|  |
| --- |
| **BỘ THÔNG TIN VÀ TRUYỀN THÔNG**  TRẦN THƯ ĐẠT MSSV: N19DCCN036 CHUYÊN NGÀNH: CÔNG NGHỆ PHẦN MỀM 2019 – 2024 Lớp: D19CQCNPM01-N  **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  ---    **ĐỒ ÁN THỰC TẬP TỐT NGHIỆP**  Đề tài: “**Xây dựng ứng dụng web tìm kiếm đặt mua sách trực tuyến**”  Người hướng dẫn : Thạc sĩ Nguyễn Anh Hào  Sinh viên thực hiện : Trần Thư Đạt  Mã số sinh viên : N19DCCN036  Lớp : D19CQCNPM01-N  Khóa : 2019-2024  Hệ : ĐẠI HỌC CHÍNH QUY  TP HCM, tháng 7/2023  TP. HCM  2023  20… |

|  |
| --- |
| **BỘ THÔNG TIN VÀ TRUYỀN THÔNG**  **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  ---    **ĐỒ ÁN THỰC TẬP TỐT NGHIỆP**  Đề tài: “**Xây dựng ứng dụng web tìm kiếm đặt mua sách trực tuyến**”  Người hướng dẫn : Thạc sĩ Nguyễn Anh Hào  Sinh viên thực hiện : Trần Thư Đạt  Mã số sinh viên : N19DCCN036  Lớp : D19CQCNPM01-N  Khóa : 2019-2024  Hệ : ĐẠI HỌC CHÍNH QUY  TP HCM, tháng 7/2023 |

**LỜI CẢM ƠN**

 Lời nói đầu tiên, em xin được gửi lời cảm ơn chân thành nhất đến các thầy cô trong Khoa Công nghệ Thông tin 2 đã tạo mọi điều kiện thuận lợi để em có thể thực hiện được bài báo cáo thực tập tốt nghiệp này. Đồng thời, em xin gửi lời cảm ơn chân thành đến thầy phụ trách hướng dẫn trực tiếp cho em-thạc sĩ Lưu Nguyễn Kỳ Thư, đã không ngần ngại những khó khăn, vất vả để luôn tích cực hỗ trợ em hết mình từ ngày bắt đầu cho đến ngày hoàn thành bài báo cáo thực tập tốt nghiệp này.

Trong quá trình thực hiện bài viết báo cáo, nếu như có nhiều những thiếu sót, em xin các thầy cô trong khoa có thể tiếp tục hỗ trợ, đóng góp ý kiến, phản hồi cũng như cung cấp thêm những hướng dẫn chi tiết cụ thể về mặt chuyên môn để em có thể thực hiện được nội dung được hoàn chỉnh nhất cho quá trình thực tập tốt nghiệp, hướng đến hoàn thành đề tài đồ án tốt nghiệp ở tương lai.

Thành phố Hồ Chí Minh, tháng 7/2023

Sinh viên thực hiện

Trần Thư Đạt

**Mục lục**

[CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI 9](#_Toc140183622)

[1.1. Tên đề tài 9](#_Toc140183623)

[1.2. Loại đề tài 9](#_Toc140183624)

[1.3. Mục tiêu 9](#_Toc140183625)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT ĐỀ TÀI 10](#_Toc140183626)

[2.1 Xây dựng hệ thống 1 10](#_Toc140183627)

[2.2 Giới thiệu về PostgresSQL 11](#_Toc140183628)

[2.3 Giới thiệu về JAVA 11](#_Toc140183629)

[2.4 Giới thiệu về React JS 13](#_Toc140183630)

[2.5 Giới thiệu về Restful 14](#_Toc140183631)

[2.6 Giới thiệu về Apache Kafka 14](#_Toc140183632)

[2.7 Giới thiệu về Change Data Capture(CDC) và Debezium 15](#_Toc140183633)

[CHƯƠNG 3: KHẢO SÁT HỆ THỐNG 16](#_Toc140183634)

[3.1 Khảo sát hệ thống website hỗ trợ nhà đầu tư giao dịch chứng khoán 16](#_Toc140183635)

[3.2 Các chức năng của hệ thống. 17](#_Toc140183636)

[CHƯƠNG 4: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 19](#_Toc140183637)

[4.1 Thiết lập mô hình thực thể ERD 19](#_Toc140183638)

[4.2 Thiết kế cơ sở dữ liệu của Công ty chứng khoán 22](#_Toc140183639)

[CHƯƠNG 5: CHƯƠNG TRÌNH 38](#_Toc140183640)

[5.1 Tổng quan 38](#_Toc140183641)

[5.2 Thiết kế giao diện cho nhà đầu tư 41](#_Toc140183642)

[5.3 Thiết kế giao diện quản trị 56](#_Toc140183643)

[CHƯƠNG 6: KẾT LUẬN 58](#_Toc140183644)

[6.1 Kết quả đạt được và hạn chế 58](#_Toc140183645)

[6.2 Hướng phát triển 58](#_Toc140183646)

[PHỤ LỤC 59](#_Toc140183647)

[DANH MỤC TÀI LIỆU THAM KHẢO 65](#_Toc140183648)

**KÝ HIỆU CÁC CỤM TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| API: Application Programming Interface | Giao diện lập trình ứng dụng |
| ERD: Entity Relationship Diagram | Mô hình thực thể kết hợp |
| IDE: Integrated development environment | Môi trường phát triển tích hợp |
| NDT | Nhà Đầu Tư |
| REST: Representational State Transfer | Một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web |
| UBCK | Ủy Ban Chứng Khoán |
| UI: User Interface | Giao diện người dùng |

**Danh mục các bảng, sơ đồ, hình**

[Hình 4.1 Mô hình thực thể kết hợp ERD 19](#_Toc112677357)

[Hình 4.2 Mô hình diagram của cơ sở dữ liệu Công ty chứng khoán 21](#_Toc112677358)

[Hình 4.3 Quan hệ quy\_dinh\_cong\_ty và chi\_tiet\_quy\_dinh\_cong\_ty 22](#_Toc112677359)

[Hình 4.4 Quan hệ giữa nhan\_vien**-**tai\_khoan và quyen 22](#_Toc112677360)

[Hình 4.5 Quan hệ giữa co\_phieu**-**lich\_su\_gia**-**san\_giao\_dich**-**quy\_dinh\_san**-**chi\_tiet\_quy\_dinh\_san 23](#_Toc112677361)

[Hình 4.6 Quan hệ giữa nha\_dau\_tu và co\_phieu 24](#_Toc112677362)

[Hình 4.7Quan hệ giữa ngan\_hang**-**tai\_khoan\_ngan\_hang và lenh\_ung 25](#_Toc112677363)

[Hình 4.8 Quan hệ giữa lenh\_dat và co\_phieu cùng với các bảng liên quan 25](#_Toc112677364)

[Hình 5.1 Mô hình Client-Server 39](#_Toc112677365)

[Hình 5.2 Mô hình thiết kế hệ thống Kafka giữa Công ty chứng khoán và Sở giao dịch 40](#_Toc112677366)

[Hình 5.3 Xem thông tin cá nhân, đổi mật khẩu/mã PIN 42](#_Toc112677367)

[Hình 5.4 Chọn chức năng quên mật khẩu 43](#_Toc112677368)

[Hình 5.5 Email gửi mã xác thực khôi phục tài khoản 43](#_Toc112677369)

[Hình 5.6 Nhập mã xác thực để tiền hành cập nhật mật khẩu mới 44](#_Toc112677370)

[Hình 5.7 Cập nhật mật khẩu mới cho lần đăng nhập kế tiếp 44](#_Toc112677371)

[Hình 5.8 Cập nhật mật khẩu mới 44](#_Toc112677372)

[Hình 5.9 Cập nhật mã PIN mới 45](#_Toc112677373)

[Hình 5.10 Sao kê lệnh đặt trong ngày 46](#_Toc112677374)

[Hình 5.11 Sao kê lệnh đặt/lệnh khớp trong một khoảng thời gian 47](#_Toc112677375)

[Hình 5.12 Quy tắc số dư T2 dành cho cổ phiếu 48](#_Toc112677376)

[Hình 5.13 Quy tắc số dư T2 dành cho Tài khoản ngân hàng 48](#_Toc112677377)

[Hình 5.14 Số dư cổ phiếu T2 ban đầu của nhà đầu tư 49](#_Toc112677378)

[Hình 5.15 Số dư T2 của cổ phiếu MWG tăng lên 100 cổ phiếu 49](#_Toc112677379)

[Hình 5.16 Số dư cổ phiếu được cập nhật vào lúc 8h sáng trước khi phiên giao dịch bắt đầu 49](#_Toc112677380)

[Hình 5.17 Cấu hình job trên PostgreSQL chạy vào 8h sáng các ngày có phiên giao dịch 50](#_Toc112677381)

[Hình 5.18 Số dư tài khoản ngân hàng ban đầu 52](#_Toc112677382)

[Hình 5.19 Số dư T2 được cập nhật sau khi khớp lệnh 52](#_Toc112677383)

[Hình 5.20 Số dư T2 được cập nhật sau 15h 53](#_Toc112677384)

[Hình 5.21 Cấu hình job trên PostgreSQL chạy vào 15h các ngày có phiên giao dịch 53](#_Toc112677385)

[Hình 5.22 Thông tin cổ phiểu đang sở hữu 55](#_Toc112677386)

[Hình 5.23 Sao kê thông tin tài khoản ngân hàng 55](#_Toc112677387)

[Hình 5.24 Quản lý danh sách tài khoản ngân hàng 56](#_Toc112677388)

[Hình 5.25 Thêm tài khoản ngân hàng mới 56](#_Toc112677389)

[Hình 5.26 Hủy liên kết tài khoản ngân hàng 56](#_Toc112677390)

[Hình 5.27: Trang quản lý quy định công ty 57](#_Toc112677391)

# GIỚI THIỆU ĐỀ TÀI

1.1. Tên đề tài : Xây dựng ứng dụng web tìm kiếm đặt mua sách trực tuyến.

1.2. Loại đề tài : Xây dựng một phần mềm ứng dụng.

1.3. Mục đích :

# CƠ SỞ LÝ THUYẾT ĐỀ TÀI

## 2.1 Xây dựng hệ thống 1

### 2.1.1 Giới thiệu về IntelliJ IDEA

IntelliJ IDEA là một IDE Java để phát triển các phần mềm máy tính. Nó được phát triển bởi JetBrains, được cấp phép Apache 2 cho một phiên bản cộng đồng và một phiên bản thương mại độc quyền. Cả hai có thể được sử dụng cho phát triển các ứng dụng lập trình mang tính thương mại.

Các lợi ích của việc sử dụng IntelliJ IDEA :

* Cho phép các nhà phát triển tập trung phát triển và quản lý tất cả các tác vụ thông thường.
* Cho phép viết, gỡ lỗi, tái cấu trúc, kiểm tra và tìm hiểu mã của bạn mà không bị ảnh hưởng.
* Xử lý liền mạch cơ sở mã hỗn hợp của Java, Ruby, Groovy, Python và Scala.
* Tự động duy trì chất lượng mã.
* Theo dõi và sửa lỗi trên tất cả các cấp độ – từ các câu đến kiến trúc tổng thể.
* Tạo mã “sạch”, nhanh chóng thực hiện mã trong thời gian ngắn nhất.
* Được thiết kế để làm việc trên tất cả các quy mô – từ cá nhân đến doanh nghiệp.
* Hỗ trợ tất cả các ngôn ngữ, công nghệ và framework chính.
* Làm việc với các hệ thống điều khiển phiên bản phổ biến và TeamCity, server tích hợp liên tục.

### 2.1.2 Giới thiệu về Visual studio Code

Visual Studio Code (VS Code hay VSC) là một trong những trình soạn thảo mã nguồn phổ biến nhất được sử dụng bởi các lập trình viên. Nhanh, nhẹ, hỗ trợ đa nền tảng, nhiều tính năng và là mã nguồn mở chính là những ưu điểm vượt trội khiến VS Code ngày càng được ứng dụng rộng rãi.

Visual Studio Code hỗ trợ chức năng debug, đi kèm với Git, có syntax highlighting, tự hoàn thành mã thông minh, snippets, và cải tiến mã nguồn. Nhờ tính năng tùy chỉnh, Visual Studio Code cũng cho phép người dùng thay đổi theme, phím tắt, và các tùy chọn khác.

## 2.2 Giới thiệu về PostgresSQL

Giới thiệu đôi nét về PostgreSQL: [PostgreSQL](https://bizflycloud.vn/tin-tuc/postgresql-la-gi-tim-hieu-ve-co-so-du-lieu-ma-nguon-mo-tien-tien-nhat-the-gioi-20180919175924611.htm) là một hệ thống quản trị cơ sở dữ liệu quan hệ-đối tượng (object-relational database management system) có mục đích chung, hệ thống cơ sở dữ liệu mã nguồn mở tiên tiến nhất hiện nay.

PostgreSQL được phát triển dựa trên POSTGRES 4.2 tại phòng khoa học máy tính Berkeley, Đại học California.

PostgreSQL được thiết kế để chạy trên các nền tảng tương tự UNIX. Tuy nhiên, PostgreSQL sau đó cũng được điều chỉnh linh động để có thể chạy được trên nhiều nền tảng khác nhau như Mac OS X, Solaris và Windows.

PostgreSQL là một phần mềm mã nguồn mở miễn phí. Mã nguồn của phần mềm khả dụng theo license của PostgreSQL, một license nguồn mở tự do. Theo đó, ta sẽ được tự do sử dụng, sửa đổi và phân phối PostgreSQL dưới mọi hình thức.

PostgreSQL không yêu cầu quá nhiều công tác bảo trì bởi có tính ổn định cao. Do đó, nếu bạn phát triển các ứng dụng dựa trên PostgreSQL, chi phí sở hữu sẽ thấp hơn so với các hệ thống quản trị dữ liệu khác.

## 2.3 Giới thiệu về JAVA

### 2.3.1 Tổng quan về Java

Java (phiên âm Tiếng Việt: "Gia-va") là một ngôn ngữ lập trình hướng thuần đối tượng. Java được thiết kế là ngôn ngữ lập trình có mục đích chung cho phép các nhà phát triển ứng dụng viết một lần, chạy ở mọi nơi, nghĩa là mã Java đã biên dịch có thể chạy trên tất cả các nền tảng hỗ trợ Java mà không cần biên dịch lại.

Các ứng dụng Java thường được biên dịch thành bytecode có thể chạy trên bất kỳ máy ảo Java (JVM) nào bất kể kiến trúc máy tính bên dưới. Cú pháp của Java tương tự như C và C++, nhưng có ít cơ sở cấp thấp hơn các ngôn ngữ trên. Java runtime cung cấp các khả năng động (chẳng hạn như phản ánh và sửa đổi mã thời gian chạy) thường không có sẵn trong các ngôn ngữ biên dịch truyền thống. Tính đến năm 2021 Java là một trong những ngôn ngữ lập trình phổ biến nhất được sử dụng.

### 2.3.2 Lợi ích của việc sử dụng Java

Ứng dụng viết bằng ngôn ngữ lập trình Java có thể chạy trên nhiều nền tảng khác nhau thông qua sự hỗ trợ của JVM(Java Vitural Machine).

Java là ngôn ngữ thuần hướng đối tượng, dễ đọc, cú pháp ràng buộc chặt chẽ. Hỗ trợ việc mở rộng, bảo trì phần mềm về sau.

Java hỗ trợ thực thi nhiều chương trình đồng thời, hỗ trợ đa luồng.

Java hỗ trợ nhiều tính năng đặc biệt như thu gom rác(Garbage collection), không sử dụng con trỏ(Pointer), hỗ trợ xử lý ngoại lệ(Exception).

### 2.3.3 Tổng quan về Java Spring Boot

[**Spring Boot**](https://topdev.vn/viec-lam-it/spring-boot-kt4317) là một dự án phát triển bởi **JAV**(ngôn ngữ Java) trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển business cho ứng dụng. Spring Boot là một framework rất mạnh và rất phổ biến để xây dựng nên các API bằng ngôn ngữ lập trình Java.

### 2.3.4 Tổng quan về JPA

JPA là viết tắt của Java Persistence API, là một tập hợp các tiêu chuẩn của Java để làm việc với cơ sở dữ liệu quan hệ.

JPA có thể xem như cầu nối giữa Java object và cơ sở dữ liệu quan hệ. Là một tập hợp các đặc tả (các interface), JPA không chứa bất kỳ phương thức thực thi nào, nó cần một JPA implementation triển khai tất cả các đặc tả mà nó định nghĩa. Những ORM tools như Hibernate, TopLink đều cung cấp trình triển khai cho JPA.

ORM viết tắt của object-relational-mapping, công nghệ cho phép chuyển đổi từ các object trong ngôn ngữ hướng đối tượng sang database quan hệ và ngược lại.

Hibernate là một trong những ORM tools phổ biến được sử dụng nhiều trong các ứng dụng Java. Từ phiên bản 3.2 trở về sau, Hibernate đã cung cấp một JPA Implement và được sử dụng khá rộng rãi trong cộng đồng Java, trong chương trình này chúng em cũng sử dụng JPA được chính Hibernate cung cấp. Ngoài ra trong cộng động Java, chúng ta còn có một số JPA implementaion khác như Apache OpenJPA, EclipeseLink, jOOQ etc.

### 2.3.5 Tổng quan về Spring Security

Spring Security thực sự chỉ là một loạt các bộ lọc servlet giúp bạn thêm authentication và authorization vào ứng dụng web của mình.

Spring Security cũng tích hợp tốt với các framework như Spring Web MVC (hay Spring Boot), cũng như với các tiêu chuẩn như OAuth2 hoặc SAML. Và nó tự động tạo các trang login / logout và bảo vệ chống lại các hành vi khai thác thông tin như CSRF.

Việc sử dụng Spring Security sẽ đảm bảo tính xác thực và phân quyền trong hệ thống, đảm bảo hệ thống hoạt động an toàn và đúng với mục đích đề ra, Spring Security hỗ trợ tạo ra những JWT Token-là chìa khóa đã thực hiện mọi thao tác trong chương trình.

### 2.3.6 Tổng quan về Spring Mail

JavaMail là một API được sử dụng để soạn, gửi và đọc các tin điện tử (email).

JavaMail API cung cấp độc lập về giao thức  và độc lập với nền tảng cho việc gửi và nhận email. Nghĩa là ta không cần bận tâm về giao thức gửi nhận mail là SMTP, POP, IMAP, …Các gói **javax.mail** và **javax.mail.activation** là 2 lớp lõi JavaMail API, nơi cung cấp các phương thức hỗ trợ việc gởi và nhận mail.

## 2.4 Giới thiệu về React JS

ReactJS là một opensource được phát triển bởi **Facebook**, bản thân nó là một thư viện Javascript được dùng để để xây dựng các tương tác với các thành phần trên website. Một trong những điểm nổi bật nhất của ReactJS đó là việc render dữ liệu không chỉ thực hiện được trên tầng Server mà còn ở dưới Client nữa.

* Một số thư viện, package khác:
  + **React router dom(V6)**: là một thư viện định tuyến (routing) tiêu chuẩn trong React. Nó giữ cho giao diện của ứng dụng đồng bộ với URL trên trình duyệt. React-Router cho phép bạn định tuyến "luồng dữ liệu" (data flow) trong ứng dụng của bạn một cách rõ ràng. Nó tương đương với sự khẳng định, nếu bạn có URL này, nó sẽ tương đương với Route này, và giao diện tương ứng.
  + **Redux**: là một predictable state management tool cho các ứng dụng Javascript. Nó giúp bạn viết các ứng dụng hoạt động một cách nhất quán, chạy trong các môi trường khác nhau (client, server, and native) và dễ dàng để test. Redux ra đời lấy cảm hứng từ tư tưởng của ngôn ngữ Elm và kiến trúc Flux của Facebook. Do vậy Redux thường dùng kết hợp với React.
  + **Axios**: Axios là một thư viện HTTP Client dựa trên Promise. Cơ bản thì nó cung cấp một API cho việc xử lý XHR (XMLHttpRequests).
  + **MUI (Material UI):** MUI là một thư viện các React Component đã được tích hợp thêm cả Google's Material Design. Theo như giới thiệu trên trang chủ thì được xây dựng kết hợp để có thể tích hợp với React. Do đó mà phần hướng dẫn trên trang chủ của Material UI cũng đã nói nên sử dụng Material UI với React.
  + **Các style được sử dụng để xây dựng giao diện:** SCSS, Style Component

## 2.5 Giới thiệu về Restful

**REST** là viết tắt của Representational State Transfer. **RESTful** **API** là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau.

Chức năng quan trọng nhất của REST là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các resource. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một RESTful API.

## 2.6 Giới thiệu về Apache Kafka

Kafka, tên gọi đầy đủ là Apache Kafka, là một nền tảng streaming phân tán sự kiện (distributed event streaming), chủ yếu được áp dụng làm hệ thống phân tán, “vận chuyển” tin nhắn và thu thập, xử lý, lưu trữ và phân tích dữ liệu ở quy mô lớn. Kafka trước đây được phát triển bởi LinkedIn, hiện tại thuộc Apache.

Ngôn ngữ chính của Kafka là **Java** hoặc **Scala**.

Kafka là một hệ thống Event Streaming, do đó tính năng chủ yếu của Kafka chính là Event streaming. Event streaming là hành động thu thập dữ liệu dưới dạng các luồng sự kiện trong thời gian thực từ các nguồn sự kiện như cơ sở dữ liệu, cảm biến và thiết bị di động và lưu trữ các luồng sự kiện này một cách lâu dài để truy xuất sau này, phân tích, xử lý chúng trong thời gian thực và định tuyến các luồng sự kiện tới các công nghệ đích khác nhau khi cần thiết.

Nhìn chung, Kafka quản lý một lượng lớn dữ liệu di chuyển từ điểm cuối này sang điểm cuối khác và đảm bảo rằng tất cả dữ liệu đều được phân phối đến đúng nơi, đúng thời điểm. Kafka nhập, lưu trữ và phân tích luồng dữ liệu khi chúng được tạo, ở bất kỳ quy mô nào, từ xử lý dữ liệu thời gian thực đến lập trình luồng dữ liệu.

Các thuật ngữ cơ bản khi sử dụng Kafka:

* **Topic**: tên đại diện của một kênh cho phép các hệ thống giao tiếp với nhau. Chúng ta cũng có thể hiểu Topic là tổng hợp của một dãy event tương tự với nhau, ví dụ: giống như các record được lưu trong cùng một table SQL.
* **Event**: Một sự kiện, hành động được diễn ra, hay có tên gọi khác là message, đó chính là thông tin được truyền từ nơi này sang nơi khác.
* **Stream**: Chuyển hóa luồng topic đầu vào thành topic đầu ra, bản chất chính là các dòng message.
* **Partition** – Phân vùng: Message đưa ra sẽ được Kafka chỉ định partition để biết message đó được lưu chính xác tại nơi đâu. Một topic có thể có một hoặc nhiều partition.
* **Publisher/Producer**: tên gọi chung của những nguồn có thể publish message lên nền tảng Kafka.
* **Consumer**: là những nguồn có thể subscribe vào một hoặc nhiều topic cụ thể và nhận message thuộc topic đó từ hệ thống.
* **Broker**: Server lưu trữ các topic.
* **Connector**: Kết nối Kafka với các hệ thống cơ sở dữ liệu khác.

## 2.7 Giới thiệu về Change Data Capture(CDC) và Debezium

### 2.7.1 Tổng quan về Change data capture

Change Data Capture(hay viết tắt là CDC) là hành động nhận biết các thay đổi trong database và là căn cứ để ta có hành động tương ứng với những thay đổi đó theo thời gian thực.

Trong những môi trường dữ liệu thường xuyên thay đổi, bên cạnh khả năng đọc ghi thì thời gian truy xuất dữ liệu cũng là yếu tố quyết định đến hiệu suất công việc.

Change Data Capture là một giải phải phù hợp cho việc đảm bảo quản lý sự thay đổi trong database với độ trễ thấp, đáng tin cậy và có khả năng mở rộng theo kích thước của dữ liệu.

### 2.7.2 Tổng quan về Debezium

Debezium là một nền tảng phân tán mã nguồn mở để thu thập dữ liệu thay đổi(Change Data Capture). Việc chúng ta cần làm là khởi động Debezium, trỏ nó vào cơ sở dữ liệu và các ứng dụng cần theo dõi, từ đó ta có thể nhận các phản hồi từ tất cả các thao insert/update/delete được thực hiện trong database.

Debezium nhanh và an toàn vì có khả năng thực hiện với cường độ cao, vì vậy ta có thể nhận phản hồi nhanh chóng và không bao giờ bỏ lỡ sự kiện, ngay cả khi có sự cố.

Bên cạnh việc sử dụng Debezium để theo dõi sự thay đổi của dữ liệu thì ta còn có một phương pháp khác nữa đó chính là Trigger. Tuy nhiên, khi so sánh với trigger, Debezium lại cho hiệu suất tốt hơn, vì đơn giản Debezium theo dõi sự thay đổi trong file log của database và độc lập với database, trong khi đó trigger lại lại một thành phần được ta cấu hình và gắn liền với từng table trong database(tức gắn liền với database). Do đó, vô hình chung việc sử dụng trigger sẽ không đạt được hiệu suất tốt bằng Debezium, đặc biệt trong những trường hợp việc cập nhật dữ liệu diễn ra liên tục.

# KHẢO SÁT HỆ THỐNG

## 3.1 Khảo sát hệ thống website hỗ trợ nhà đầu tư giao dịch chứng khoán

Một công ty hỗ trợ nhà đầu tư giao dịch chứng khoán được tổ chức như sau :

***Thông tin về Công ty:***

Mỗi công ty sẽ có nhiều nhà đầu tư.

Ngoài ra còn có những quy định về công ty Mã quy định, Tên quy định.

Mỗi quy định công ty có những chi tiết quy định công ty gồm có Mã quy định, ngày áp dụng, giá trị.

***Thông tin về các nhà đầu tư:***

Mỗi nhà đầu tư (NDT) sẽ có 1 mã tài khoản (Mã nhà đầu tư) duy nhất (do công ty chứng khoán cấp) để phân biệt với các nhà đầu tư khác, họ, tên, ngày sinh, nơi sinh, phái, địa chỉ, email, điện thoại, số CMND/CCCD/Passport, ngày cấp, nơi cấp, trạng thái tài khoản của nhà đầu tư, mật mã giao dịch, mật khẩu đặt lệnh, ngày đăng ký tài khoản.

Mỗi NDT còn phải cung cấp mã tài khoản trong ngân hàng, và tên ngân hàng mà mình gửi tiền. Một nhà đầu tư có thể có nhiều tài khoản tiền trong 1 ngân hàng, hoặc nhiều ngân hàng.

***Thông tin về các cổ phiếu:***

Mã cổ phiếu thực chất là mã của công ty khi niêm yết lên sàn giao dịch đã được UBCK nhà nước cấp.

Mỗi công ty sẽ có một mã cổ phiếu (MACP) duy nhất, có tên công ty, địa chỉ, điện thoại, Fax, địa chỉ website và email, số lượng cổ phiếu niêm yết.

Mỗi mã cổ phiếu sẽ có giá sàn, giá tham chiếu, giá trần trong từng ngày.

* Giá tham chiếu là giá đóng cửa của ngày hôm trước.
* Giá sàn = giá tham chiếu – biên độ giá\* giá tham chiếu.
* Giá trần = giá tham chiếu + biên độ giá\* giá tham chiếu.

Hiện tại có 3 sàn giao dịch chứng khoán: sàn HSX của thành phố HCM, HNX, UPCOM.

Một cổ phiếu chỉ được quyền đăng ký giao dịch tại 1 sàn tại một thời điểm.

Mỗi sàn giao dịch có Tên sàn và mã sàn, một sàn giao dịch thì có nhiều cổ phiếu.

Mỗi sàn sẽ có nhiều Quy định khác nhau. Quy định sàn có Mã quy định, Tên quy định.

Mỗi quy định sàn sẽ có chi tiết quy định sàn sẽ có Mã sàn, Mã quy định, ngày áp dụng, giá trị.

***Thông tin giao dịch:* (đặt lệnh mua hay bán)**

Trong quá trình giao dịch sẽ có nhiều loại lệnh LO (Khớp lệnh liên tục), ATO/ATC (Khớp lệnh định kỳ), MB (Lệnh thị trường) …

Một lệnh đặt chỉ thuộc 1 loại lệnh, Với 1 loại lệnh có thể có nhiều lệnh đặt.

Một NDT có thể đặt lệnh mua (hay bán) một cổ phiếu nào đó. Thông tin về lệnh đặt gồm có loại giao dịch (Mua hay bán), số lượng đặt, giá đặt, ngày giờ đặt, mã cổ phiếu cần giao dịch, loại lệnh đặt. Khi đặt lệnh, NDT phải chỉ rõ tài khoản cần thiết để mua hoặc bán.

Một lệnh đặt chỉ thuộc về1 tài khoản; 1 tài khoản có thể được dùng để đặt nhiều lệnh.

Trạng thái của một lệnh khi chưa khớp sẽ là ‘Chờ khớp’, nếu lệnh được khớp thì trạng thái sẽ chuyển thành ‘Khớp 1 phần’ hoặc ‘Khớp hết’. Nếu NDT hủy lệnh thì trạng thái lệnh là ‘Đã hủy’. Hết phiên các lệnh tự chuyển sang “Chưa Khớp”.

Mỗi lệnh khớp sẽ có thêm giá khớp, khối lượng khớp và thời gian khớp.

Một lệnh mua hoặc bán có thể khớp với nhiều lệnh bán/mua khác, nhưng 1 lệnh khớp chỉ thuộc 1 lệnh đặt mua/bán. Mỗi lệnh khớp có thể có số lượng khớp và giá khớp khác nhau.

Cổ phiếu hoặc số tiền đã khớp ở ngày T thì đến ngày T+2 mới được cập nhật vào tài khoản của nhà đầu tư.

Một NDT có thể sở hữu nhiều mã cổ phiếu. Một mã cổ phiếu có thể có nhiều NDT sở hữu.

Lịch sử giá chứa các thông tin của những cổ phiếu giao dịch bao gồm giá tham chiếu, giá trần, giá sàn, giá đóng cửa, giá mở cửa theo ngày.

## Các chức năng của hệ thống.

### 3.2.1 Yêu cầu chức năng của hệ thống:

Công việc của người quản lý:

* Xử lý xác nhận duyệt tài khoản cho nhà đầu tư.
* Quản lý thông tin nhà đầu tư.

Công việc của nhà đầu tư tham gia giao dịch:

* Đăng ký tài khoản (chờ xác nhận của công ty) / Đăng nhập
* Xem thông tin cổ phiếu trên thị trường và cổ phiếu đang sở hữu, yêu thích.
* Đặt lệnh mua hoặc bán.
* Xem thông tin lịch sử giao dịch đặt lệnh và khớp lệnh trong ngày.
* Xem thông tin cá nhân, đổi mật khẩu và mã pin.
* Xem thông tin tài khoản tiền, cổ phiếu theo từng thời gian T0, T1, T2.
* Quản lý thêm/xóa tài khoản ngân hàng của chính mình.
* Ứng tiền trong hệ thống.

Đối tượng tham gia hệ thống:

* Nhà đầu tư tham gia mua hoặc bán cổ phiếu
* Nhân viên quản lý hệ thống

### 3.2.2 Yêu cầu phi chức năng:

Hệ thống đáng tin cậy, chính xác, giao diện thân thiện, dễ sử dụng, truy cập dữ liệu nhanh chóng.

Đảm bảo tính bảo mật cho người điều hành hệ thống.

Đảm bảo tính bảo mật cho nhà đầu tư.

Phải có tính linh hoạt cao.

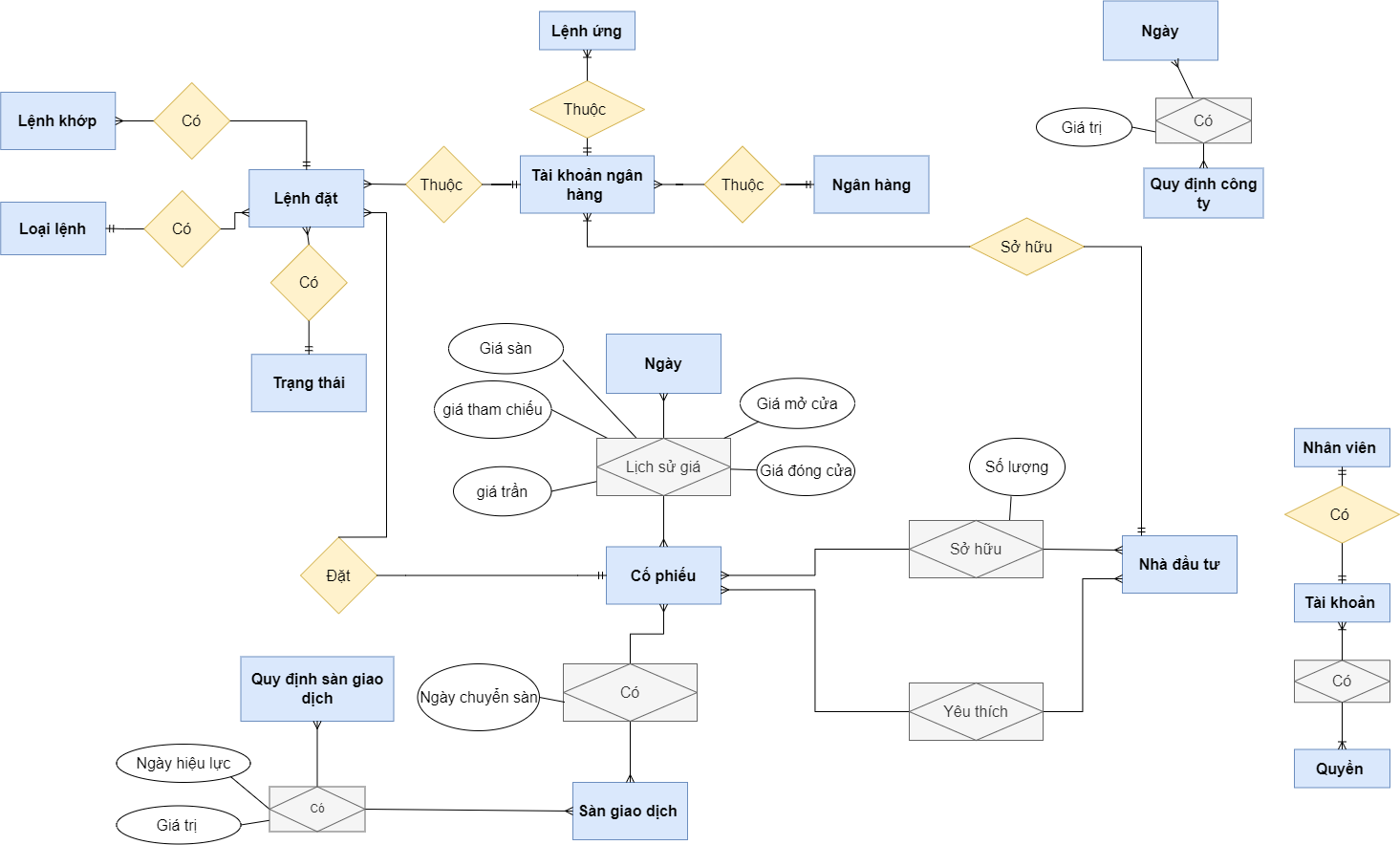
# PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 4.1 Thiết lập mô hình thực thể ERD

### 4.1.1 Xác định các thực thể

* **nha\_dau\_tu** (ma\_nha\_dau\_tu, cccd, dia\_chi, email, ho, ten, mat\_khau\_dat\_lenh, mat\_khau\_giao\_dich, ngay\_cap\_cccd, ngay\_dang\_ky, ngay\_sinh, noi\_cap\_cccd, noi\_sinh, phai, phone, trang\_thai, xac\_thuc\_email)
* **co\_phieu** (ma\_co\_phieu, ten\_cong\_ty, dia\_chi, email, fax, phone, so\_luong, website)
* **tai\_khoan\_ngan\_hang** (so\_tai\_khoan, so\_du)
* **ngan\_hang** (ma\_ngan\_hang, ten\_ngan\_hang)
* **lenh\_da**t (ma\_lenh\_dat, gia, loai\_giao\_dich, so\_luong, thoi\_gian)
* **trang\_thai\_lenh** (ma\_trang\_thai, ten\_trang\_thai)
* **lenh\_khop** (ma\_lenh\_khop, gia, so\_luong, thoi\_gian)
* **loai\_lenh** (ma\_loai\_lenh, ten\_loai\_lenh)
* **san\_giao\_dich** (ma\_san, ten\_san)
* **quy\_dinh\_san** (ma\_quy\_dinh, ten\_quy\_dinh)y
* **quy\_dinh\_cong\_ty** (ma\_quy\_dinh, ten\_quy\_dinh)
* **quyen** (ma\_quyen, ten\_quyen)
* **nhan\_vien** (ma\_nhan\_vien, cccd, ho, ten, phai, email, phone)
* **tai\_khoan** (ma\_tai\_khoan, ten\_dang\_nhap, mat\_khau)
* **lenh\_ung** (ma\_lenh\_ung, ngay\_yeu\_cau, ngay\_nhan, phi\_ung, so\_tien)

### 4.1.2 Mô hình thực thể kết hợp ERD(Entity RelationShip Diagram)



Hình 4.1 Mô hình thực thể kết hợp ERD

### 4.1.3 Mô hình dữ liệu quan hệ

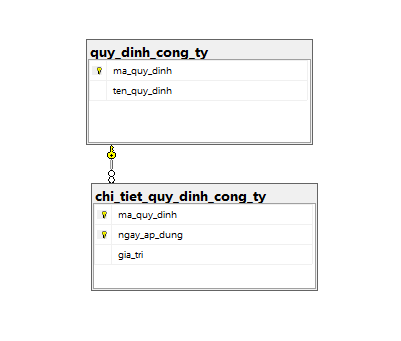
* **nha\_dau\_tu** (ma\_nha\_dau\_tu, cccd, dia\_chi, email, ho, ten, mat\_khau\_dat\_lenh, mat\_khau\_giao\_dich, ngay\_cap\_cccd, ngay\_dang\_ky, ngay\_sinh, noi\_cap\_cccd, noi\_sinh, phai, phone, trang\_thai, xac\_thuc\_email)
* **co\_phieu** (ma\_co\_phieu, ten\_cong\_ty, dia\_chi, email, fax, phone, so\_luong, website)
* **tai\_khoan\_ngan\_hang** (so\_tai\_khoan, so\_du, ma\_nha\_dau\_tu, ma\_ngan\_hang)
* **ngan\_hang** (ma\_ngan\_hang, ten\_ngan\_hang)
* **lenh\_dat** (ma\_lenh\_dat, gia, loai\_giao\_dich, so\_luong, thoi\_gian, ma\_co\_phieu, ma\_tai\_khoan, ma\_loai\_lenh, ma\_trang\_thai)
* **trang\_thai\_lenh** (ma\_trang\_thai, ten\_trang\_thai)
* **lenh\_khop** (ma\_lenh\_khop, gia, so\_luong, thoi\_gian, ma\_lenh\_dat)
* **loai\_lenh** (ma\_loai\_lenh, ten\_loai\_lenh)
* **san\_giao\_dich** (ma\_san, ten\_san)
* **quy\_dinh\_san** (ma\_quy\_dinh, ten\_quy\_dinh)
* **chi\_tiet\_quy\_dinh\_san** (ma\_quy\_dinh, ma\_san, ngay\_ap\_dung, gia\_tri)
* **thong\_tin\_chuyen\_san** (ma\_co\_phieu, ma\_san, ngay\_chuyen\_san)
* **quy\_dinh\_cong\_ty** (ma\_quy\_dinh, ten\_quy\_dinh)
* **chi\_tiet\_quy\_dinh\_cong\_ty** (ma\_quy\_dinh, ngay\_ap\_dung, gia\_tri)
* **quyen** (ma\_quyen, ten\_quyen)
* **nhan\_vien** (ma\_nhan\_vien, cccd, ho, ten, phai, email, phone)
* **tai\_khoan** (ma\_tai\_khoan, ten\_dang\_nhap, mat\_khau, ma\_nhan\_vien)
* **taikhoan\_quyen** (ma\_tai\_khoan, ma\_quyen)
* **lenh\_ung** (ma\_lenh\_ung, ngay\_yeu\_cau, ngay\_ban, phi\_ung, so\_tien, tai\_khoan\_ngan\_hang)
* **lich\_su\_gia** (ma\_co\_phieu, ngay, gia\_dong\_cua, gia\_mo\_cua, gia\_tham\_chieu, gia\_san, gia\_tran)
* **so\_huu** (ma\_co\_phieu, ma\_nha\_dau\_tu, soluong)
* **yeu\_thich** (ma\_co\_phieu, ma\_nha\_dau\_tu)

## 4.2 Thiết kế cơ sở dữ liệu của Công ty chứng khoán

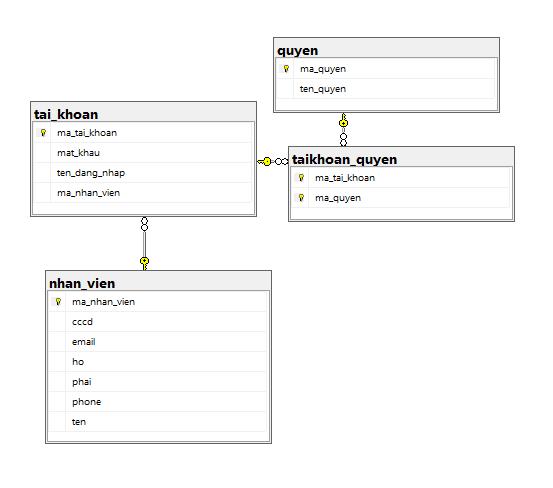
### 4.2.1 Mô hình diagram của cơ sở dữ liệu Công ty chứng khoán



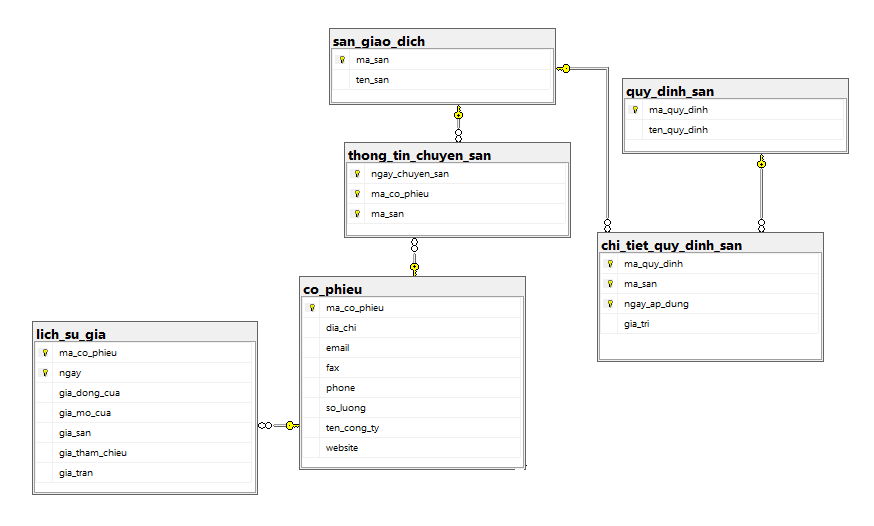
Hình 4.2 Mô hình diagram của cơ sở dữ liệu Công ty chứng khoán



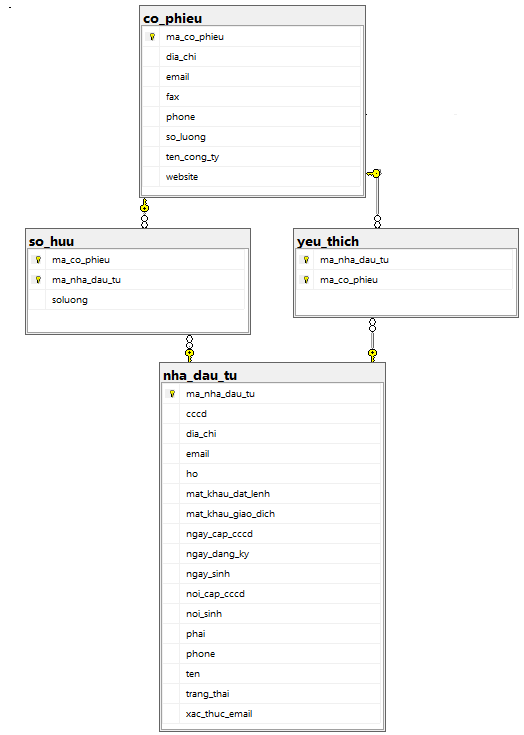
Hình 4.3 Quan hệ quy\_dinh\_cong\_ty và chi\_tiet\_quy\_dinh\_cong\_ty



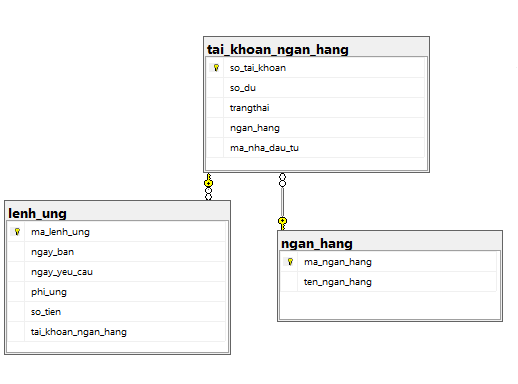
Hình 4.4 Quan hệ giữa nhan\_vien**-**tai\_khoan và quyen



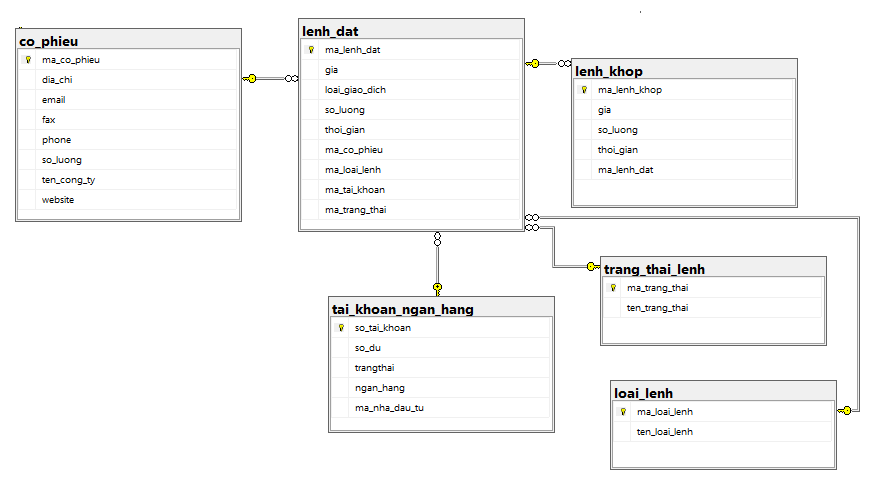
Hình 4.5 Quan hệ giữa co\_phieu**-**lich\_su\_gia**-**san\_giao\_dich**-**quy\_dinh\_san**-**chi\_tiet\_quy\_dinh\_san



Hình 4.6 Quan hệ giữa nha\_dau\_tu và co\_phieu



Hình 4.7Quan hệ giữa ngan\_hang**-**tai\_khoan\_ngan\_hang và lenh\_ung



Hình 4.8 Quan hệ giữa lenh\_dat và co\_phieu cùng với các bảng liên quan

### 4.2.2 Từ điển dữ liệu của Công ty chứng khoán

* Bảng : **quyen**

**quyen** (ma\_quyen, ten\_quyen)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_quyen | integer | Mã quyền |  | X |  | X |
| 2 | ten\_quyen | character varying(30) | Tên quyền | X |  |  | X |

* Bảng **tai\_khoan**

**tai\_khoan** (ma\_tai\_khoan, ten\_dang\_nhap, mat\_khau, ma\_nhan\_vien)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_tai\_khoan | UUID | Mã tài khoản | X | X |  | X |
| 2 | ten\_dang\_nhap | character varying(20) | Tên đăng nhập | X |  |  | X |
| 3 | mat\_khau | character varying(255) | Mật khẩu |  |  |  | X |
| 4 | ma\_nhan\_vien | character varying(30) | Mã nhân viên | X |  | X | X |

* Bảng **taikhoan\_quyen**

**taikhoan\_quyen** (ma\_tai\_khoan, ma\_quyen)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_tai\_khoan | UUID | Mã tài khoản |  | X | X | X |
| 2 | ma\_quyen | integer | Mã quyền |  | X | X | X |

* Bảng **nhan\_vien**

**nhan\_vien** (ma\_nhan\_vien, cccd, ho, ten, phai, email, phone)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_nhan\_vien | character varying(30) | Mã nhân viên |  | X |  | X |
| 2 | ho | character varying(50) | Họ nhân viên |  |  |  | X |
| 3 | ten | character varying(10) | Tên nhân viên |  |  |  | X |
| 4 | phai | boolean | Giới tính |  |  |  | X |
| 5 | phone | character varying(15) | Số điện thoại | X |  |  | X |
| 6 | email | character varying(50) | Email | X |  |  | X |
| 7 | cccd | character varying(13) | CMND | X |  |  | X |

* Bảng **quy\_dinh\_cong\_ty**

**quy\_dinh\_cong\_ty** (ma\_quy\_dinh, ten\_quy\_dinh)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_quy\_dinh | character varying(30) | Mã quy định |  | X |  | X |
| 2 | ten\_quy\_dinh | character varying(50) | Tên quy định |  |  |  | X |

* Bảng **chi\_tiet\_quy\_dinh\_cong\_ty**

**chi\_tiet\_quy\_dinh\_cong\_ty** (ma\_quy\_dinh, ngay\_ap\_dung, gia\_tri)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_quy\_dinh | character varying(30) | Mã quy định công ty |  | X | X | X |
| 2 | ngay\_ap\_dung | timestamp with timezone | Ngày áp dụng |  | X |  | X |
| 3 | gia\_tri | numeric | Giá trị |  |  |  | X |

* Bảng **trang\_thai\_lenh**

**trang\_thai\_lenh** (ma\_trang\_thai, ten\_trang\_thai)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_trang\_thai | character varying(10) | Mã trạng thái |  | X |  | X |
| 2 | ten\_trang\_thai | character varying(50) | Tên trạng thái | X |  |  | X |

* Bảng **lenh\_khop**

**lenh\_khop** (ma\_lenh\_khop, gia, so\_luong, thoi\_gian, ma\_lenh\_dat)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_lenh\_khop | UUID | Mã lệnh khớp |  | X |  | X |
| 2 | so\_luong | integer | Số lượng |  |  |  | X |
| 3 | gia | numeric | Giá |  |  |  | X |
| 4 | thoi\_gian | timestamp with time zone | Thời gian |  |  |  | X |
| 5 | ma\_lenh\_dat | UUID | Mã lệnh đặt |  |  | X | X |

* Bảng **loai\_lenh**

**loai\_lenh** (ma\_loai\_lenh, ten\_loai\_lenh)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_loai\_lenh | character varying(5) | Mã loại lệnh |  | X |  | X |
| 2 | ten\_loai\_lenh | character varying(50) | Tên loại lệnh |  |  |  | X |

* Bảng **lenh\_ung**

**lenh\_ung** (ma\_lenh\_ung, ngay\_yeu\_cau, ngay\_ban, phi\_ung, so\_tien, tai\_khoan\_ngan\_hang)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_lenh\_ung | UUID | Mã loại lệnh |  | X |  | X |
| 2 | tai\_khoan\_ngan\_hang | character varying(30) | Số tài khoản ngân hàng |  |  | X | X |
| 3 | ngay\_yeu\_cau | timestamp with time zone | Ngày yêu cầu |  |  |  | X |
| 4 | ngay\_ban | timestamp with time zone | Ngày bán |  |  |  | X |
| 5 | so\_tien | numeric | Số Tiền |  |  |  | X |
| 6 | phi\_ung | numeric | Phí Ứng |  |  |  | X |

* Bảng **ngan\_hang**

**ngan\_hang** (ma\_ngan\_hang, ten\_ngan\_hang)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_ngan\_hang | character varying(30) | Mã ngân hàng |  | X |  | X |
| 2 | ten\_ngan\_hang | character varying(50) | Tên ngân hàng | X |  |  | X |

* Bảng **tai\_khoan\_ngan\_hang**

**tai\_khoan\_ngan\_hang** (so\_tai\_khoan, so\_du,trang\_thai, ma\_nha\_dau\_tu, ngan\_hang)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | so\_tai\_khoan | character varying(30) | Số tài khoản |  | X |  | X |
| 2 | so\_du | numeric | Số dư |  |  |  | X |
| 3 | trang\_thai | boolean | Trạng thái hoạt động |  |  |  | X |
| 4 | ngan\_hang | character varying(30) | Mã ngân hàng |  |  | X | X |
| 5 | ma\_nha\_dau\_tu | character varying(10) | Mã nhà đầu tư |  |  | X | X |

* Bảng **lenh\_dat**

**lenh\_dat** (ma\_lenh\_dat, gia, loai\_giao\_dich, so\_luong, thoi\_gian, ma\_co\_phieu, ma\_tai\_khoan, ma\_loai\_lenh, ma\_trang\_thai)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_lenh\_dat | UUID | Mã lệnh đặt |  | X |  | X |
| 2 | loai\_giao\_dich | boolean | Loại giao dịch |  |  |  | X |
| 3 | so\_luong | integer | Số lượng |  |  |  | X |
| 4 | gia | numeric | Giá |  |  |  | X |
| 5 | thoi\_gian | timestamp with time zone | Thời gian |  |  |  | X |
| 6 | ma\_trang\_thai | character varying(10) | Mã trạng thái |  |  | X | X |
| 7 | ma\_loai\_lenh | character varying(5) | Mã loại lệnh |  |  | X | X |
| 8 | ma\_co\_phieu | character varying(10) | Mã cổ phiếu |  |  | X | X |
| 9 | ma\_tai\_khoan | character varying(30) | Số tài khoản ngân hàng |  |  | X | X |

* Bảng: **nha\_dau\_tu**

**nha\_dau\_tu** (ma\_nha\_dau\_tu, cccd, dia\_chi, email, ho, ten, mat\_khau\_dat\_lenh, mat\_khau\_giao\_dich, ngay\_cap\_cccd, ngay\_dang\_ky, ngay\_sinh, noi\_cap\_cccd, noi\_sinh, phai, phone, trang\_thai, xac\_thuc\_email)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SST | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not null |
| 1 | ma\_nha\_dau\_tu | character varying(10) | Mã nhà đầu tư |  | X |  | X |
| 2 | cccd | character varying(13) | Căn cước công dân | X |  |  |  |
| 3 | dia\_chi | character varying(255) |  |  |  |  |  |
| 4 | email | character varying(50) |  | X |  |  |  |
| 5 | ho | character varying(50) |  |  |  |  |  |
| 6 | ten | character varying(10) |  |  |  |  |  |
| 7 | ngay\_sinh | date |  |  |  |  |  |
| 8 | phai | boolean | Phái |  |  |  |  |
| 9 | phone | character varying(15) | Số điện thoại | X |  |  |  |
| 10 | mat\_khau\_dat\_lenh | character varying(255) | Mật khẩu đặt lệnh |  |  |  |  |
| 11 | mat\_khau\_giao\_dich | character varying(255) | Mật khẩu giao dịch |  |  |  |  |
| 12 | ngay\_cap\_cccd | date |  |  |  |  |  |
| 13 | ngay\_dang\_ky | date | Ngày đăng ký |  |  |  |  |
| 14 | noi\_cap\_cccd | character varying(100) |  |  |  |  |  |
| 15 | noi\_sinh | character varying(30) |  |  |  |  |  |
| 16 | trang\_thai | boolean | Trạng thái tài khoản |  |  |  |  |
| 17 | xac\_thuc\_email | boolean | Xác thực email |  |  |  |  |

* Bảng: **co\_phieu**

**co\_phieu** (ma\_co\_phieu, ten\_cong\_ty, dia\_chi, email, fax, phone, so\_luong, website)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_co\_phieu | character varying(10) | Mã cổ phiếu |  | X |  | X |
| 2 | ten\_cong\_ty | character varying(50) | Tên công ty |  |  |  |  |
| 3 | dia\_chi | character varying(255) | Địa Chỉ |  |  |  |  |
| 4 | email | character varying(50) |  |  |  |  |  |
| 5 | fax | character varying(15) | Số fax |  |  |  |  |
| 6 | phone | character varying(15) |  |  |  |  |  |
| 7 | website | character varying(100) |  |  |  |  |  |
| 8 | so\_luong | integer | Số lượng phát hành |  |  |  |  |

* Bảng: **so\_huu**

**so\_huu** (ma\_co\_phieu, ma\_nha\_dau\_tu, soluong)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_nha\_dau\_tu | character varying(10) | Mã nhà đầu tư |  | X | X | X |
| 2 | ma\_co\_phieu | character varying(10) | Mã chứng khoán |  | X | X | X |
| 3 | soluong | integer | Số lượng |  |  |  |  |

* Bảng: **yeu\_thich**

**yeu\_thich** (ma\_co\_phieu, ma\_nha\_dau\_tu)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_nha\_dau\_tu | character varying(10) | Mã nhà đầu tư |  | X | X | X |
| 2 | ma\_co\_phieu | character varying(10) | Mã chứng khoán |  | X | X | X |

* Bảng: **lich\_su\_gia**

**lich\_su\_gia** (ma\_co\_phieu, ngay, gia\_dong\_cua, gia\_mo\_cua, gia\_tham\_chieu, gia\_san, gia\_tran)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_co\_phieu | character varying(10) | Mã chứng khoán |  | X | X | X |
| 2 | ngay | date | Ngày hiệu lực |  | X |  | X |
| 3 | gia\_dong\_cua | numeric | Giá đóng cửa |  |  |  | X |
| 4 | gia\_mo\_cua | numeric | Giá mở cửa |  |  |  | X |
| 5 | gia\_san | numeric | Giá sàn |  |  |  | X |
| 6 | gia\_tham\_chieu | numeric | Giá tham chiếu |  |  |  | X |
| 7 | gia\_tran | numeric | Giá trần |  |  |  | X |

* Bảng: **san\_giao\_dich**

**san\_giao\_dich** (ma\_san, ten\_san)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_san | character varying(10) | Mã sàn |  | X |  | X |
| 2 | ten\_san | character varying(50) | Tên sàn | X |  |  |  |

* Bảng: **thong\_tin\_chuyen\_san**

**thong\_tin\_chuyen\_san** (ma\_co\_phieu, ma\_san, ngay\_chuyen\_san)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_san | character varying(10) | Mã sàn |  | X |  | X |
| 2 | ma\_co\_phieu | character varying(10) | Mã chứng khoán |  | X |  | X |
| 3 | ngay\_chuyen\_san | timestamp with time zone | Ngày chuyển sàn |  | X |  | X |

* Bảng: **quy\_dinh\_san**

**quy\_dinh\_san** (ma\_quy\_dinh, ten\_quy\_dinh)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhất | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_quy\_dinh | character varying(30) | Mã quy định sàn |  | X |  | X |
| 2 | ten\_quy\_dinh | character varying(50) | Tên quy định sàn | X |  |  |  |

* Bảng: **chi\_tiet\_quy\_dinh\_san**

**chi\_tiet\_quy\_dinh\_san** (ma\_quy\_dinh, ma\_san, ngay\_ap\_dung, gia\_tri)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu | Diễn giải | Khóa duy nhât | Khóa chính | Khóa ngoại | Not Null |
| 1 | ma\_san | character varying(10) | Mã sàn |  | X | X | X |
| 2 | ma\_quy\_dinh | character varying(30) | Mã quy định |  | X | X | X |
| 3 | ngay\_ap\_dung | timestamp with time zone | Ngày áp dụng |  | X |  | X |
| 4 | gia\_tri | numeric | Giá trị |  |  |  | X |

# CHƯƠNG TRÌNH

## 5.1 Tổng quan

### 5.1.1 Giao tiếp giữa hai Web service

Trong chương trình của chúng ta có 2 Web Service nằm ở 2 địa chỉ khác nhau. Đồng thời cả 2 đều truy cập đến 2 Database độc lập.

Giữa 2 Service sẽ giao tiếp với nhau thông qua API và trao đổi message thông qua một hệ thống trung gian là Kafka. Dữ liệu trong quá trình trao đổi được bảo mật thông qua cơ chế *Authorize và Authentication và SSL.*

### 5.1.2 Kiến trúc của Web Service

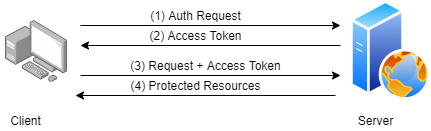
**Sử dụng kiến trúc phân lớp gồm có các lớp:**

* Lớp Interface trên cùng chịu trách nhiệm giao tiếp và xác thực các yêu cầu từ người dùng.
* Lớp Business chịu trách nhiệm xác định và điều phối các quy trình nghiệp vụ gồm nhiều bước và kéo dài. Những quy trình này phải được sắp xếp và thực hiện theo một thứ tự chính xác. Kiểm tra các quy tắc nghiệp vụ, ràng buộc logic và thực hiện các công việc.
* Lớp Data Access Layer chịu trách nhiệm chính lưu trữ và truy xuất dữ liệu từ các nguồn dữ liệu ([Data Sources](https://topdev.vn/blog/dung-gi-de-luu-tru-data-thay-the-local-storage/)). Hơn nữa còn tạo thuận lợi cho việc dễ cấu hình và bảo trì.

### 5.1.3 Authorize và Authentication

Người sử dụng giao tiếp với Web Service thông qua API. Vậy làm sao để xác định được thông tin người sử dụng và những tài nguyên nào mà người đó được phép truy cập.

Để giải quyết vấn đề này, chúng ta sẽ sử dụng một phương pháp bảo mật khá phổ biến đó là hệ thống sẽ xác định người sử dụng thông một đoạn mã được gọi là Token. Token được gắn vào Header mỗi khi gửi Request. Đoạn mã này chứa các thông tin đã được mã hóa gồm các thông tin cơ bản của người dùng và thời gian hết hạn. Khi mã hết hạn. Người dùng bắt buộc phải đăng nhập lại để lấy mã mới.

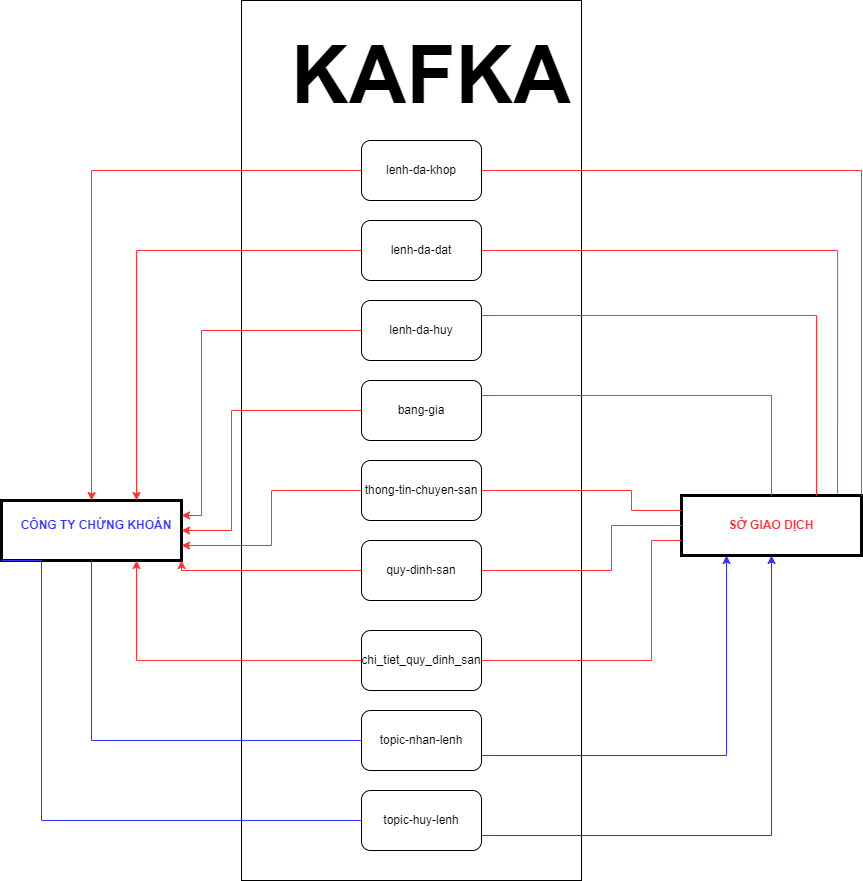


Hình 5.1 Mô hình Client-Server

### 5.1.4 Kiến trúc hệ thống Kafka

Bên cạnh sử dụng Web service để trao đổi dữ liệu trong quá trình vận hành, đa phần trong các chức năng của chương trình chúng ta đều sử dụng Kafka vì những điểm mạnh ưu việt như truyền dữ liệu nhanh, an toàn, khả năng chịu lỗi cao với hàng chục nghìn đến tận hàng triệu message ghi nhận cùng lúc dựa vào cấu hình của máy chủ Kafka vận hành.

Các topic chúng ta sử dụng trong quá trình giao tiếp giữa **Công ty chứng khoán** và **Sở giao dịch**, trong đó chiều mũi tên là chiều đi ra của message:



Hình 5.2 Mô hình thiết kế hệ thống Kafka giữa Công ty chứng khoán và Sở giao dịch

* Giải thích về chức năng của các topic:
* **lenh-da-khop:** producer là sở giao dịch, consumer là công ty chứng khoán, khi có một lệnh vừa được khớp thì lệnh khớp đó sẽ được gửi tới topic này. **lenh-da-dat:** producer là sở giao dịch, consumer là công ty chứng khoán, khi có một lệnh vừa được đặt mới thì lệnh đặt đó sẽ được gửi tới topic này.
* **lenh-da-huy**: producer là sở giao dịch, consumer là công ty chứng khoán, khi có một lệnh vừa bị hủy thì lệnh đặt bị hủy đó sẽ được gửi tới topic này.
* **bang-gia**: producer là sở giao dịch, consumer là công ty chứng khoán,khi thông tin của bảng giá trực tuyến có thay đổi, thông tin đó sẽ được gửi tới topic này.
* **thong-tin-chuyen-san**: producer là sở giao dịch, consumer là công ty chứng khoán, khi thông tin chuyển sàn của cổ phiếu có sự thay đổi, thì sự thay đổi đó sẽ được gửi tới topic này.
* **quy-dinh-san**: producer là sở giao dịch, consumer là công ty chứng khoán, khi quy định sàn có sự thay đổi, thì sự thay đổi đó sẽ được gửi tới topic này.
* **chi-tiet-quy-dinh-san:** producer là sở giao dịch, consumer là công ty chứng khoán, khi chi tiết quy định sàn có sự thay đổi, thì sự thay đổi đó sẽ được gửi tới topic này.
* **topic-nhan-lenh**: producer là công ty chứng khoán, consumer là sở giao dịch, mỗi khi có lệnh mới phát sinh tại công ty, thì công ty sẽ chuyển tiếp lệnh đó vào topic này để gửi lên sở giao dịch.
* **topic-huy-lenh**: producer là công ty chứng khoán, consumer là sở giao dịch, mỗi khi có lệnh hủy phát sinh tại công ty, thì công ty sẽ chuyển tiếp lệnh đó vào topic này để gửi lên sở giao dịch.

### 5.1.5 Bảo mật SSL cho Kafka

**SSL là viết tắt của từ Secure Sockets Layer. SSL là tiêu chuẩn của công nghệ bảo mật, truyền thông mã hoá giữa máy chủ Web server và trình duyệt.** Tiêu chuẩn này hoạt động và đảm bảo rằng các dữ liệu truyền tải giữa máy chủ và trình duyệt của người dùng đều riêng tư và toàn vẹn. SSL hiện tại cũng là tiêu chuẩn bảo mật cho hàng triệu website trên toàn thế giới, nó bảo vệ dữ liệu truyền đi trên môi trường internet được an toàn.

**SSL** đảm bảo rằng tất cả các dữ liệu được truyền giữa các máy chủ web và các trình duyệt được mang tính riêng tư, tách rời.

Trong chương trình của mình, em đã sử dụng **SSL** để đảm bảo bảo mật trong quá trình truyền message giữa **Công ty chứng khoán** và **Sở giao dịch** thông qua Kafka.

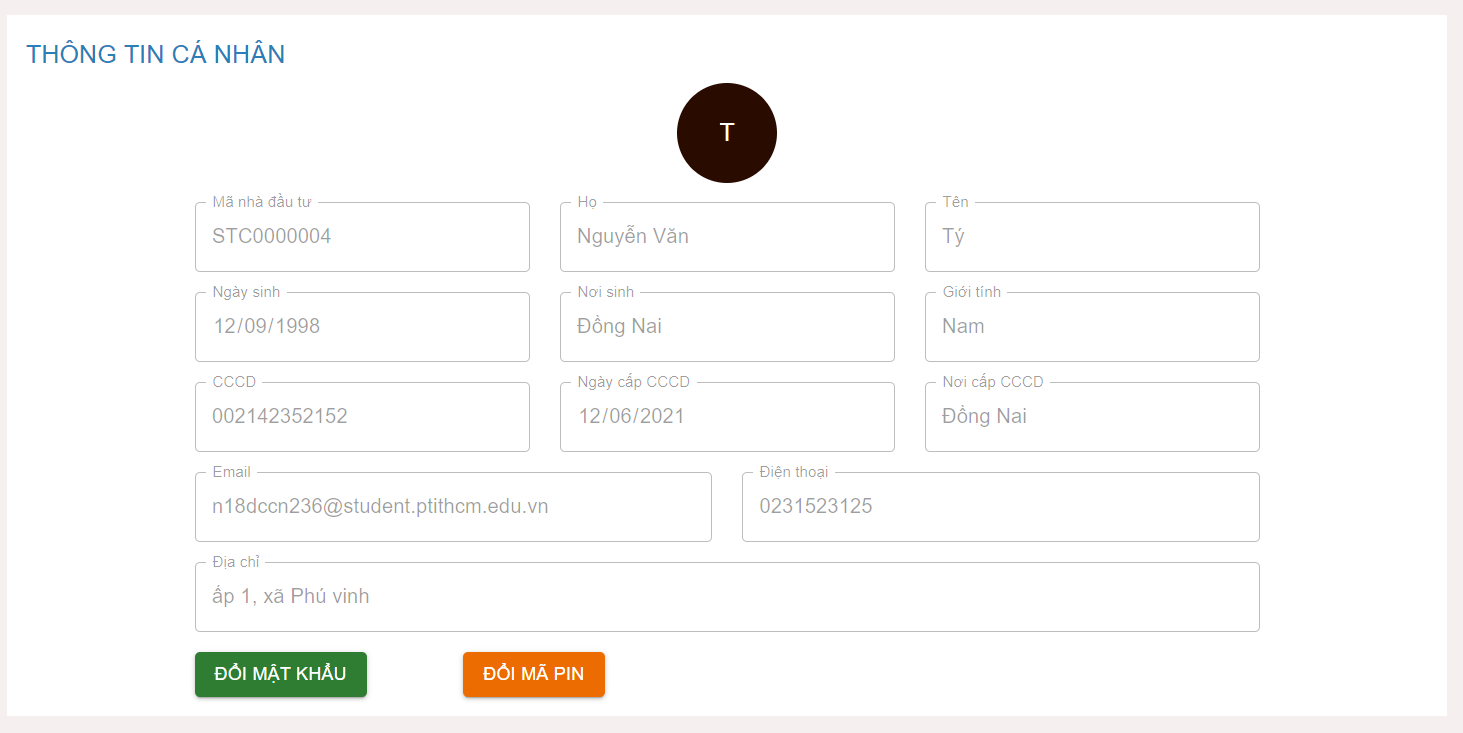
### 5.1.6 Dữ liệu cập nhật tức thời

Phần quan trọng nhất trong hệ thống của chúng ta là đặt lệnh để có thể mua bán cổ phiếu. Xem chính xác và ngay lập tức sự biến động giá là rất cần thiết và quan trọng để nhà đầu tư đưa ra các quyết định mua bán kịp thời.

Để đáp ứng được yêu cầu trên ta cần phải thiết kế đường truyền WebSocket, mỗi khi có sự thay dổi về lệnh đặt, thông tin về giá,… tất cả thông tin sẽ được truyền từ Sở giao dịch về công ty chứng khoán, và từ công ty chứng khoán chúng ta sẽ truyền dữ liệu nhận được qua bên ngoài thông qua WebSocket.

## 5.2 Thiết kế giao diện cho nhà đầu tư

* **Xem thông tin cá nhân, đổi mật khẩu và đổi mã PIN**

****

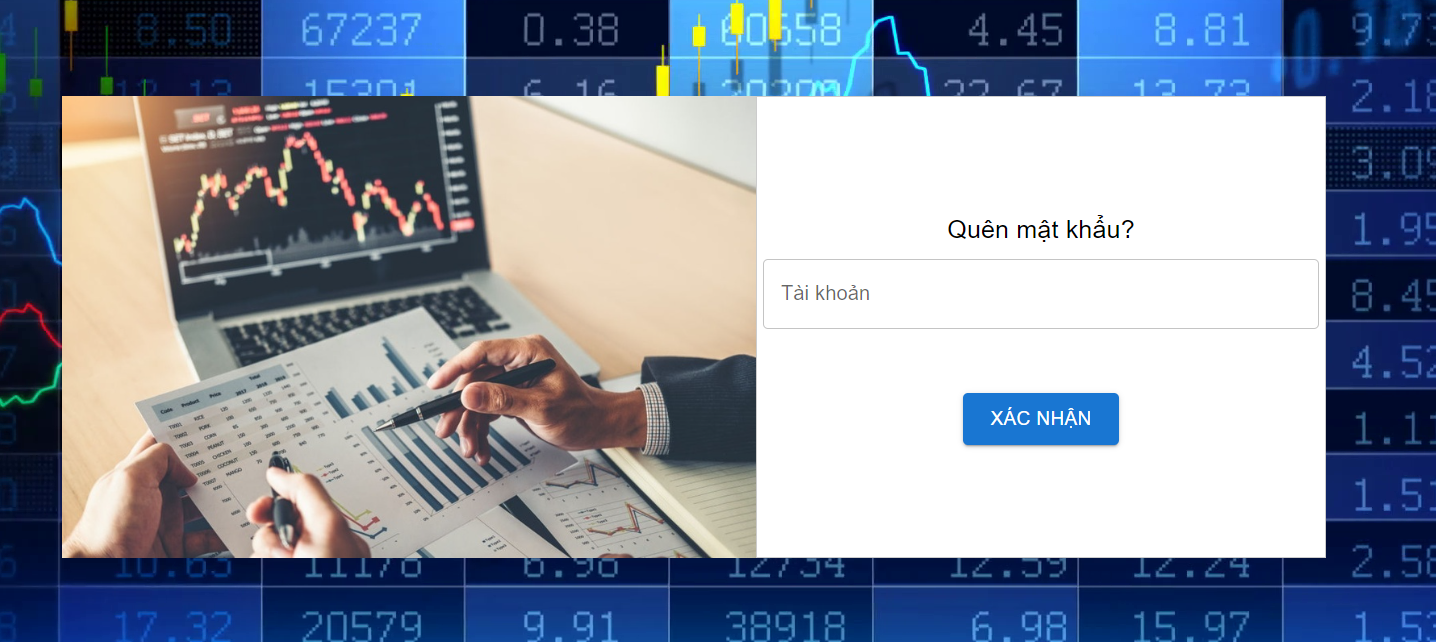
Hình 5.3 Xem thông tin cá nhân, đổi mật khẩu/mã PIN

Nếu như nhà đầu tư tạo thành khoản thành công và được phê duyệt, thông tin tài khoản đăng nhập bao gồm Mã nhà đầu tư, Mật khẩu đặt lệnh(gồm 6 ký tự chữ số) và Mật khẩu giao dịch(gồm 8 ký tự chữ số) được gửi về email nhà đầu tư.

Sau khi đăng nhập thành công, nhà đầu tư hoàn toàn có thể kiểm tra lại thông tin của nhân của chính mình, ngoài ra còn có khả năng thay đổi mật khẩu giao dịch và mã PIN.

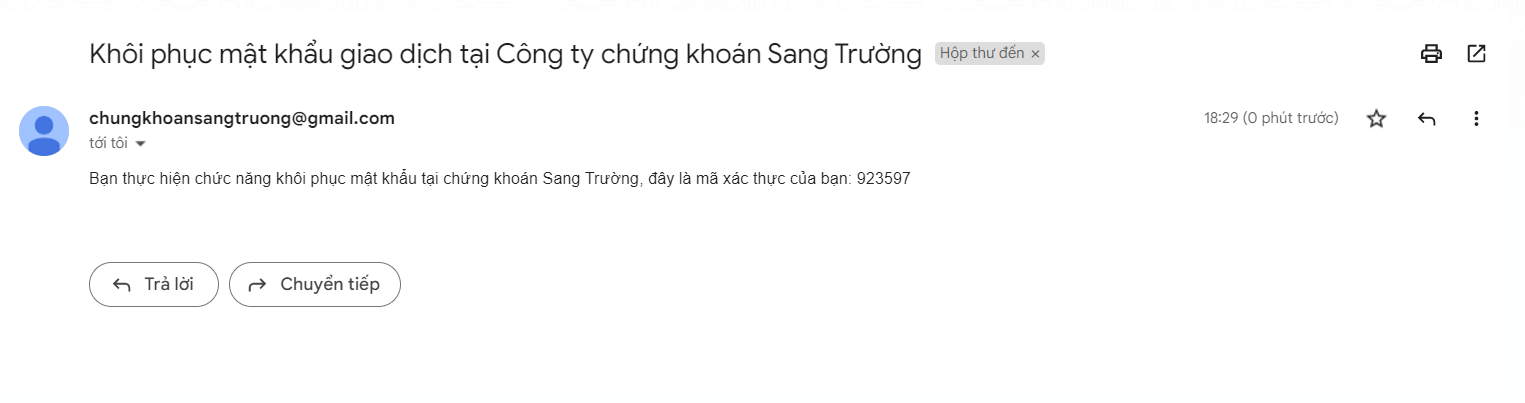
* **Quên mật khẩu**

Việc người dùng quên mật khẩu đăng nhập vào chương trình là một điều diễn ra rất thường xuyên. Để hỗ trợ cho nhà đầu tư lấy lại mật khẩu truy cập, nhà đầu tư chọn vào mục **Quên mật khẩu tại trang đăng nhập,** sau đó nhà dầu tư nhập vào mã nhà đầu tư cần khôi phục.



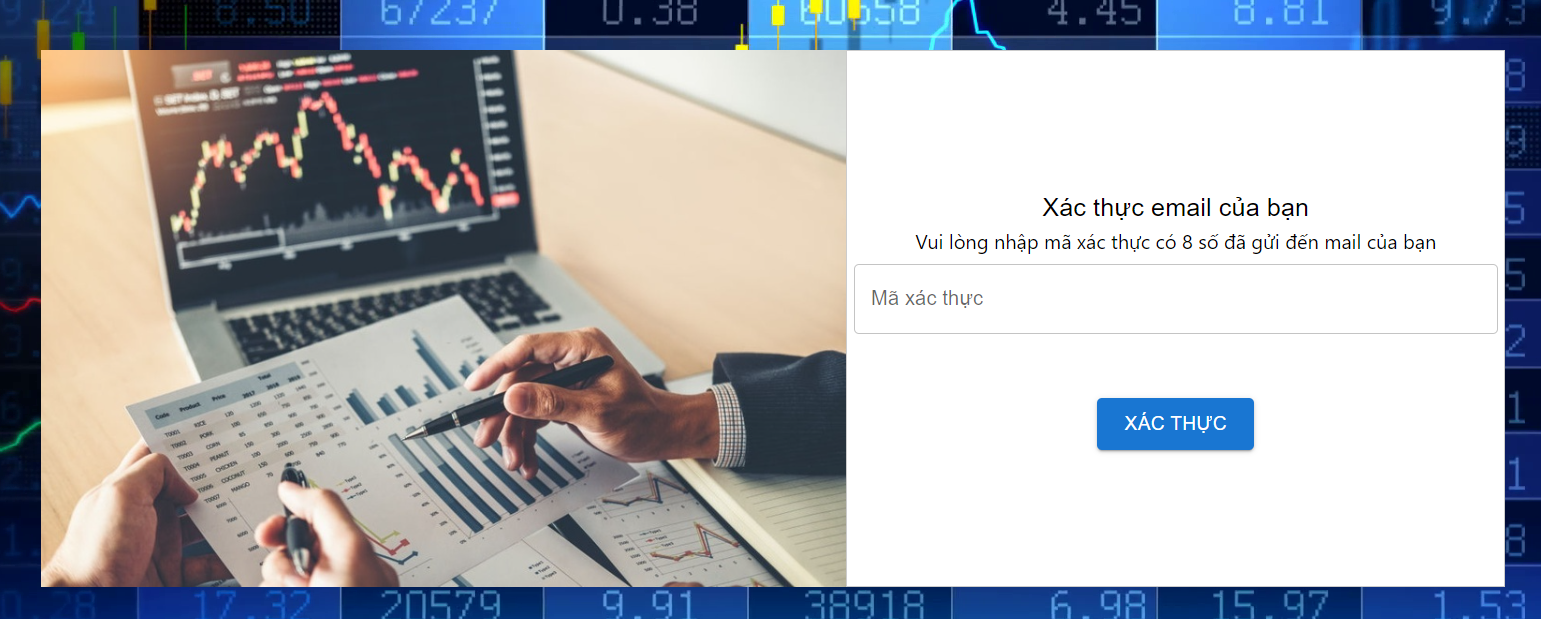
Hình 5.4 Chọn chức năng quên mật khẩu

Khi đó một email khôi phục sẽ được công ty tự động gửi tới email đăng ký ban đầu của nhà đầu tư.

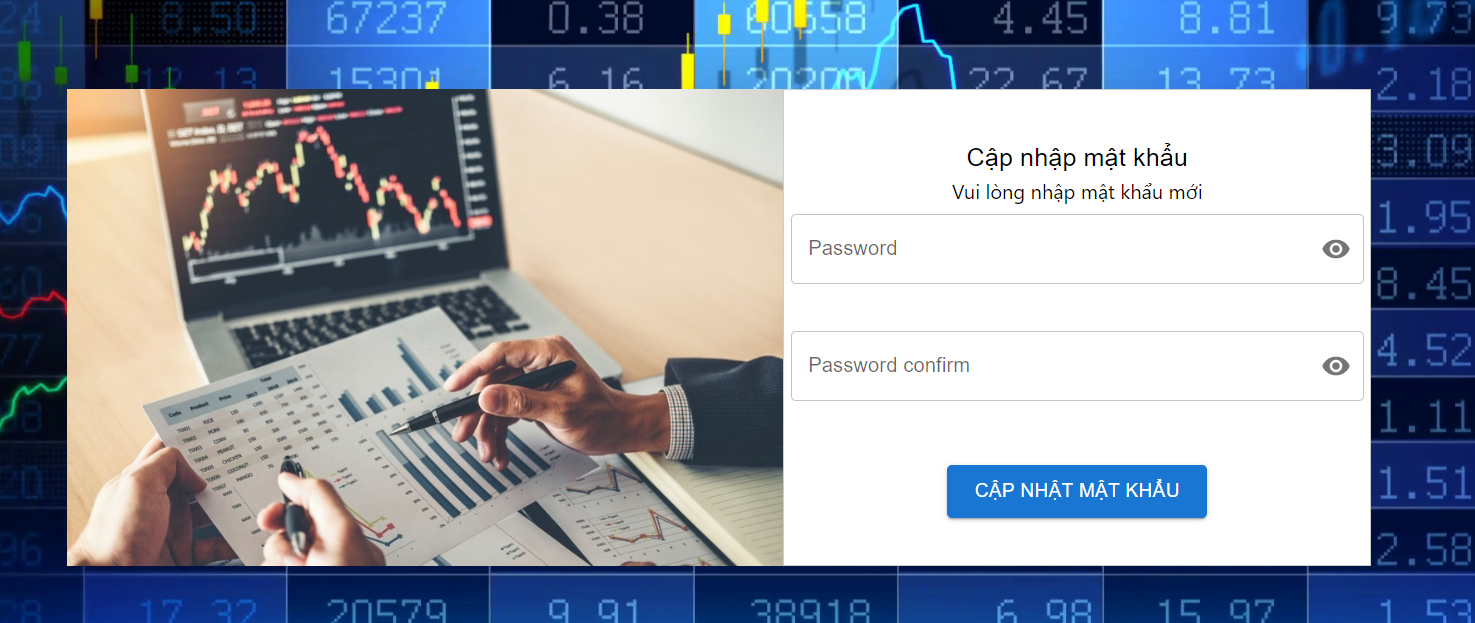


Hình 5.5 Email gửi mã xác thực khôi phục tài khoản

Nhà đầu tư nhập chính xác mã khôi phục được gửi thông qua email, trong trường hợp xác thực thành công, nhà đầu tư tiến hành cập nhật mật khẩu mới:



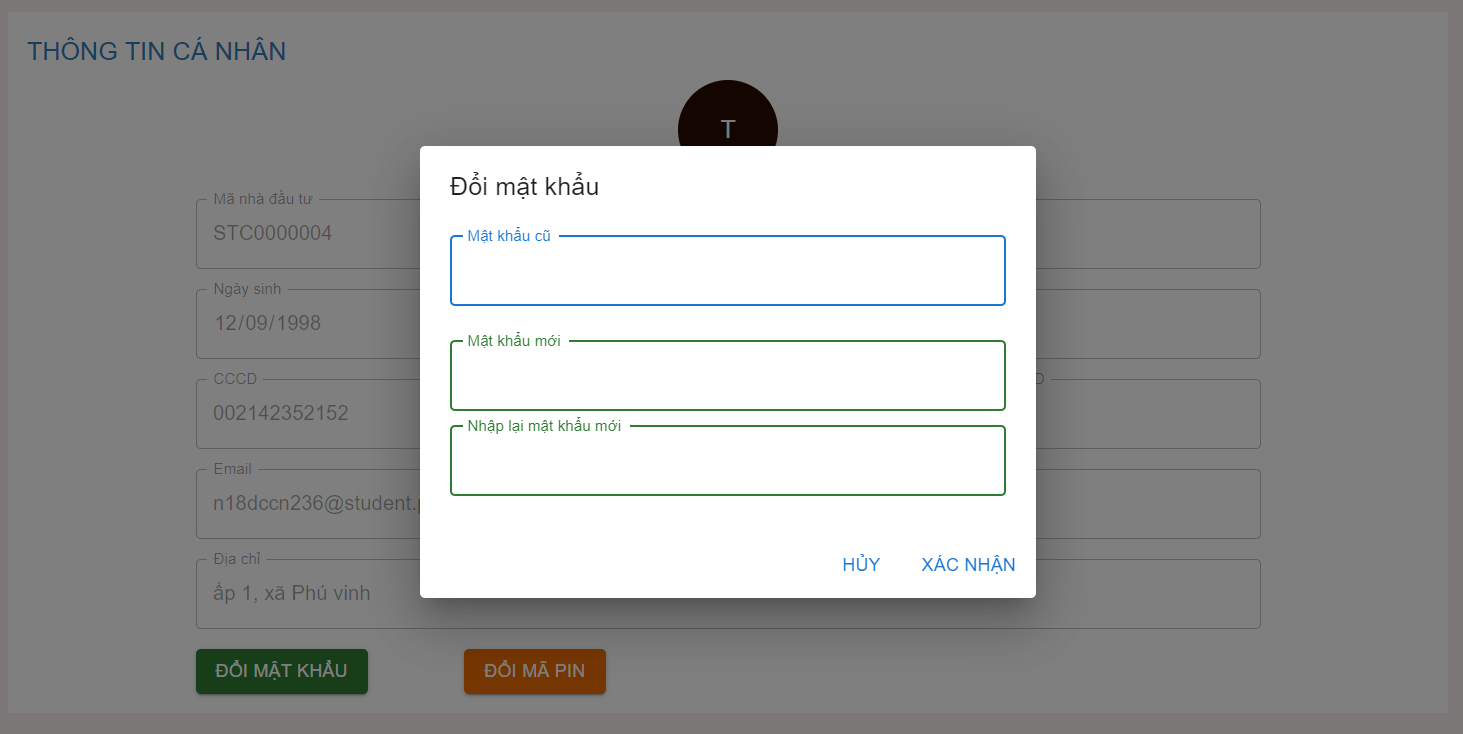
Hình 5.6 Nhập mã xác thực để tiền hành cập nhật mật khẩu mới



Hình 5.7 Cập nhật mật khẩu mới cho lần đăng nhập kế tiếp

* **Đổi mật khẩu**

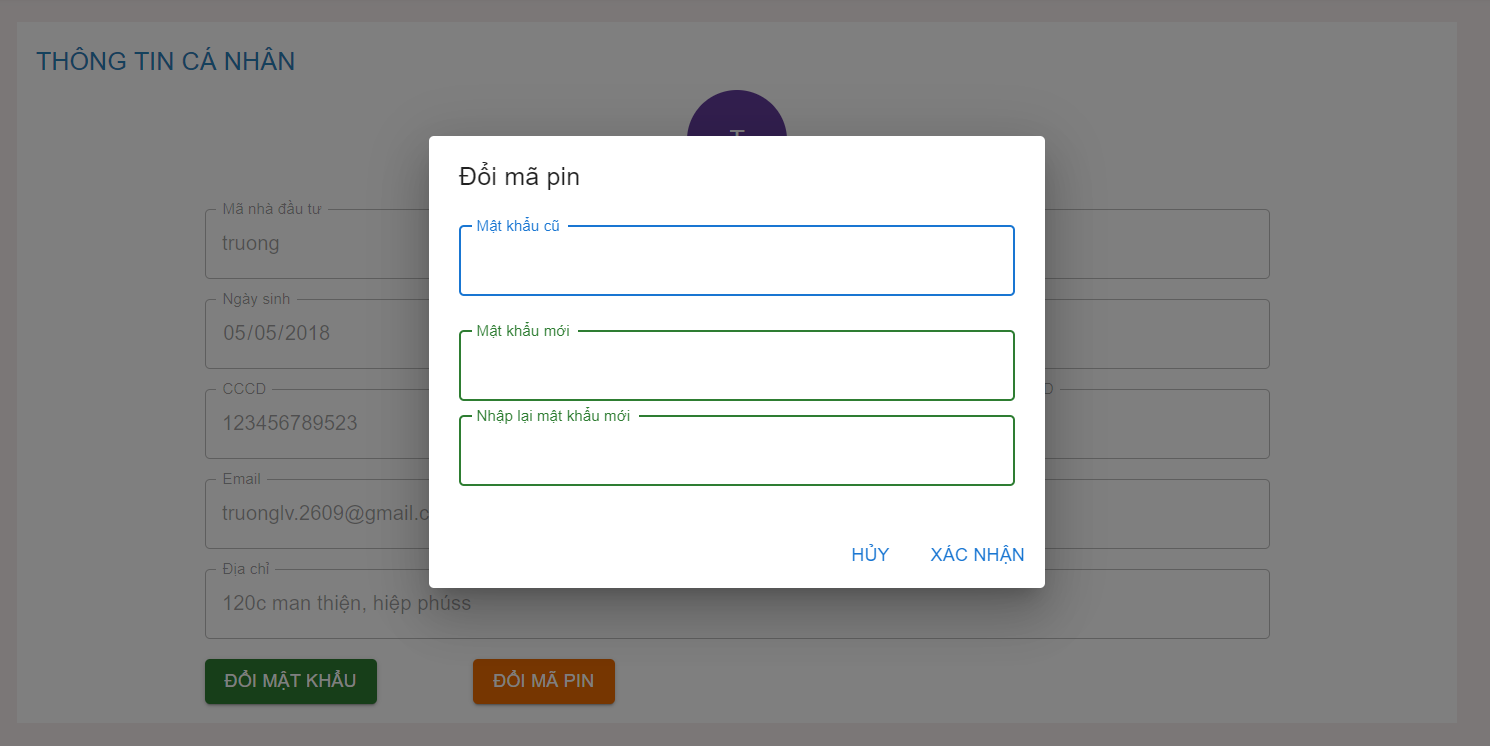
Đổi mật khẩu nhằm mục đích thay đổi mật khẩu trong khi đăng nhập vào trang web, gia tăng tính bảo mật theo thời gian, đồng thời giúp nhà đầu tư dễ nhớ, thuận tiện trong quá trình giao dịch.



Hình 5.8 Cập nhật mật khẩu mới

* **Đổi mã PIN**

Đổi mã PIN nhằm mục đích thay đổi mật khẩu trong khi đặt lệnh , gia tăng tính bảo mật theo thời gian, đồng thời giúp nhà đầu tư dễ nhớ, thuận tiện trong quá trình giao dịch.



Hình 5.9 Cập nhật mã PIN mới

* **Sao kê lệnh đặt trong ngày.**

Trong trang này chỉ hiển thị thông tin những lệnh được đặt trong ngày hôm nay.

Các thông tin hiển thị bao gồm:

* Mã cổ phiếu mà nhà đầu tư đã đặt.
* Loại giao dịch là Mua hoặc Bán.
* Loại lệnh mà nhà đầu tư đã dùng đặt lệnh.
* Tài khoản chính là tài khoản ngân hàng thực hiện lệnh đặt.
* Thời gian đặt lệnh.
* Khối lượng là số lượng cổ phiếu đã đặt lệnh.
* Giá chính là giá mà nhà đầu tư ban đầu nhập vào.
* Khối lượng khớp chính là khối lượng cổ phiếu đã khớp với lệnh đặt.
* Giá khớp đối với lệnh mua thì có thể nhỏ hơn hoặc bằng giá đặt. Còn đối với lệnh bán thì lớn hơn hoặc bằng.
* Giá trị khớp bằng khối lượng khớp nhân với giá khớp.
* Trạng thái bao gồm: **Khớp Hết, Chờ Khớp, Khớp 1 Phần, Đã Hủy.**
* Hủy: hủy lệnh đối với các lệnh đang chờ khớp.



Hình 5.10 Sao kê lệnh đặt trong ngày

* **Sao kê lệnh trong một khoảng thời gian**

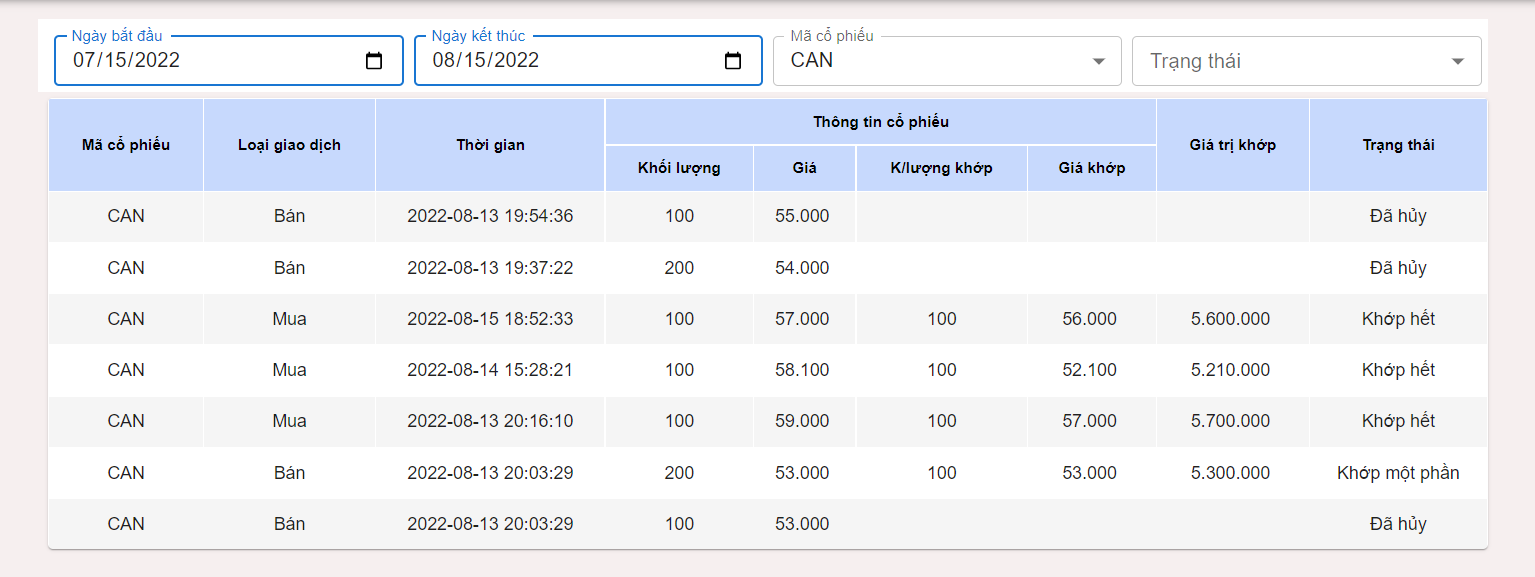
Nhà đầu tư có thể sao kê lệnh khớp/lệnh đặt trong 1 khoảng thời gian, mặc định sẽ lọc những lệnh đặt trong vòng 1 tuần gần nhất theo các trạng thái lệnh khác nhau cùng với tính năng tìm kiếm theo mã cổ phiếu.

Các thông tin hiển thị bao gồm:

* Mã cổ phiếu mà nhà đầu tư đã đặt.
* Loại giao dịch là Mua hoặc Bán.
* Thời gian đặt lệnh.
* Khối lượng chính là số lượng cổ phiếu nhà đầu tư đã đặt.
* Giá chính là giá mà nhà đầu tư đặt lệnh.
* Khối lượng khớp lệnh là số lượng cổ phiếu đã giao dịch thành công.
* Giá khớp đối với lệnh mua thì có thể nhỏ hơn hoặc bằng giá đặt. Còn đối với lệnh bán thì lớn hơn hoặc bằng.
* Giá trị khớp bằng khối lượng khớp nhân với giá khớp.

Trạng thái bao gồm: **Khớp Hết, Khớp 1 Phần** đối với lệnh khớp.

Trạng thái bao gồm : **Khớp Hết, Chờ Khớp, Khớp 1 Phần, Đã Hủy,Gửi lệnh** đối với lệnh đặt**.**



Hình 5.11 Sao kê lệnh đặt/lệnh khớp trong một khoảng thời gian

* **Quy tắc số dư T+2**

Để tránh khó hiểu trong những phần tiếp theo, trong phần này sẽ tập trung giải thích về T+2.

Thị trường chứng khoán Việt Nam đã phát triển được 22 năm (từ 28/7/2000). Theo thống kê đến nay có 6.1 triệu tài khoản được mở và đang giao dịch. Số lệnh giao dịch chứng khoán hằng ngày lên tới con số hàng trăm ngàn.

Cũng như hệ thống giao dịch của ngân hàng hay một số ngành khác, hệ thống giao dịch chứng khoán cũng được xử lý bằng hệ thống máy chủ. Tuy nhiên, việc vận hành một khối lượng giao dịch rất lớn như trên không hề đơn giản. Việc phát sinh các lỗi kỹ thuật trong giao dịch nếu không được xử lý kịp thời sẽ ảnh hưởng cực kỳ lớn tới thị trường. Và việc sửa chữa, xử lý cần có thời gian. Đây chính là lý do chúng ta áp dụng chu kỳ thanh toán ngày T+2.

Tóm lại, chu kỳ thanh toán ngày T+2 được quy định nhằm tạo khoảng trống về thời gian để khắc phục sự cố xảy ra để đảm bảo thị trường được vận hành thông suốt.

Nếu là lệnh mua thì khi khớp lệnh, cổ phiếu sẽ được chuyển về khối lượng trong T2. Nếu là lệnh bán thì tiền sẽ được chuyển vào số dư trong T2. Khối lượng và số dư trong T2 sẽ được chuyển về tài khoản chính thông qua từng ngày chuyển là T2 về T1, T1 về T0 và từ T0 về tài khoản chính.

Phiên giao dịch chứng khoán của một ngày sẽ bắt đầu từ 9 giờ đến 14 giờ 45 mỗi ngày trừ thứ bảy, chủ nhật và các ngày lễ theo quy định.

Trong khoảng thời gian này nhà đầu tư có thể thực hiện giao dịch mua cổ phiếu trên sàn giao dịch chứng khoán. Khi các lệnh đặt là mua được khớp với nhau hoặc (liên tục) thì ngay lập tức số lượng cổ phiếu mua được sẽ chạy vào trong số dư T+2. Cổ phiếu sẽ về vào lúc 8 giờ sáng trước phiên giao dịch bắt đầu 1 tiếng thì lúc đó số dư T+2 sẽ tự động cập nhật số lượng vào số dư T+1 của tài khoản. Và tiếp tục như thế bắt đầu một phiên giao dịch thì cổ phiếu của dư T+1 sẽ chuyển vào T+0 rồi từ T+0 sẽ chuyển vào cổ phiếu hiện có.

Như vậy sau khi giao dịch mua thành công cố phiếu thì sẽ mất khoản 3 ngày để cổ phiếu mua được từ số dư T2 chạy vào cổ phiếu hiện có thông qua T1 và T0 thời gian chính xác là 8 giờ sáng khi phiên giao dịch bắt đầu mở cửa.

Nếu nhà đầu tư thực hiện lệnh mua cổ phiếu vào phiên giao dịch cuối cùng của ngày là thứ 6 thì phải chờ đến 8 giờ sáng thứ 4 thì số lượng cổ phiếu mới về tới tài khoản vì sàn chứng khoán sẽ đóng cửa vào Thứ bảy và Chủ nhật.

Polygon

Description automatically generated with medium confidence

Hình 5.12 Quy tắc số dư T2 dành cho cổ phiếu

Phiên giao dịch chứng khoán của một ngày sẽ bắt đầu từ 9 giờ sáng đến 14 giờ 45 ngày trừ thứ bảy, chủ nhật và các ngày lễ theo quy định.

Thì trong khoảng thời gian này nhà đầu tư khi thực hiện giao dịch bán cổ phiếu hiện có của mình. Trong phiên giao dịch khi các lệnh đặt được khớp với nhau (hoặc liên tục) thì ngay lập tức tiền cổ phiếu bán được sẽ chạy vào trong số dư T2. Sau 15 giờ chiều thì số dư T2 sẽ tự động cập nhật số tiền vào số dư T1 của tài khoản(trừ số tiền bán cổ phiếu thành công trong ngày hôm nay). Và tiếp tục như thế hết một phiên giao dịch thì tiền của dư T1 sẽ chuyển vào T0 rồi từ T0 sẽ chuyển vào tài khoản.

Như vậy sau khi giao dịch bán thành công cố phiếu thì sẽ mất khoản 3 ngày để tiền từ số dư T2 chạy vào tài khoản thông qua T1 và T0.

Nếu nhà đầu tư thực hiện lệnh khớp vào phiên giao dịch cuối cùng là ngày thứ 6 thì phải chờ đến 15 giờ chiều thứ 4 mới về tới tài khoản vì sàn chứng khoán sẽ đóng cửa vào Thứ bảy và Chủ nhật.

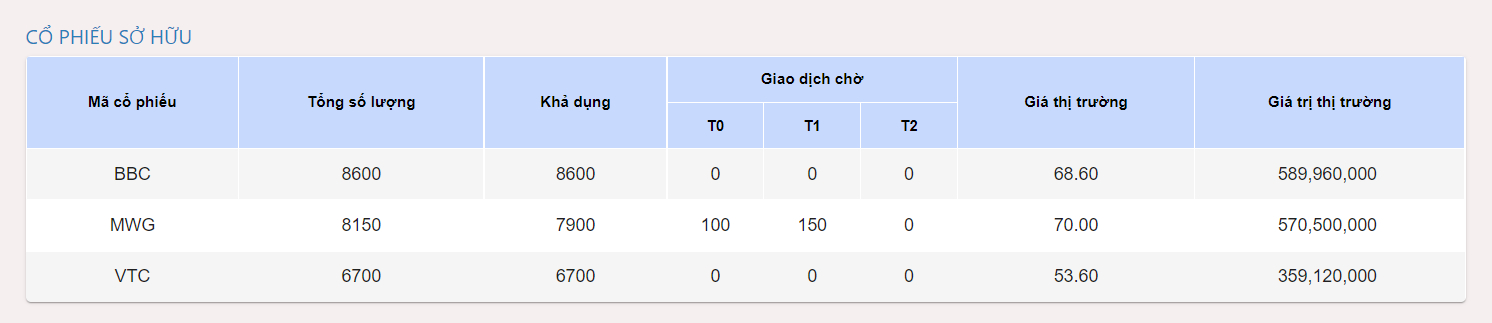
A picture containing polygon

Description automatically generated

Hình 5.13 Quy tắc số dư T2 dành cho Tài khoản ngân hàng

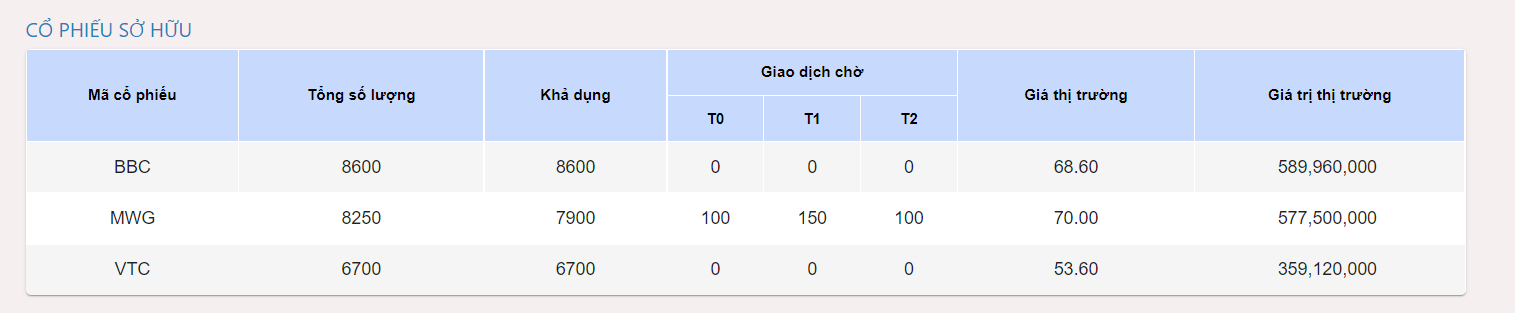
* **Minh họa quy trình cập nhật số dư cổ phiếu T2 cho nhà đầu tư:**

Sao kê số dư cổ phiếu T2 của nhà đầu tư trước khi lệnh mua được khớp:



Hình 5.14 Số dư cổ phiếu T2 ban đầu của nhà đầu tư

Khi lệnh mua của nhà đầu tư được khớp thành công, số lượng cổ phiếu ngay lập tức được chuyển về số dư T2. Ví dụ bên dưới cho thấy số lượng T2 của cổ phiếu MWG tăng thêm 100 cổ phiếu.



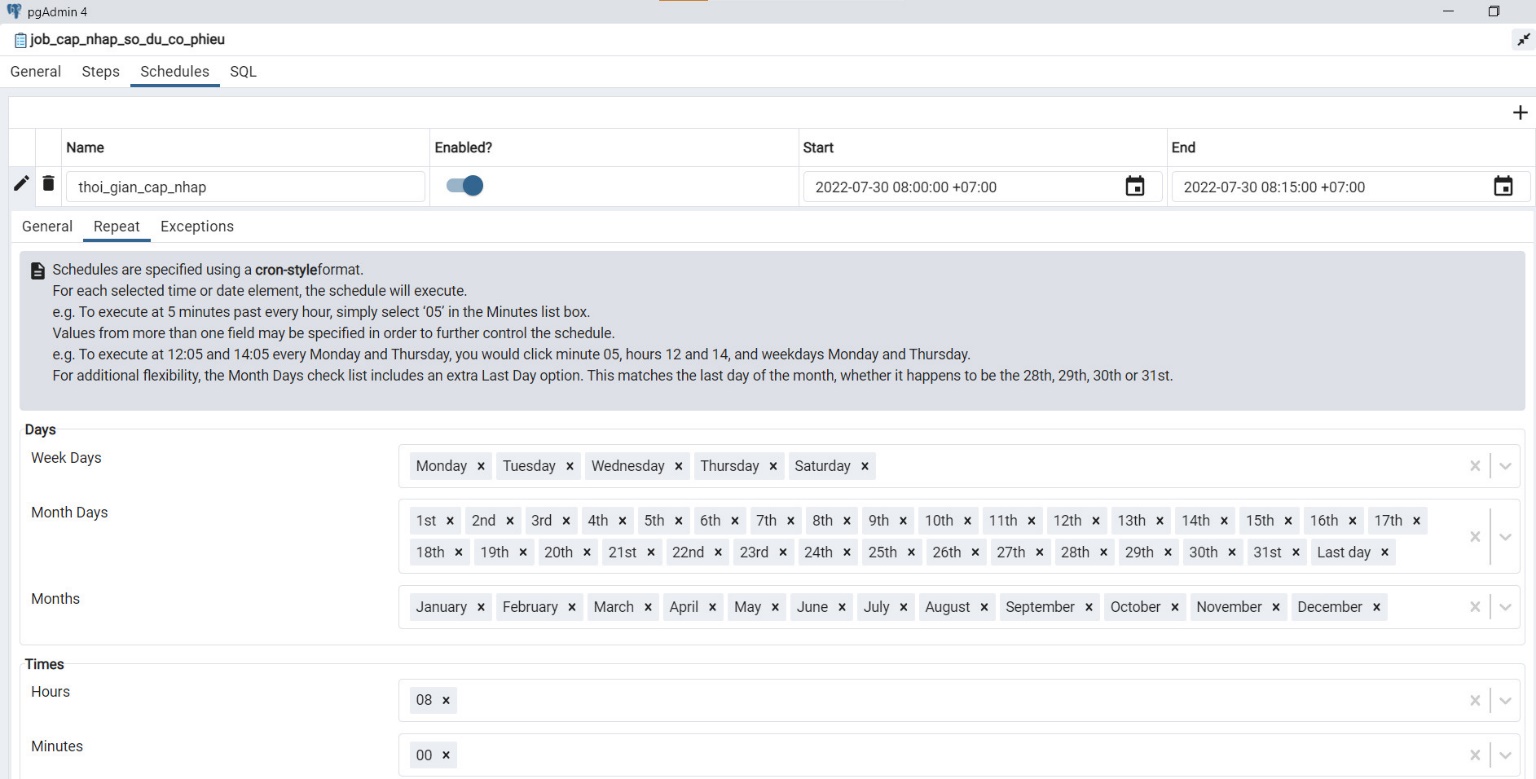
Hình 5.15 Số dư T2 của cổ phiếu MWG tăng lên 100 cổ phiếu

Đúng vào lúc 8h sáng phiên giao dịch ngày hôm sau(trừ ngày thứ bảy và ngày chủ nhật), thì số dư cổ phiếu T0 chuyển về số dư khả dụng, số dư cổ phiếu T1 chuyển vào T0, số dư cổ phiếu T2 chuyển vào T1.



Hình 5.16 Số dư cổ phiếu được cập nhật vào lúc 8h sáng trước khi phiên giao dịch bắt đầu

Để chương trình có thể cập nhật các số dư cổ phiếu một cách tự động và chính xác, ta tạo một Job trên PostgreSQL, job này sẽ chạy tự động vào lúc 8h sáng hàng ngày(trừ thứ bảy và chủ nhật):



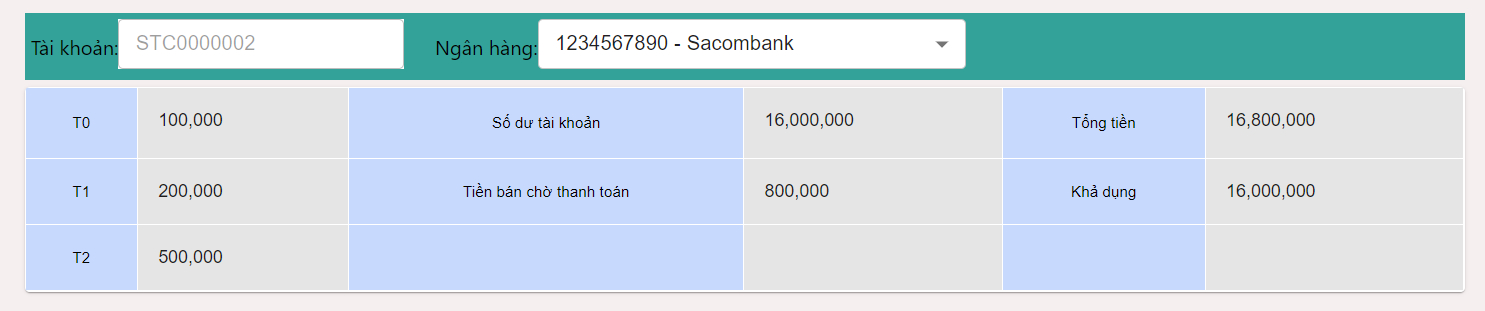
Hình 5.17 Cấu hình job trên PostgreSQL chạy vào 8h sáng các ngày có phiên giao dịch

* StoreProducer phục vụ Job **Cập nhật số dư cổ phiếu theo chu kỳ T2**

|  |
| --- |
| CREATE OR REPLACE PROCEDURE public.job\_cap\_nhap\_co\_phieu(  )  LANGUAGE 'plpgsql'  AS $BODY$  declare  v\_MANDT varchar;  v\_MACP varchar;  v\_SOLUONG INT;  v\_SOLUONGT0 INT;  v\_SOLUONGT1 INT;  v\_SOLUONGT2 INT;  T2CUR CURSOR FOR SELECT ma\_nha\_dau\_tu, ma\_co\_phieu, so\_luong\_t0, so\_luong\_t1, so\_luong\_t2 FROM sohuut2;  begin  open T2CUR;  loop  FETCH NEXT FROM T2CUR INTO v\_MANDT, v\_MACP, v\_SOLUONGT0, v\_SOLUONGT1, v\_SOLUONGT2;  exit when not found;  SELECT soluong INTO v\_SOLUONG FROM so\_huu WHERE ma\_nha\_dau\_tu = v\_MANDT AND ma\_co\_phieu = v\_MACP;  UPDATE so\_huu SET soluong = v\_SOLUONG + COALESCE (v\_SOLUONGT0, 0)  WHERE ma\_nha\_dau\_tu = v\_MANDT AND ma\_co\_phieu = v\_MACP;  UPDATE sohuut2 SET so\_luong\_t0 = v\_SOLUONGT1, so\_luong\_t1 = v\_SOLUONGT2, so\_luong\_t2 = 0  WHERE ma\_nha\_dau\_tu = v\_MANDT AND ma\_co\_phieu = v\_MACP;  end loop;  CLOSE T2CUR;  end;  $BODY$; |

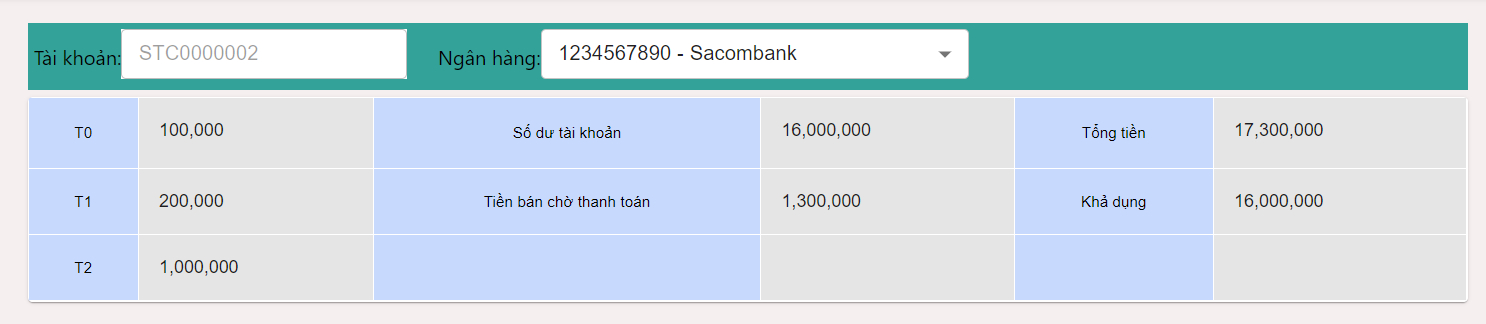
* **Minh họa quy trình cập nhật số dư tài khoản ngân hàng T2 cho nhà đầu tư:**

Sao kê số dư tiền T2 của nhà đầu tư trước khi lệnh bán được khớp:



Hình 5.18 Số dư tài khoản ngân hàng ban đầu

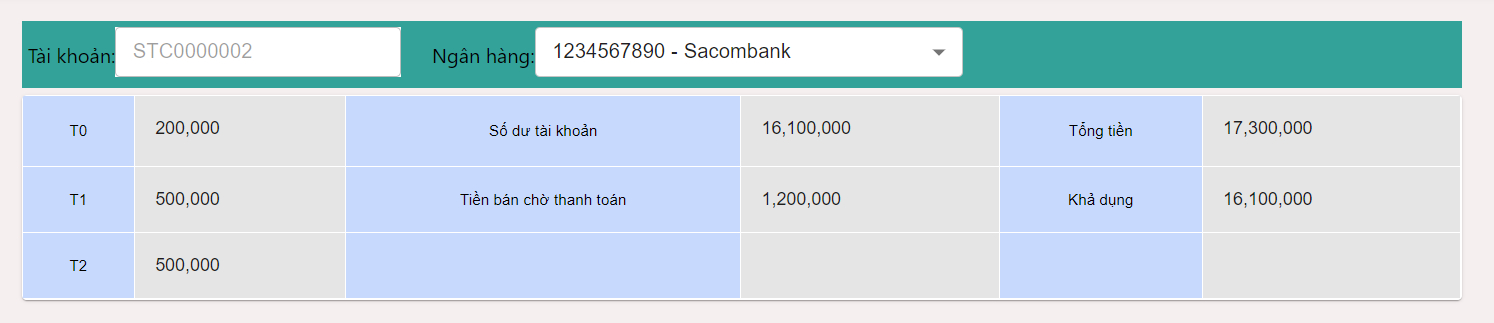
Khi lệnh bán của nhà đầu tư được khớp thành công, số tiến bán được từ cổ phiếu ngay lập tức được chuyển về số dư T2. Ví dụ bên dưới cho thấy số tiền được khớp từ lệnh bán chuyển về T2, tiền tại T2 từ **500.000** tăng lên thành **1.000.000**.



Hình 5.19 Số dư T2 được cập nhật sau khi khớp lệnh

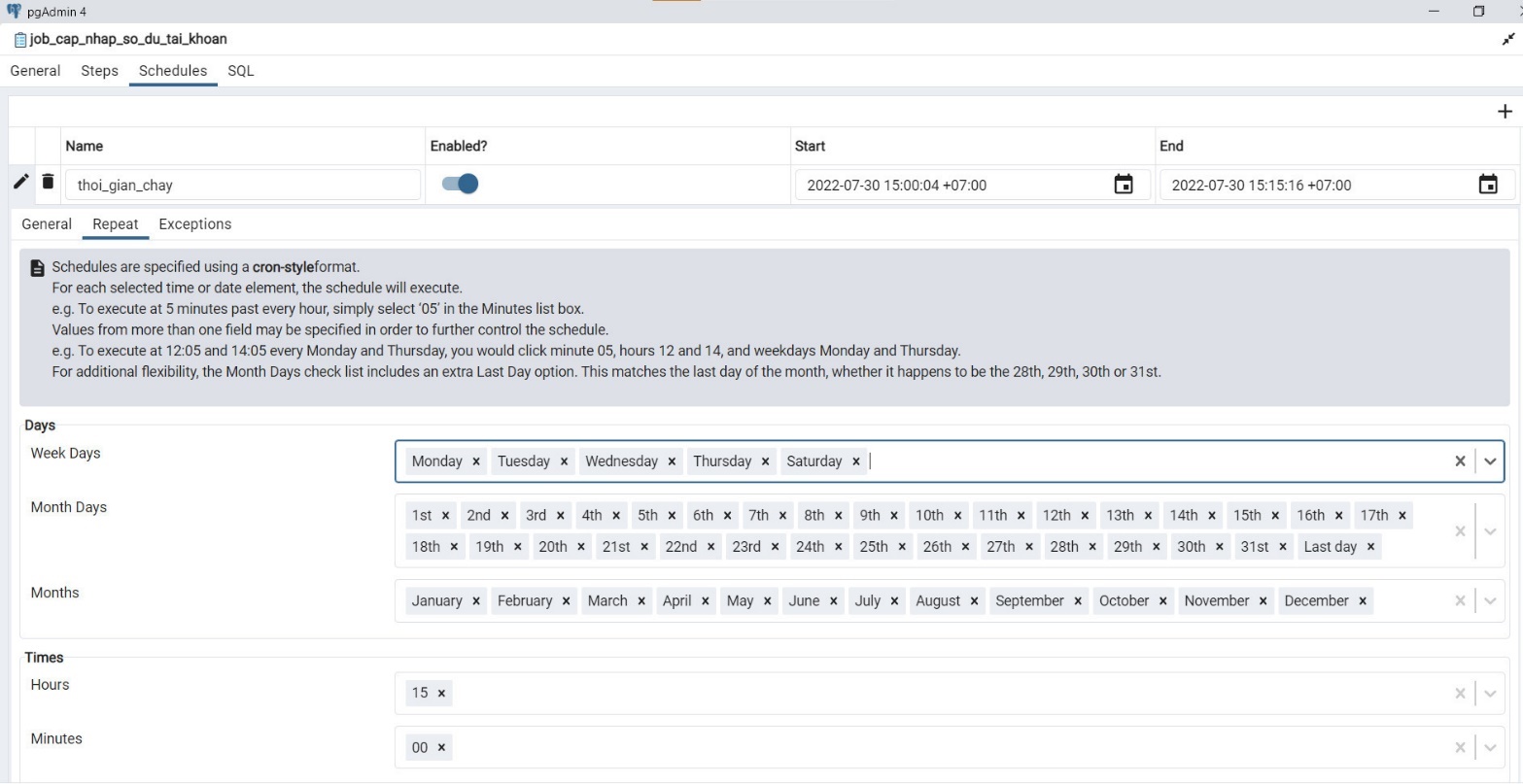
Đúng vào lúc 15h chiều hằng ngày(trừ ngày thứ bảy và ngày chủ nhật), thì số dư tiền T0 chuyển về số dư khả dụng, số dư tiền T1 chuyển vào T0, số dư tiền T2 chuyển vào T1.

Tuy nhiên, có một điều đặc biệt, số tiền bán được cổ phiếu của ngày hôm nay mặc dù nằm tại T2 nhưng sẽ không chuyển ngay về T1 vào lúc 15h, mà phải đợi sang 15h của ngày giao dịch kế tiếp. Ví dụ, số dư T2 từ ngày giao dịch hôm trước là **500.000**, sau khi khớp lệnh bán được số tiền **500.000** thì số dư T2 được cập nhật thành **1.000.000** và nhà đầu tư có thể theo dõi được điều này. Vào lúc 15h, chỉ có số tiền **500.000** của ngày giao dịch hôm trước được chuyển về số dư T1, số tiền **500.000** nhận được khi khớp lệnh trong ngày giao dịch hôm nay vẫn còn ở số dư T2.



Hình 5.20 Số dư T2 được cập nhật sau 15h

Để chương trình có thể cập nhật các số dư tiền một cách tự động và chính xác, ta tạo một Job trên PostgreSQL, job này sẽ chạy tự động vào lúc 15h chiều hàng ngày(trừ thứ bảy và chủ nhật):



Hình 5.21 Cấu hình job trên PostgreSQL chạy vào 15h các ngày có phiên giao dịch

* StoreProducer phục vụ Job **Cập nhật số dư tài khoản ngân hàng theo chu kỳ T2**

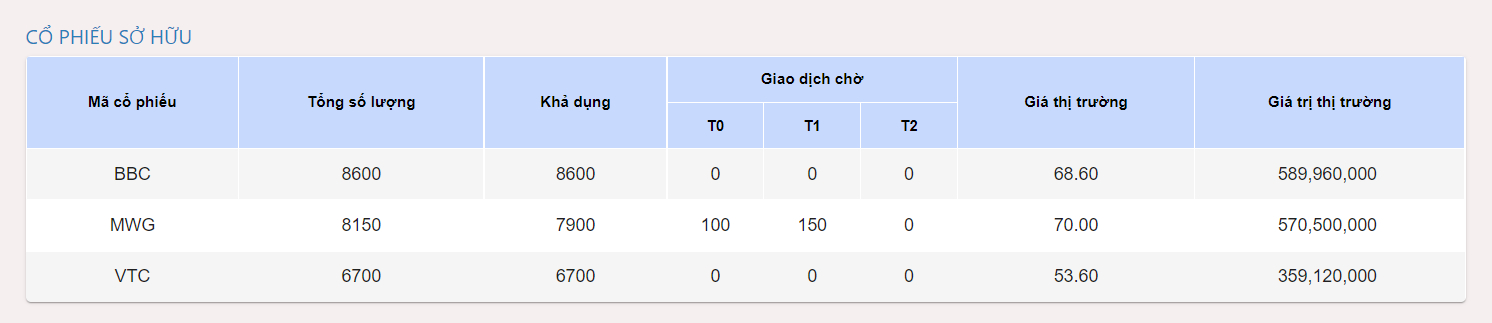
|  |
| --- |
| CREATE OR REPLACE PROCEDURE public.job\_cap\_nhap\_tien(  )  LANGUAGE 'plpgsql'  AS $BODY$  declare  v\_STK varchar;  v\_SODU numeric;  v\_SODUT0 numeric;  v\_SODUT1 numeric;  v\_SODUT2 numeric;  v\_SODUT3 numeric;  T2CUR CURSOR FOR SELECT so\_tai\_khoan, so\_du\_t0, so\_du\_t1, so\_du\_t2, so\_du\_t3 FROM taikhoannganhangt2;  begin  open T2CUR;  loop  FETCH NEXT FROM T2CUR INTO v\_STK, v\_SODUT0, v\_SODUT1, v\_SODUT2, v\_SODUT3;  exit when not found;  SELECT so\_du INTO v\_SODU FROM tai\_khoan\_ngan\_hang WHERE so\_tai\_khoan = v\_STK;  UPDATE tai\_khoan\_ngan\_hang SET so\_du = v\_SODU + COALESCE(v\_SODUT0, 0 ) WHERE so\_tai\_khoan = v\_STK;  UPDATE taikhoannganhangt2 SET so\_du\_t0 = v\_SODUT1,so\_du\_t1 = v\_SODUT2,so\_du\_t2 = v\_SODUT3, so\_du\_t3=0 WHERE so\_tai\_khoan = v\_STK;  end loop;  CLOSE T2CUR;  end;  $BODY$; |

* **Xem những cổ phiếu đang sở hữu**

Hiển thị mã cổ phiếu mà nhà đầu tư đó đang nắm giữ.

Các thông tin hiển thị bao gồm:

* Mã cổ phiếu.
* Tổng số lượng cổ phiếu
* Số cổ phiếu khả dụng tức là những cổ phiếu có thể giao dịch được.
* Cổ phiếu T0, T1, T2 là những cổ phiếu đang trong quá trình chờ về theo T(n) vào 8h hằng ngày không tính thứ 7 và chủ nhật. Trong đó n số ngày chờ để nhận cổ phiếu tính từ ngày hôm đó. Nghĩa là số lượng trong T0 ngày mai ta có thể nhận được.
* Giá thị trường chính là giá khớp mới nhất ngay trên bảng điện.
* Giá trị thị trường chính bằng Tổng số lượng nhân với Giá thị trường.

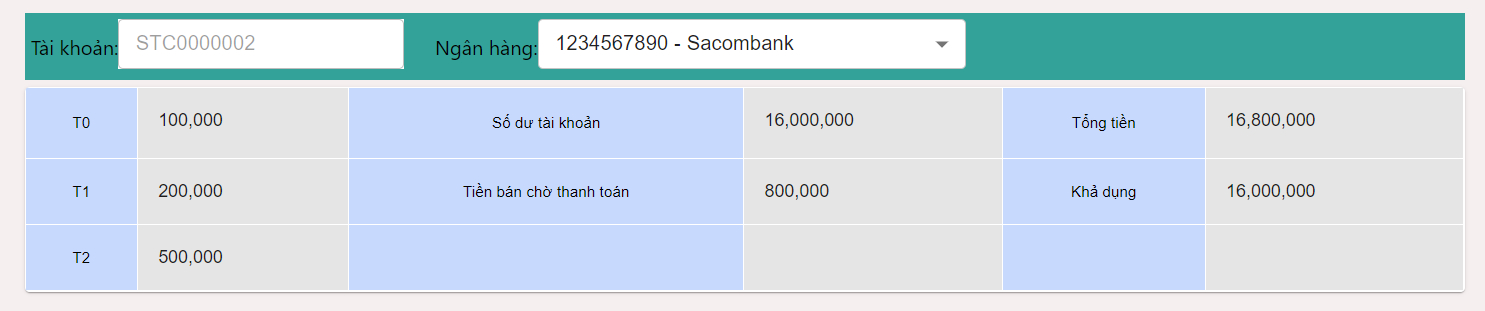


Hình 5.22 Thông tin cổ phiểu đang sở hữu

* **Sao kê thông tin tài khoản ngân hàng**

Hiển thị tất cả các tài khoản ngân hàng của nhà đầu tư. Những tài khoản này đã được xác thực và đăng ký giao dịch tại công ty.

Chúng ta thay đổi xem thông tin từng tài khoản bằng cách thay đổi mục chọn trên drop box.



Hình 5.23 Sao kê thông tin tài khoản ngân hàng

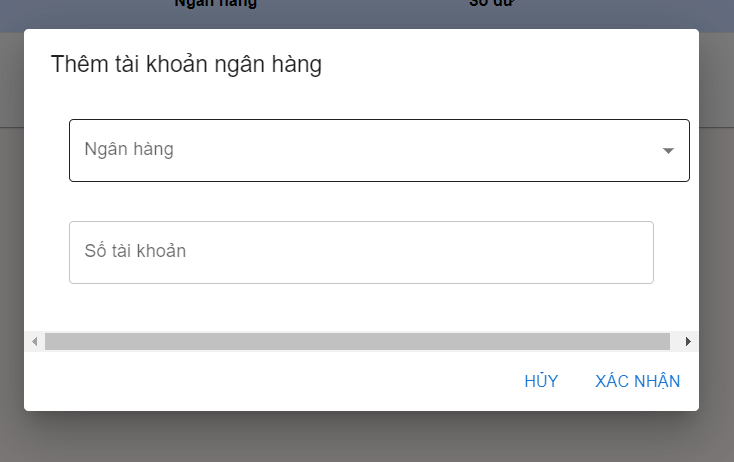
* **Quản lý tài khoản ngân hàng**

Nhà đầu tư có thể chủ động quản lý danh sách các tài khoản ngân hàng của mình bằng cách thêm tài khoản mới hoặc xóa tài khoản ngân hàng không nằm trong kế hoạch sử dụng nữa, miễn sao tài khoản ngân hàng được thêm mới là hợp lệ, nằm trong danh sách các ngân hàng mà công ty cho phép liên kết đồng thời chưa từng được sử dụng bởi bất kỳ nhà dầu tư nào trước đây.



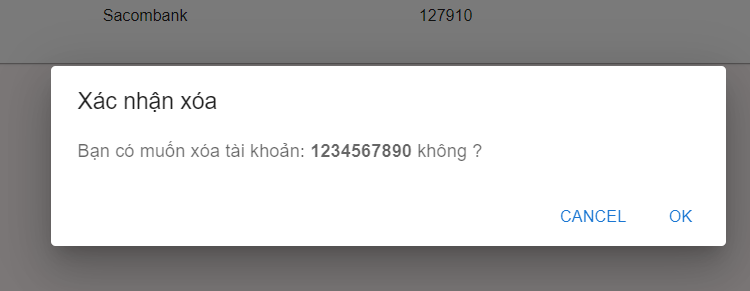
Hình 5.24 Quản lý danh sách tài khoản ngân hàng

* Thực hiện thêm tài khoản ngân hàng:



Hình 5.25 Thêm tài khoản ngân hàng mới

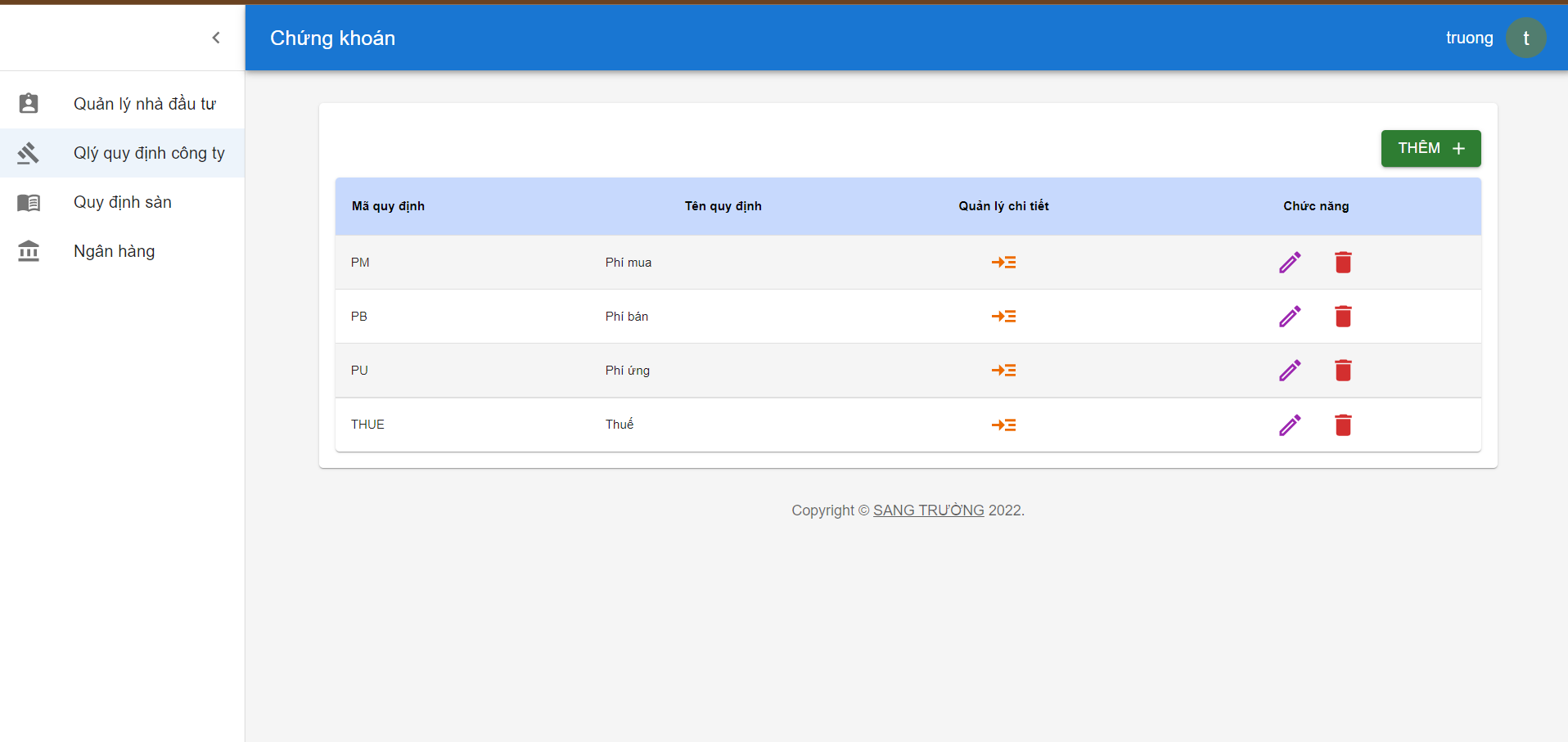
* Thực hiện chức năng hủy lên kết với tài khoản ngân hàng:



Hình 5.26 Hủy liên kết tài khoản ngân hàng

## 5.3 Thiết kế giao diện quản trị

* **Quản lý quy định công ty**

****

Hình 5.27: Trang quản lý quy định công ty

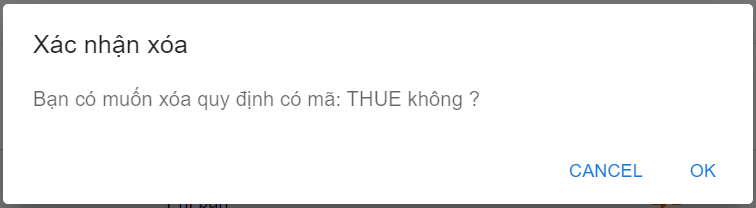
* ***Thêm/Sửa quy định***
  + Khi nhấp nút  (Thêm) hoặc nút  (Sửa) chương trình sẽ show popup lên để nhập các thông tin cần thiết về quy định

****

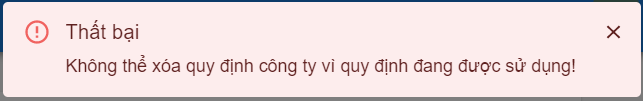
* + Các thông tin không được để trống:

****

* ***Xóa quy định***
  + Khi chọn xóa một quy định bất kỳ, thì sẽ show form để xác nhận xóa, tránh trường hợp nhấn nhầm

****

* + Chỉ xóa được những quy định chưa được áp dụng trong chi tiết quy định:

****

# KẾT LUẬN

## 6.1 Kết quả đạt được và hạn chế

* **Kết quả đạt được:**
* Xây dựng được cơ sở dữ liệu cho website công ty chứng khoán.
* Xây dựng website giao dịch cổ phiếu với những chức năng cơ bản và đơn giản nhất:
  + Quản lý thông quá trình sao kê các lệnh đặt/lệnh khớp theo khoảng thời gian.
  + Các chức năng quản lý thông tin tài khoản nhà đầu tư.
  + Xây dựng hệ thống Kafka là trung gian liên lạc giữa Sở giao dịch và Công ty chứng khoán.
  + Thông báo email về các hoạt động liên quan đến tài khoản nhà đầu tư như đăng ký, thay đổi mật khẩu, hỗ trợ xác thực.
  + Chương trình sử dụng Java SpringBoot để xây dựng phần server, HTML/CSS/JS-ReactJS để thiết kế phần giao diện.
* Xây dựng trang quản trị cho nhân viên nhằm mục đích điều hành các hoạt động cơ bản trong công ty.

## 6.2 Hướng phát triển

* Trên cơ sở kế thừa những gì đã đạt được từ chương trình hiện tại, tiến hành xây dựng chương trình bổ sung thêm các chức năng như Ứng tiền, Backup restore dữ liệu, phát triển hỗ trợ nền tảng mobile.

# PHỤ LỤC

* Function Sao kê **Số dư tài khoản ngân hàng chu kỳ T2**

|  |
| --- |
| CREATE OR REPLACE FUNCTION public.uft\_sao\_ke\_tai\_khoan\_ngan\_hang(  ma\_ndt character varying)  RETURNS TABLE(so\_tai\_khoan character varying, so\_du numeric, so\_du\_kha\_dung numeric, sodut0 numeric, sodut1 numeric, sodut2 numeric, banchothanhtoan numeric, tongsotien numeric, ten\_ngan\_hang character varying)  LANGUAGE 'plpgsql'  COST 100  VOLATILE PARALLEL UNSAFE  ROWS 1000  AS $BODY$  BEGIN  RETURN QUERY SELECT  D.so\_tai\_khoan,  D.so\_du+ D.SOTIENDATLENH,  D.so\_du as so\_du\_kha\_dung,  D.SODUT0,  D.SODUT1,  D.SODUT2,  D.BANCHOTHANHTOAN,  D.so\_du+ D.SOTIENDATLENH + D.BANCHOTHANHTOAN as tongsotien,  D.ten\_ngan\_hang  FROM  (SELECT tai\_khoan\_ngan\_hang.so\_tai\_khoan,  tai\_khoan\_ngan\_hang.ma\_nha\_dau\_tu,  tai\_khoan\_ngan\_hang.so\_du,  SUM(COALESCE(lenh\_dat.so\_luong,0)\*COALESCE(lenh\_dat.gia,0)) as SOTIENDATLENH,  COALESCE(taikhoannganhangt2.so\_du\_t0, 0) SODUT0,  COALESCE(taikhoannganhangt2.so\_du\_t1, 0) SODUT1,  COALESCE(taikhoannganhangt2.so\_du\_t2, 0) + COALESCE(taikhoannganhangt2.so\_du\_t3, 0) SODUT2,  COALESCE(taikhoannganhangt2.so\_du\_t0, 0) + COALESCE(taikhoannganhangt2.so\_du\_t1, 0) + COALESCE(taikhoannganhangt2.so\_du\_t2, 0) +COALESCE(taikhoannganhangt2.so\_du\_t3, 0)BANCHOTHANHTOAN,  ngan\_hang.ma\_ngan\_hang,  ngan\_hang.ten\_ngan\_hang  FROM tai\_khoan\_ngan\_hang  LEFT JOIN taikhoannganhangt2  ON tai\_khoan\_ngan\_hang.so\_tai\_khoan = taikhoannganhangt2.so\_tai\_khoan  LEFT JOIN lenh\_dat  on tai\_khoan\_ngan\_hang.so\_tai\_khoan = lenh\_dat.ma\_tai\_khoan and (lenh\_dat.ma\_trang\_thai<>'KH' AND lenh\_dat.ma\_trang\_thai<>'DH')  LEFT JOIN ngan\_hang  on tai\_khoan\_ngan\_hang.ngan\_hang = ngan\_hang.ma\_ngan\_hang  group by tai\_khoan\_ngan\_hang.so\_tai\_khoan,SODUT0,SODUT1,SODUT2,BANCHOTHANHTOAN,ngan\_hang.ma\_ngan\_hang) AS D  where D.ma\_nha\_dau\_tu = ma\_ndt;  END;  $BODY$; |

* Function **Sao kê cổ phiếu chu kỳ T2**

|  |
| --- |
| CREATE OR REPLACE FUNCTION public.uft\_sao\_ke\_co\_phieu(  ma\_ndt character varying)  RETURNS TABLE(ma\_co\_phieu character varying, so\_luong integer, tongso bigint, soluongt0 integer, soluongt1 integer, soluongt2 integer)  LANGUAGE 'plpgsql'  COST 100  VOLATILE PARALLEL UNSAFE  ROWS 1000  AS $BODY$  BEGIN  RETURN QUERY SELECT  S.ma\_co\_phieu,  S.soluong,  S.TONGSO + COALESCE(S.SOLUONGDAT,0) AS tongso,  S.SOLUONGT0,  S.SOLUONGT1,  S.SOLUONGT2  FROM (SELECT so\_huu.ma\_co\_phieu,  so\_huu.soluong,  COALESCE(so\_huu.soluong, 0) + COALESCE(SOHUUT2.so\_luong\_t0, 0) + COALESCE(SOHUUT2.so\_luong\_t1, 0) + COALESCE(SOHUUT2.so\_luong\_t2, 0)  AS TONGSO,  SUM(lenh\_dat.so\_luong) SOLUONGDAT,  COALESCE(SOHUUT2.so\_luong\_t0, 0) SOLUONGT0,  COALESCE(SOHUUT2.so\_luong\_t1, 0) SOLUONGT1,  COALESCE(SOHUUT2.so\_luong\_t2, 0) SOLUONGT2  FROM so\_huu  LEFT JOIN SOHUUT2  ON so\_huu.ma\_co\_phieu = SOHUUT2.ma\_co\_phieu  AND so\_huu.ma\_nha\_dau\_tu = SOHUUT2.ma\_nha\_dau\_tu  LEFT JOIN tai\_khoan\_ngan\_hang  ON so\_huu.ma\_nha\_dau\_tu = tai\_khoan\_ngan\_hang.ma\_nha\_dau\_tu  LEFT JOIN lenh\_dat  on (tai\_khoan\_ngan\_hang.so\_tai\_khoan = lenh\_dat.ma\_tai\_khoan  and lenh\_dat.ma\_trang\_thai<>'KH' AND lenh\_dat.ma\_trang\_thai<>'DH') and lenh\_dat.ma\_co\_phieu =so\_huu.ma\_co\_phieu  WHERE so\_huu.ma\_nha\_dau\_tu = ma\_ndt  group by so\_huu.ma\_co\_phieu,so\_huu.soluong,sohuut2.so\_luong\_t0,sohuut2.so\_luong\_t1,sohuut2.so\_luong\_t2) AS S;    END;  $BODY$;  ALTER FUNCTION public.uft\_sao\_ke\_co\_phieu(character varying)  OWNER TO postgres; |

* Function **Sao kê lệnh đặt trong ngày**

|  |
| --- |
| CREATE OR REPLACE FUNCTION public.udf\_lenh\_dat\_trong\_ngay(  ma\_ndt character varying)  RETURNS TABLE(id bigint, ma\_co\_phieu character varying, ma\_lenh\_dat character varying, slkhop integer, soluong bigint, thoi\_gian timestamp with time zone, ma\_trang\_thai character varying, ma\_loai\_lenh character varying, gia numeric, giakhop numeric, loai\_giao\_dich boolean, ma\_tai\_khoan character varying, ten\_trang\_thai character varying, giatrikhop numeric)  LANGUAGE 'plpgsql'  COST 100  VOLATILE PARALLEL UNSAFE  ROWS 1000  AS $BODY$  BEGIN  RETURN QUERY SELECT  ROW\_NUMBER() OVER (ORDER BY B.ma\_lenh\_dat DESC) ID,  B.ma\_co\_phieu,  cast(B.ma\_lenh\_dat as character varying),  B.SLKHOP,  B.SOLUONG,  B.thoi\_gian,  B.ma\_trang\_thai,  B.ma\_loai\_lenh,  B.gia,  B.GIAKHOP,  B.loai\_giao\_dich,  B.ma\_tai\_khoan,  trang\_thai\_lenh.ten\_trang\_thai,  B.SLKHOP \* B.GIAKHOP GIATRIKHOP  FROM  (SELECT D.ma\_co\_phieu, D.ma\_lenh\_dat, K.so\_luong SLKHOP, D.so\_luong + SUM(COALESCE(K.so\_luong, 0))  OVER (PARTITION BY D.ma\_lenh\_dat ORDER BY K.ma\_lenh\_khop DESC, K.ma\_lenh\_khop) SOLUONG,  K.thoi\_gian,  CASE  WHEN  (D.so\_luong = 0 AND  K.ma\_lenh\_khop = MAX(K.ma\_lenh\_khop) OVER (PARTITION BY D.ma\_lenh\_dat)) THEN D.ma\_trang\_thai  ELSE 'KP'  END  ma\_trang\_thai, D.ma\_loai\_lenh, D.gia, K.gia GIAKHOP, D.loai\_giao\_dich, D.ma\_tai\_khoan  FROM lenh\_dat D  JOIN lenh\_khop K  ON D.ma\_lenh\_dat = K.ma\_lenh\_dat  WHERE CAST(D.thoi\_gian AS DATE) = CAST(NOW() AS DATE)  UNION  SELECT lenh\_dat.ma\_co\_phieu, lenh\_dat.ma\_lenh\_dat, NULL SLKHOP,lenh\_dat.so\_luong SOLUONG,lenh\_dat.thoi\_gian, lenh\_dat.ma\_trang\_thai,lenh\_dat.ma\_loai\_lenh, lenh\_dat.gia, NULL GIAKHOP, lenh\_dat.loai\_giao\_dich, lenh\_dat.ma\_tai\_khoan  FROM lenh\_dat  WHERE so\_luong > 0  AND lenh\_dat.ma\_trang\_thai <> 'CKH'  AND CAST(lenh\_dat.thoi\_gian AS DATE) = CAST(NOW() AS DATE)) AS B  JOIN tai\_khoan\_ngan\_hang  ON B.ma\_tai\_khoan = tai\_khoan\_ngan\_hang.so\_tai\_khoan  JOIN trang\_thai\_lenh  ON B.ma\_trang\_thai = trang\_thai\_lenh.ma\_trang\_thai  WHERE tai\_khoan\_ngan\_hang.ma\_nha\_dau\_tu = ma\_ndt  ORDER BY ma\_lenh\_dat DESC, SOLUONG DESC;  END;  $BODY$; |

* Function **Sao kê lệnh đặt theo khoảng thời gian**

|  |
| --- |
| CREATE OR REPLACE FUNCTION public.udf\_lenh\_dat\_theo\_thoi\_gian(  ma\_ndt character varying,  thoi\_gian\_bat\_dau timestamp with time zone,  thoi\_gian\_ket\_thuc timestamp with time zone,  ma\_co\_phieu\_in character varying)  RETURNS TABLE(id bigint, ma\_co\_phieu character varying, ma\_lenh\_dat character varying, slkhop integer, soluong bigint, thoi\_gian timestamp with time zone, ma\_trang\_thai character varying,ma\_loai\_lenh character varying, gia numeric, giakhop numeric, loai\_giao\_dich boolean, ma\_tai\_khoan character varying, ten\_trang\_thai character varying, giatrikhop numeric)  LANGUAGE 'plpgsql'  COST 100  VOLATILE PARALLEL UNSAFE  ROWS 1000  AS $BODY$  BEGIN  RETURN QUERY SELECT  ROW\_NUMBER() OVER (ORDER BY B.ma\_lenh\_dat DESC) ID,  B.ma\_co\_phieu,  cast(B.ma\_lenh\_dat as character varying),  B.SLKHOP,  B.SOLUONG,  B.thoi\_gian,  B.ma\_trang\_thai,  B.ma\_loai\_lenh,  B.gia,  B.GIAKHOP,  B.loai\_giao\_dich,  B.ma\_tai\_khoan,  trang\_thai\_lenh.ten\_trang\_thai,  B.SLKHOP \* B.GIAKHOP GIATRIKHOP  FROM  (SELECT D.ma\_co\_phieu, D.ma\_lenh\_dat, K.so\_luong SLKHOP, D.so\_luong + SUM(COALESCE(K.so\_luong, 0))  OVER (PARTITION BY D.ma\_lenh\_dat ORDER BY K.ma\_lenh\_khop DESC, K.ma\_lenh\_khop) SOLUONG,  K.thoi\_gian,  CASE  WHEN  (D.so\_luong = 0 AND  K.ma\_lenh\_khop = MAX(K.ma\_lenh\_khop) OVER (PARTITION BY D.ma\_lenh\_dat)) THEN D.ma\_trang\_thai  ELSE 'KP'  END  ma\_trang\_thai, D.ma\_loai\_lenh, D.gia, K.gia GIAKHOP, D.loai\_giao\_dich, D.ma\_tai\_khoan  FROM lenh\_dat D  JOIN lenh\_khop K  ON D.ma\_lenh\_dat = K.ma\_lenh\_dat  WHERE (D.thoi\_gian > thoi\_gian\_bat\_dau or D.thoi\_gian = thoi\_gian\_bat\_dau) AND D.thoi\_gian< thoi\_gian\_ket\_thuc  UNION  SELECT lenh\_dat.ma\_co\_phieu, lenh\_dat.ma\_lenh\_dat, NULL SLKHOP,lenh\_dat.so\_luong SOLUONG,lenh\_dat.thoi\_gian, lenh\_dat.ma\_trang\_thai,lenh\_dat.ma\_loai\_lenh, lenh\_dat.gia, NULL GIAKHOP, lenh\_dat.loai\_giao\_dich, lenh\_dat.ma\_tai\_khoan  FROM lenh\_dat  WHERE so\_luong > 0 AND (lenh\_dat.thoi\_gian > thoi\_gian\_bat\_dau or lenh\_dat.thoi\_gian = thoi\_gian\_bat\_dau) AND lenh\_dat.thoi\_gian< thoi\_gian\_ket\_thuc  ) AS B  JOIN tai\_khoan\_ngan\_hang  ON B.ma\_tai\_khoan = tai\_khoan\_ngan\_hang.so\_tai\_khoan  JOIN trang\_thai\_lenh  ON B.ma\_trang\_thai = trang\_thai\_lenh.ma\_trang\_thai  WHERE tai\_khoan\_ngan\_hang.ma\_nha\_dau\_tu = ma\_ndt AND B.ma\_co\_phieu like CONCAT('%',ma\_co\_phieu\_in,'%')  ORDER BY ma\_lenh\_dat DESC, SOLUONG DESC;  END;  $BODY$; |

* Function **Sao kê lệnh khớp theo khoảng thời gian**

|  |
| --- |
| CREATE OR REPLACE FUNCTION public.udf\_lenh\_khop\_theo\_thoi\_gian(  ma\_ndt character varying,  thoi\_gian\_bat\_dau timestamp with time zone,  thoi\_gian\_ket\_thuc timestamp with time zone,  ma\_co\_phieu\_in character varying)  RETURNS TABLE(id bigint, ma\_co\_phieu character varying, ma\_lenh\_dat character varying, slkhop integer, soluong bigint, thoi\_gian timestamp with time zone, ma\_trang\_thai character varying, gia numeric, giakhop numeric, loai\_giao\_dich boolean, ma\_tai\_khoan character varying, ten\_trang\_thai character varying, giatrikhop numeric)  LANGUAGE 'plpgsql'  COST 100  VOLATILE PARALLEL UNSAFE  ROWS 1000  AS $BODY$  BEGIN  RETURN QUERY SELECT  ROW\_NUMBER() OVER (ORDER BY B.ma\_lenh\_dat DESC) ID,  B.ma\_co\_phieu,  cast(B.ma\_lenh\_dat as character varying),  B.SLKHOP,  B.SOLUONG,  B.thoi\_gian,  B.ma\_trang\_thai,  B.gia,  B.GIAKHOP,  B.loai\_giao\_dich,  B.ma\_tai\_khoan,  trang\_thai\_lenh.ten\_trang\_thai,  B.SLKHOP \* B.GIAKHOP GIATRIKHOP  FROM  (SELECT D.ma\_co\_phieu, D.ma\_lenh\_dat, K.so\_luong SLKHOP, D.so\_luong + SUM(COALESCE(K.so\_luong, 0))  OVER (PARTITION BY D.ma\_lenh\_dat ORDER BY K.ma\_lenh\_khop DESC, K.ma\_lenh\_khop) SOLUONG,  K.thoi\_gian,  CASE  WHEN  (D.so\_luong = 0 AND  K.ma\_lenh\_khop = MAX(K.ma\_lenh\_khop) OVER (PARTITION BY D.ma\_lenh\_dat)) THEN D.ma\_trang\_thai  ELSE 'KP'  END  ma\_trang\_thai, D.gia, K.gia GIAKHOP, D.loai\_giao\_dich, D.ma\_tai\_khoan  FROM lenh\_dat D  JOIN lenh\_khop K  ON D.ma\_lenh\_dat = K.ma\_lenh\_dat  WHERE (D.thoi\_gian > thoi\_gian\_bat\_dau or D.thoi\_gian = thoi\_gian\_bat\_dau) AND D.thoi\_gian< thoi\_gian\_ket\_thuc  UNION  SELECT lenh\_dat.ma\_co\_phieu, lenh\_dat.ma\_lenh\_dat, NULL SLKHOP,lenh\_dat.so\_luong SOLUONG,lenh\_dat.thoi\_gian, lenh\_dat.ma\_trang\_thai, lenh\_dat.gia, NULL GIAKHOP, lenh\_dat.loai\_giao\_dich, lenh\_dat.ma\_tai\_khoan  FROM lenh\_dat  WHERE so\_luong > 0 AND (lenh\_dat.thoi\_gian > thoi\_gian\_bat\_dau or lenh\_dat.thoi\_gian = thoi\_gian\_bat\_dau) AND lenh\_dat.thoi\_gian< thoi\_gian\_ket\_thuc  AND lenh\_dat.ma\_trang\_thai<>'DH' AND lenh\_dat.ma\_trang\_thai <> 'CK' ) AS B  JOIN tai\_khoan\_ngan\_hang  ON B.ma\_tai\_khoan = tai\_khoan\_ngan\_hang.so\_tai\_khoan  JOIN trang\_thai\_lenh  ON B.ma\_trang\_thai = trang\_thai\_lenh.ma\_trang\_thai  WHERE tai\_khoan\_ngan\_hang.ma\_nha\_dau\_tu = ma\_ndt AND B.ma\_co\_phieu like CONCAT('%',ma\_co\_phieu\_in,'%')  ORDER BY ma\_lenh\_dat DESC, SOLUONG DESC;  END;  $BODY$; |

# DANH MỤC TÀI LIỆU THAM KHẢO

1. Kiến thức cơ bản về PostgreSQL: <https://www.postgresqltutorial.com/>
2. Kiến thức về Java Springboot: <https://docs.spring.io/>
3. Kiến thức về ReactJS: <https://reactjs.org/>
4. Kiến thức cơ bản về thị trường chứng khoán:

<https://www.tcbs.com.vn/vi_VN/ho-tro/chi-tiet?chuyen-muc=5&url=cac-loai-lenh-giao-dich-chung-khoan>