cầu về chức năng

Các chức năng liên quan đến kiến trúc Multi-tenant của hệ thống

- Đăng ký tenant (chung cư mới): hệ thống sẽ cung cấp biểu mẫu đăng ký dành cho ban quản lý chung cư, sau khi đăng ký thì hệ thống sẽ khởi tạo không gian làm việc riêng dành cho chung cư đó.
- Lựa chọn chung cư để làm việc: Một người dùng có thể vừa là tổ trưởng của chung cư này vừa làm thủ quỹ của chung cư khác. Khi người dùng đó đăng nhập, hệ thống sẽ yêu cầu chọn chung cư muốn làm việc.
- Thêm một tài khoản người dùng (có sẵn trong hệ thống) vào chung cư: Tổ trưởng (admin) của chung cư có thể thêm người dùng khác vào hệ thống với vai trò tổ phó hoặc thủ quỹ để đáp ứng nhu cầu quản lý.

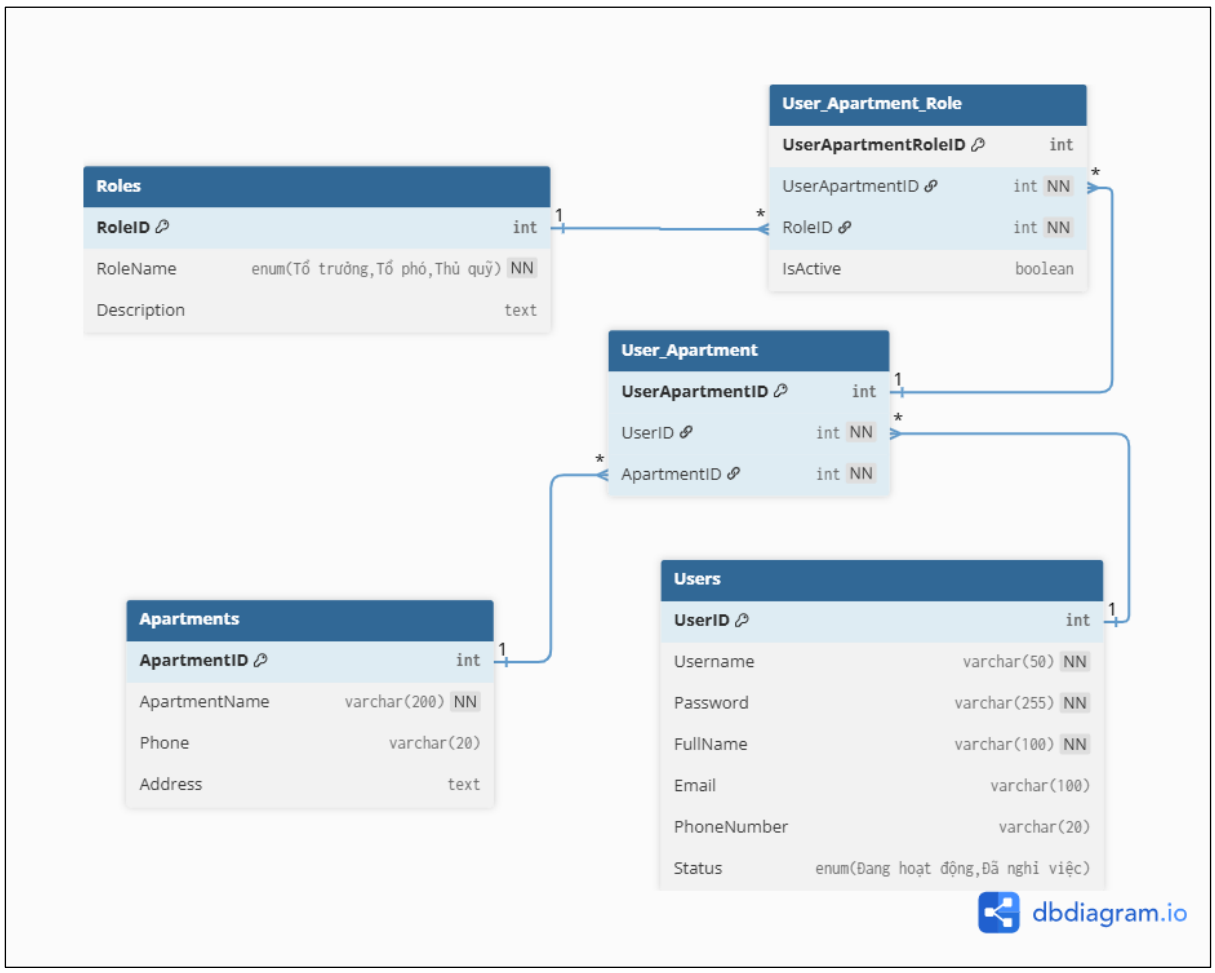
Chức năng dành cho mỗi tenant (mỗi chung cư):

- Quản lý hộ gia đình
- Quản lý nhân khẩu

- Quản lý các đợt thu phí
- Quản lý các loại phí (chỉ dành cho vai trò tổ trưởng)
- Quản lý xe cộ
- Quản lý người dùng (chỉ dành cho vai trò tổ trưởng)

4.3 Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu cho kiến trúc Multi-tenant trong hệ thống quản lý chung cư:



Trong hệ thống, mỗi chung cư được được xem như một **tenant** riêng biệt. Dữ liệu của từng chung cư được phân tách rõ ràng thông qua khóa ngoại **ApartmentID** của bảng **Apartments,** đảm bảo tính độc lập giữa các **tenant** trong cùng một hệ thống

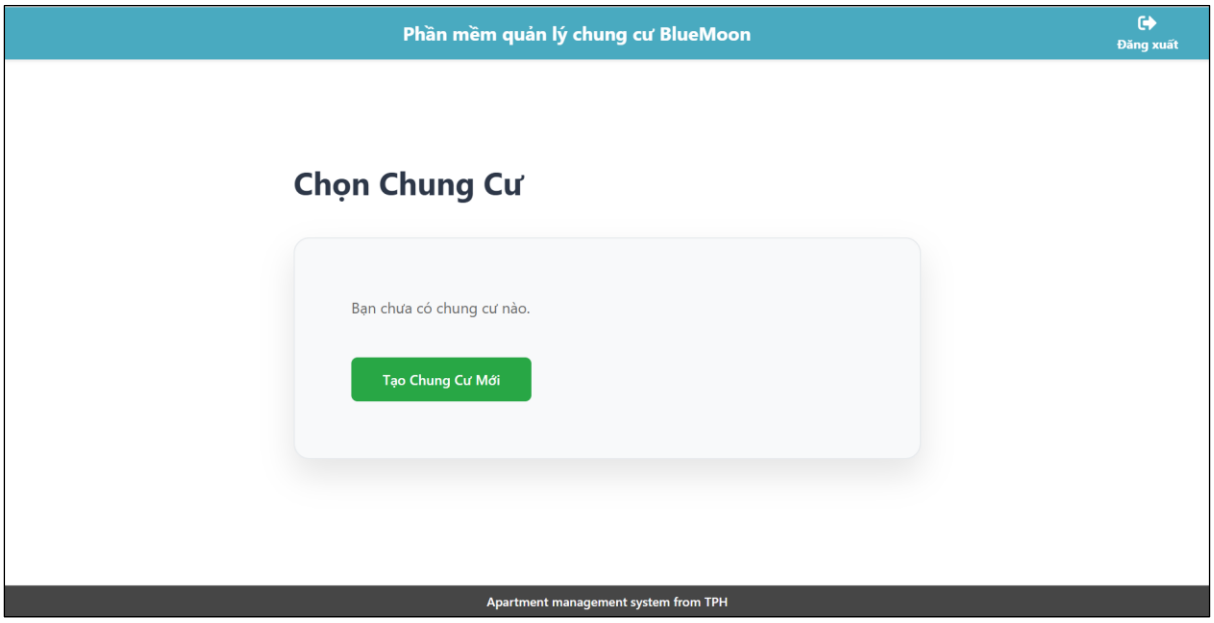
- Bảng **Apartments**: Đại diện cho từng chung cư – mỗi bản ghi là **tenant** riêng biệt.
- Bảng **Users**: Lưu thông tin người dùng của hệ thống. Bởi vì một người có thể thuộc nhiều chung cư nên không gắn **User** với một chung cư cố định

- Bảng **User_Apartment**: Nối giữa hai bảng **Users** và **Apartments** xác định người dùng nào đang tham gia vào chung cư nào. Một người dùng có thể tham gia vào nhiều chung cư và một chung cư có thể có nhiều người dùng.
- Bảng **Roles**: Định nghĩa các vai trò cho người dùng (**User**) cho toàn hệ thống (Tổ trưởng, tổ phó, thủ quỹ).
- Bảng **User_Apartment_Role**: Bảng nối giữa bảng **User_Apartment** và bảng **Roles**, vai trò (**Role**) cụ thể cho người dùng (**User**) trong từng chung cư (**Apartment**). Một người có thể là thủ quỹ của chung cư này nhưng lại làm tổ phó của chung cư kia.

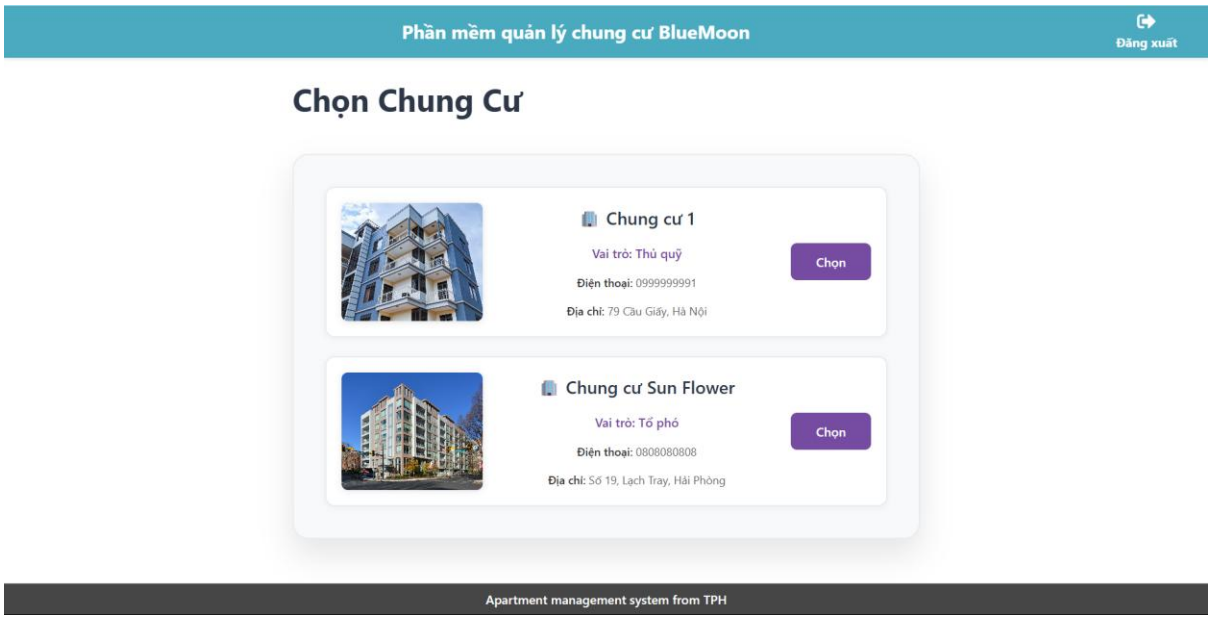
4.4 Các chức năng

Các chưng năng của hệ thống liên quan đến kiến trúc Multi-tenant

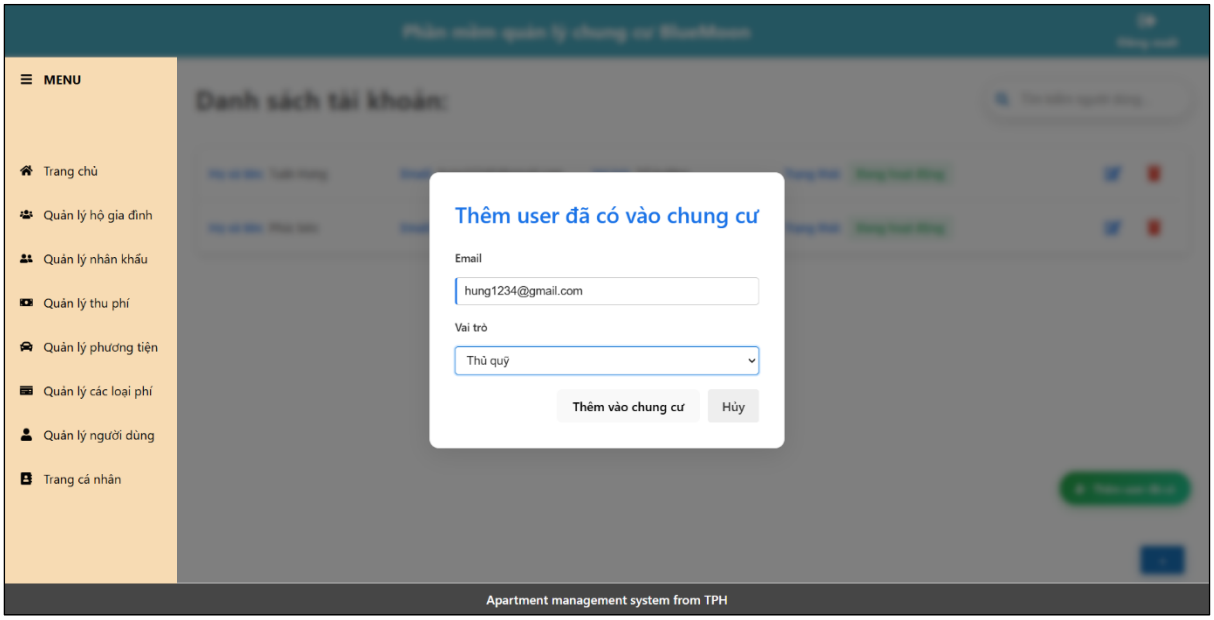
- Khởi tạo một **tenant** (chung cư) mới trong hệ thống: Sau khi đăng nhập, người dùng có thể khởi tạo chung cư của mình với hệ thống. Người khởi tạo sẽ giữ vai trò tổ trưởng (**admin**) của chung cư đó.



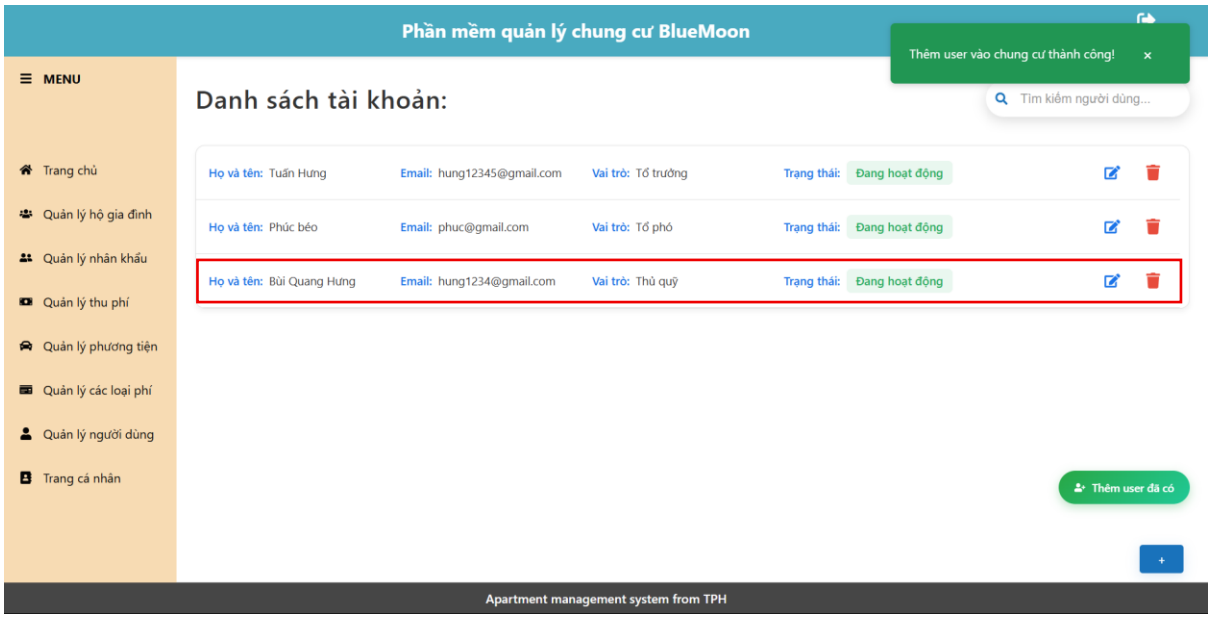
- Lựa chọn chung cư muốn làm việc:



- Chức năng thêm người dùng vào hệ thống chung cư: Tổ trưởng (admin) của chung cư sẽ thêm tài khoản người dùng vào trong chung cư của họ.



- Thao tác thêm tài khoản người dùng thành công, tài khoản đó xuất hiện trong danh sách các tài khoản thuộc chung cư được hệ thống hiển thị



5 Đánh giá kết quả đạt được

Sau quá trình tìm hiểu, áp dụng, thiết kế và triển khai kiến trúc đa khách hàng (Multi-tenant) vào hệ thống phần mềm quản lý chung cư, em đã đạt được các kết quả sau:

- **Hiểu và áp dụng thành công kiến trúc Multi-tenant**
  - Tìm hiểu và nắm vững lý thuyết về kiến trúc **Multi-tenant**, so sánh ưu nhược điểm để lựa chọn mô hình phù hợp cho hệ thống phần mềm quản lý chung cư.
  - Áp dụng thành công mô hình này vào thiết kế cơ sở dữ liệu, đảm bảo mỗi chung cư (**tenant**) có không gian dữ liệu riêng biệt, cách ly hoàn toàn và bảo mật.
- **Xây dựng một hệ thống quản lý chung cư, linh hoạt, mở rộng**
  - Cải thiện thành công hệ thống quản lý chung cư linh hoạt từ hệ thống phần mềm trước đó (mô hình đơn lẻ, chỉ áp dụng cho một chung cư duy nhất).
  - Hệ thống cho phép nhiều chung cư cùng sử dụng một phần mềm, nhưng mỗi chung cư lại có dữ liệu và quyền quản trị riêng.
  - Cơ sở dữ liệu và các API được thiết kế tối ưu để truy xuất, cập nhất dữ liệu theo từng **tenant**, đảm bảo tính năng mở rộng.
- **Đảm bảo tính bảo mật và phân quyền**

- Dữ liệu của các chung cư (**tenant**) được cách lý với nhau tuyệt đối nhờ sử dụng trường **ApartmentID** xuyên suốt các bảng dữ liệu.
- Xây dựng hệ thống phân quyền chặt chẽ theo vai trò (Tổ trưởng, Tổ phó, Thủ quỹ) đảm bảo người dùng chỉ truy cập được các chức năng phù hợp trong phạm vi với vai trò của mình.

- **Hạn chế và phướng phát triển**
  - Hệ thống hiện tại mới dừng ở mức đáp ứng các chức năng CRUD cơ bản cho từng **tenant**. Một số tính năng nâng cao như báo cáo tổng hợp đa **tenant**, tích hợp thanh toán trực tuyến, cảnh báo thông minh... có thể được phát triển thêm trong tương lai.
  - Cần tiếp tục kiểm thử, tối ưu hiệu năng khi số lượng **tenant** và dữ liệu tăng lớn.

6 Tài liệu tham khảo

Chong, F., Carraro, G., & Wolter, R. (2006). *Multi-Tenant Data Architecture*. Microsoft Corporation.

https://renatoargh.wordpress.com/wp-content/uploads/2018/01/article-multi-tenant-data-architecture-2006.pdf

Bezemer, C.-P., & Zaidman, A. (2010). *Multi-tenant SaaS applications: maintenance dream or nightmare?* In 2010 IEEE International Conference on Software Maintenance.
https://azaidman.github.io/publications/bezemerIWPSE2010.pdf

Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
https://ptgmedia.pearsoncmg.com/images/9780321127426/samplepages/0321127420.pdf

Erl, T., Mahmood, Z., & Puttini, R. (2013). *Cloud Computing: Concepts, Technology & Architecture*. Prentice Hall.

https://ptgmedia.pearsoncmg.com/images/9780133387520/samplepages/0133387526.pdf

Multitenant SaaS database tenancy patterns.

https://learn.microsoft.com/en-us/azure/azure-sql/database/saas-tenancy-app-design-patterns?view=azuresql