

Trường Đại học Giao Thông Vận Tải

Khoa đào tạo quốc tế



BÁO CÁO BÀI TẬP LỚN

Môn học: Lập trình Java

Đề tài: App Quản lý sinh viên

Giảng viên hướng dẫn: Vũ Huấn

Sinh viên thực hiện: Lê Đình Tú

Lưu Huy Hiếu

Lớp: CNTT VA1

Khóa: K62

Hà Nội, Ngày 24 tháng 4 năm 2023



MỤC LỤC

Chương 1: Giới thiệu chung	4
1. Giới thiệu dự án	4
2. Mô tả dự án	4
Chương 2: Xây dựng chương trình.....	6
1. Giới thiệu và nêu chức năng của các package	6
2. Giới thiệu và các tương tác trong giao diện.	7
2.1. Kết nối database	7
2.2. Trang đăng nhập/đăng kí	7
2.3. Trang chính sau khi đăng nhập.	15
2.4. Trang Home	17
2.5. Trang Student Management.....	20
2.6. Trang Course Management.....	30
2.7. Trang Student Restoration	31
2.8. Trang Course Resstoration	34
2.9. Trang Setting.....	35
Chương 3: Kết luận và hướng phát triển.....	38
Chương 4. Tài liệu tham khảo.....	39

Lời mở đầu

Trong môi trường giáo dục hiện đại, việc quản lí học viên và khóa học là vô cùng quan trọng để đảm bảo sự thành công trong việc đào tạo và phát triển các học viên.

Ứng dụng Quản lí học viên được phát triển với mục đích giúp cho các trung tâm đào tạo có thể quản lí các thông tin sinh viên, chương trình đào tạo và khóa học,... một cách dễ dàng và tiện lợi.

Chúng tôi đã nghiên cứu và phát triển ứng dụng Quản lí học viên với mong muốn tạo ra một công cụ quản lí toàn diện, đơn giản và tiện lợi cho các trung tâm đào tạo. Báo cáo dưới đây sẽ trình bày chi tiết về các tính năng và lợi ích của ứng dụng này.

Chương 1: Giới thiệu chung

1. Giới thiệu dự án

Em xin được giới thiệu về dự án của nhóm em , một app quản lí học viên được phát triển bằng ngôn ngữ lập trình JAVA.

App của nhóm chúng em được thiết kế để quản lí học viên và các khóa học ở các trung tâm. Trong quá trình làm dự án, nhóm em đã tổ chức file theo mô hình MVC để dễ dàng quản lí và phát triển app.

Đối với việc quản lí khóa học và học viên của các trung tâm thì việc quản lí khá khó khăn nên đây là một app sẽ giúp việc quản lí dễ dàng hơn.

Ngoài ra, dự án cũng giúp cho chúng em có cơ hội học tập và rèn luyện kĩ năng lập trình JAVA.

Nhóm em rất mong muốn được chia sẻ với các bạn về dự án của nhóm em và hi vọng rằng game của nhóm em sẽ đem lại sự thích thú và giải trí cho các bạn

2. Mô tả dự án

- App sẽ có các trang chính như sau:

- + Home
- + Student Management
- +Course Management
- +Student Restoration

+Course Restoration

+Setting

- Một số chức năng chính của trang

+ Home: thấy được tổng số học viên đăng kí khóa học, số khóa học đang có ở trung tâm và 1 biểu đồ thấy được lượng học viên đăng kí theo ngày

+ Student Management: Có bảng thông tin học viên, có các phần tìm kiếm, xuất File, thêm/chỉnh sửa/xóa học viên.

+ Course Management: Có bảng thông tin khóa học, có các phần tìm kiếm, xuất File, thêm/chỉnh sửa/xóa khóa học.

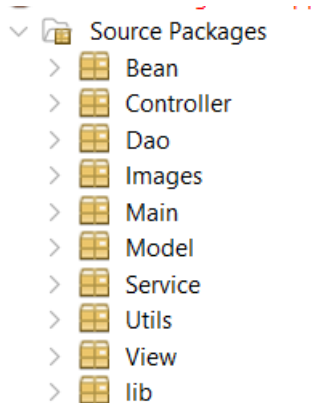
+ Student Restoration: Có bảng thông tin học viên đã bị xóa, có phần tìm kiếm, xuất File, khôi phục học viên, xóa cứng học viên

+ Course Restoration: Có bảng thông tin khóa học đã bị xóa, có phần tìm kiếm, xuất File, khôi phục khóa học, xóa cứng khóa học

+ Setting: có các chức năng đổi nền, đổi ngôn ngữ, trợ giúp và đăng xuất

Chương 2: Xây dựng chương trình

1. Giới thiệu và nêu chức năng của các package

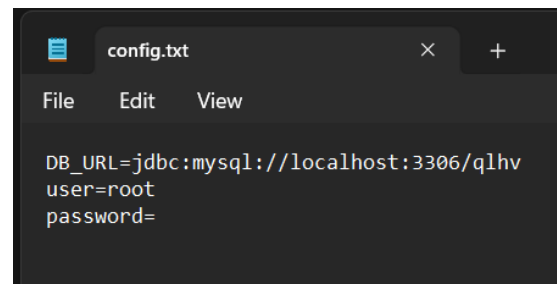
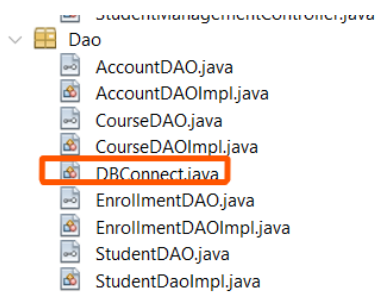


- **Bean:** chứa các đối tượng (object) mô tả dữ liệu trong ứng dụng. Thông thường, các đối tượng này được dùng để truyền dữ liệu giữa các thành phần của ứng dụng, ví dụ như truyền dữ liệu từ Controller tới View.
- **Controller:** chứa các lớp xử lý logic điều khiển (controller) trong ứng dụng. Các lớp này thường xử lý yêu cầu (request) từ phía người dùng và thực hiện các hành động tương ứng, ví dụ như truy xuất và cập nhật dữ liệu trong database.
- **Dao:** chứa các file hình ảnh và icon được sử dụng trong ứng dụng.
- **Main:** chứa file main của ứng dụng, thường là file khởi động và cấu hình.
- **Model:** chứa các lớp mô tả dữ liệu trong database, tương tự như package Bean.
- **Service:** chứa các lớp cung cấp dịch vụ (service) cho ứng dụng. Các dịch vụ này thường thực hiện các tác vụ chung, ví dụ như gửi email hoặc thực hiện các tính toán phức tạp.

- Utils: chứa các công cụ (class) có thể tái sử dụng nhiều lần hỗ trợ cho việc phát triển và thực thi ứng dụng.
- View: chứa các file giao diện (view) của ứng dụng.
- Lib: chứa các thư viện (library) sử dụng trong ứng dụng, ví dụ như các thư viện hỗ trợ việc kết nối database hoặc xử lý hình ảnh.

2. Giới thiệu và các tương tác trong giao diện.

2.1. Kết nối database



```

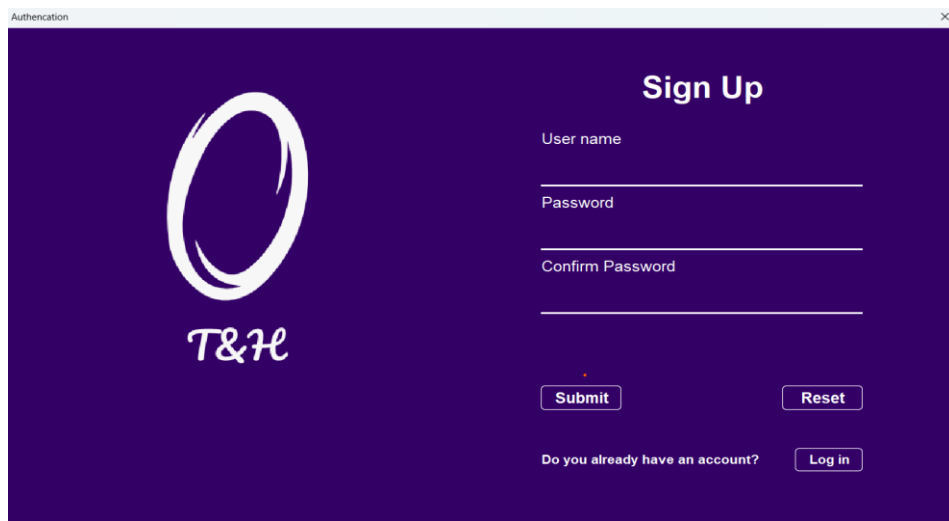
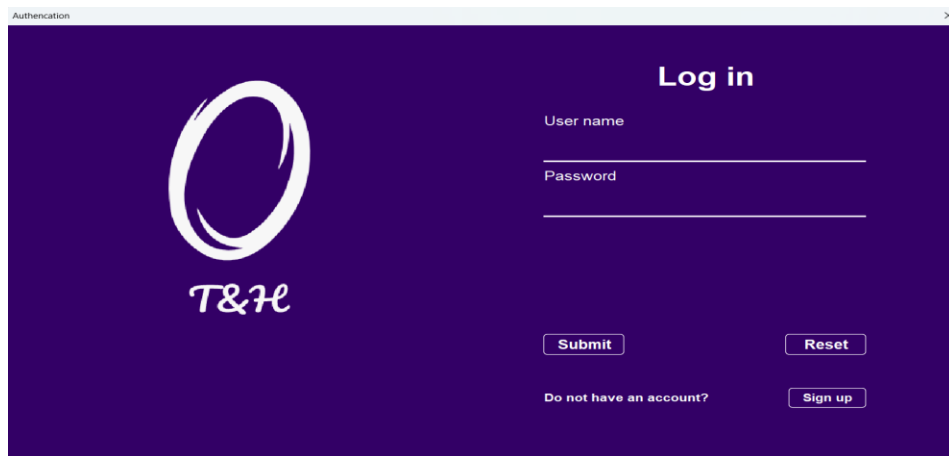
public static Connection getConnection() {
    try {
        Class.forName(className:"com.mysql.cj.jdbc.Driver");
        Connection conn = DriverManager.getConnection(url: getUrl(), user: getUser(), password: getPassword());
        return conn;
    } catch (ClassNotFoundException | SQLException ex) {
        return null;
    }
}

public static Map<String, String> readConfig() {
    Map<String, String> configMap = new HashMap<>();
    try (BufferedReader reader = new BufferedReader(new FileReader(fileName: CONFIG_FILE_PATH))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] keyValue = line.split(regex: "=");
            if (keyValue.length == 2) {
                configMap.put(keyValue[0], keyValue[1]);
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return configMap;
}

```

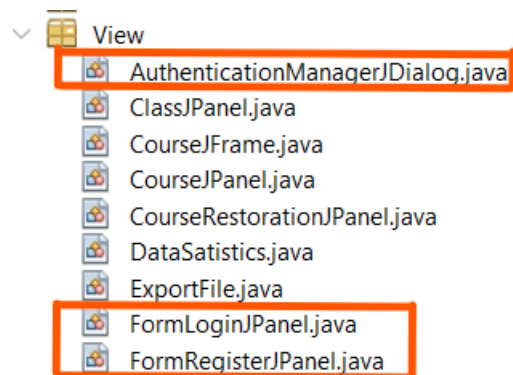
2.2. Trang đăng nhập/đăng kí

2.2.1. Giao diện:



2.2.2. Xây dựng giao diện và tương tác với giao diện

a) Xây dựng giao diện.

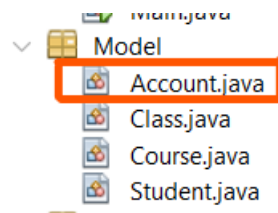


- File AuthenticationManagerJDialog.java: thiết kế ra 1Jframe trong đó có 2 Jpanel : 1Jpanel để hiện Logo và 1 Jpanel để diện FormLoginJPanel và FormRegisterJPanel.

- File FormLoginJPanel.java và FormRegisterJPanel.java: thiết kế ra 2 cái Jpanel đăng nhập và đăng kí.

b) Xây dựng tương tác

b.1. File Account



```
package model;
public class Account {
    private int accountCode;
    private String userName;
    private String password;
    private boolean status;

    public Account() {}

    public int getAccountCode() {
        return accountCode;
    }

    public void setAccountCode(int accountCode) {
        this.accountCode = accountCode;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
    }

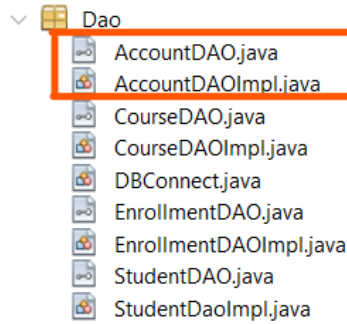
    public void setPassword(String password) {
        this.password = password;
    }

    public boolean isStatus() {
        return status;
    }

    public void setStatus(boolean status) {
        this.status = status;
    }
}
```

- Xây dựng các thuộc tính ,phương thức quản lý thông tin tài khoản

b.2. File AccountDao/ AccountDaoImpl



○ File AccountDao:

```
package Dao;

import model.Account;

public interface AccountDAO {
    public Account login(String userName, String passWord);
    public boolean register(Account account);
    public boolean isUsernameExist(String username);
}
```

○ File AccountDaoImpl:

```
@Override
public Account login(String userName, String passWord) {
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    Account account = null;

    try {
        conn = (Connection) DBConnect.getConnection();
        String sql = "SELECT * FROM account WHERE user_name LIKE ? AND password LIKE ?";
        ps = conn.prepareStatement(sql);
        ps.setString(1, userName);
        ps.setString(2, passWord);
        rs = ps.executeQuery();
        if (rs.next()) {
            account = new Account();
            account.setAccountCode(rs.getInt("account_code"));
            account.setUserName(userName);
            account.setPassword(rs.getString("password"));
            account.setStatus(rs.getBoolean("status"));
        }
    } catch (SQLException e) {
        if (conn != null) {
            try {
                conn.rollback(); // Hủy bỏ transaction nếu có lỗi xảy ra
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (ps != null) {
                ps.close();
            }
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

```
@Override
public boolean isUsernameExist(String username) {
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    try {
        conn = (Connection) DBConnect.getConnection();

        String sql = "SELECT COUNT(*) AS count FROM account WHERE user_name = ?";
        ps = conn.prepareStatement(sql);
        ps.setString(1, username);
        rs = ps.executeQuery();
        if (rs.next()) {
            int count = rs.getInt("count");
            if (count > 0) {
                return true;
            }
        }
    } catch (SQLException e) {
        System.out.println(e);
        if (conn != null) {
            e.printStackTrace();
        }
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (ps != null) {
                ps.close();
            }
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException ex) {
            System.out.println("Lỗi đóng kết nối");
        }
    }
    return false;
}
```

```

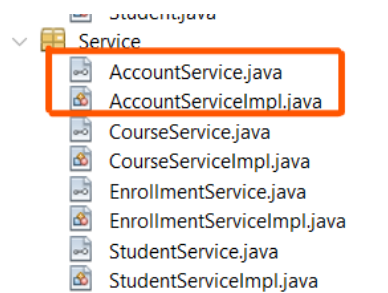
@Override
public boolean register(Account account) {
    Connection conn = null;
    PreparedStatement ps = null;
    boolean result = false;

    try {
        conn = (Connection) DBConnect.getConnection();
        conn.setAutoCommit(false); // Bắt đầu transaction
        String sql = "INSERT INTO account (user_name, password, status) VALUES (?, ?, ?)";
        ps = conn.prepareStatement(sql);
        ps.setString(1, account.getUserName());
        ps.setString(2, account.getPassword());
        ps.setBoolean(3, account.isStatus());
        int rows = ps.executeUpdate();
        conn.commit(); // Kết thúc transaction và lưu các thao tác dữ liệu vào cơ sở dữ liệu
        if (rows > 0) {
            result = true;
        }
    } catch (SQLException e) {
        System.out.println("lỗi tạo tài khoản");
        if (conn != null) {
            try {
                conn.rollback(); // Hủy bỏ transaction nếu có lỗi xảy ra
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    } finally {
        try {
            if (ps != null) {
                ps.close();
            }
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException ex) {
            System.out.println("lỗi đóng kết nối");
        }
    }
    return result;
}

```

- Giải thích: Định nghĩa và xây dựng các phương thức tương tác với bảng account trên DB, bao gồm phương thức login để kiểm tra tên đăng nhập và mật khẩu của tài khoản, phương thức register để đăng ký tài khoản và phương thức isUsernameExist để kiểm tra xem tên đăng nhập đã tồn tại trong cơ sở dữ liệu hay chưa.

b.3. File AccountService/AccountServiceImpl



- File AccountService:

```

package Service;

import model.Account;

/**
 *
 * @author MUCKHOTAU
 */
public interface AccountService {
    public Account login(String userName, String password);
    public boolean register(Account account);
    public boolean isUsernameExist(String username);
}

```

○ File AccountServiceImpl:

```

package Service;

import Dao.AccountDAO;
import Dao.AccountDAOImpl;
import model.Account;

/**
 *
 * @author MUCKHOTAU
 */
public class AccountServiceImpl implements AccountService {
    private AccountDAO accountDAO = null;
    public AccountServiceImpl() {
        accountDAO = new AccountDAOImpl();
    }

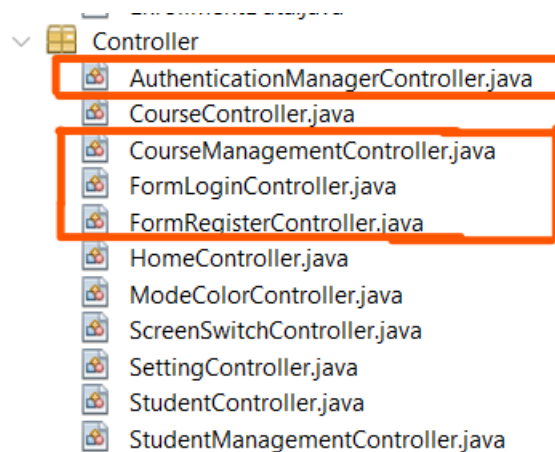
    @Override
    public Account login(String userName, String password) {
        return accountDAO.login(userName, password);
    }

    @Override
    public boolean register(Account account) {
        return accountDAO.register(account);
    }

    @Override
    public boolean isUsernameExist(String username) {
        return accountDAO.isUsernameExist(username);
    }
}

```

b.4. File FormRegisterController / ForLoginController / AuthenticationManagerController



- File FormRegisterController / FormLoginController:

- Xét sự kiện click vào nút Submit. Khi người dùng nhập không đúng thì sẽ xuất hiện dòng báo lỗi và khi các phần nhập đều đúng thì người dùng sẽ tạo tài khoản thành công và tài khoản sẽ được lưu trên database để lần sau người dùng có thể đăng nhập bằng tài khoản vừa tạo.

```
btnSubmit.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        String password = new String(value: jpfPassword.getPassword());
        String confirmPassword = new String(value: jpfConfirmPassword.getPassword());
        try {
            if (jtfUserName.getText().length() == 0
                || password.length() == 0
                || confirmPassword.length() == 0) {
                jlbMsg.setText(text: "Please enter required data");
            } else {
                if (!jtfUserName.getText().matches(regex: "^[A-Za-z0-9-._%+]{0}[A-Za-z0-9-._+\\.[A-Za-z]{2,})$")) {
                    jlbMsg.setText(text: "Username must be email.");
                } else if (!jtfUserName.getText().matches(regex: "^[\\S]+$")) {
                    jlbMsg.setText(text: "Username must not contain any whitespace.");
                } else if (accountService.isUsernameExist(username: jtfUserName.getText())) {
                    jlbMsg.setText(text: "The account name already exists on the system");
                } else if (!password.matches(regex: "^[\\S]{6,}$")) {
                    jlbMsg.setText(text: "Password must be at least 6 characters long ");
                } else if (!password.equals(anoject: confirmPassword)) {
                    jlbMsg.setText(text: "Password re-enter mismatch");
                } else {
                    Account account = new Account();
                    account.setUserName(username: jtfUserName.getText());
                    account.setPassword(password);
                    account.setStatus(status: true);
                    accountService.register(account);
                    jlbMsg.setText(text: "Successful registration");
                }
            }
        } catch (Exception ex) {
            System.out.println(ex.toString());
            jlbMsg.setText(text: "Connection error");
        }
    }
});
```

- Xét sự kiện click vào nút Reset: Thì toàn bộ text người dùng đã nhập sẽ được xóa đi để người dùng nhập lại toàn bộ.

```
btnReset.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        jtfUserName.setText(t: "");
        jpfPassword.setText(t: "");
        jpfConfirmPassword.setText(t: "");
    }
});
```

- Xét sự kiện click vào nút Login: Thì giao diện Đăng nhập sẽ xuất hiện để người dùng Đăng nhập tài khoản đã tạo thay vì Đăng kí tài khoản mới.

```

        btnLogin.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                JPanel node = new FormLoginJPanel();
                jpnUserAuth.removeAll();
                jpnUserAuth.setLayout(new BorderLayout());
                jpnUserAuth.add(comp: node);
                jpnUserAuth.validate();
                jpnUserAuth.repaint();
            }
        });

```

- Xét sự kiện click vào SignUp: Thì giao diện Đăng kí sẽ xuất hiện để người tạo tài khoản mới.

```

        btnSignUp.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                JPanel node = new FormRegisterJPanel();
                jpnUserAuth.removeAll();
                jpnUserAuth.setLayout(new BorderLayout());
                jpnUserAuth.add(comp: node);
                jpnUserAuth.validate();
                jpnUserAuth.repaint();
            }
        });

```

- AuthenticationManagerController: Đoạn code để khi chạy chương trình luôn hiện giao diện phần đăng nhập trước.

```

package Controller;

import View.FormLoginJPanel;
import java.awt.BorderLayout;
import java.awt.Dialog;
import javax.swing.JPanel;

public class AuthenticationManagerController {

    public static Dialog authenticationManagerDialog;
    public static JPanel jpnUserAuth;

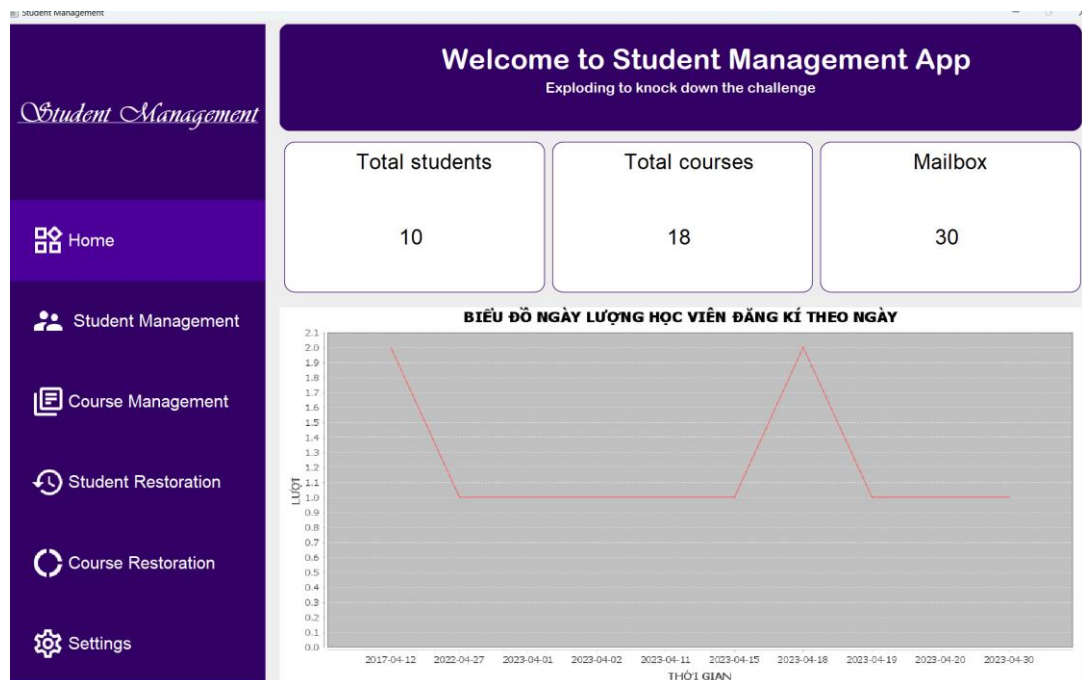
    public AuthenticationManagerController(Dialog authenticationManagerDialog, JPanel jpnUserAuth) {
        AuthenticationManagerController.authenticationManagerDialog = authenticationManagerDialog;
        AuthenticationManagerController.jpnUserAuth = jpnUserAuth;
    }

    public void createUI() {
        if (jpnUserAuth != null) {
            FormLoginJPanel loginPanel = new FormLoginJPanel();
            jpnUserAuth.removeAll();
            jpnUserAuth.setLayout(new BorderLayout());
            jpnUserAuth.add(comp: loginPanel, constraints: BorderLayout.CENTER);
            jpnUserAuth.validate();
            jpnUserAuth.repaint();
        }
    }
}

```

2.3. Trang chính sau khi đăng nhập.

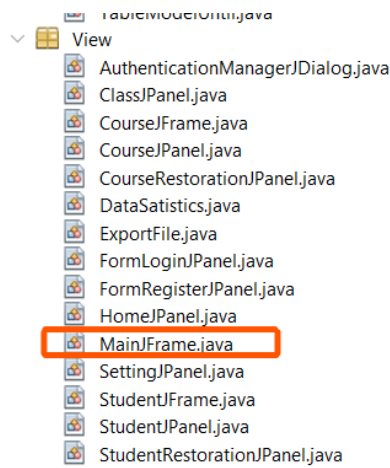
2.3.1. Giao diện.



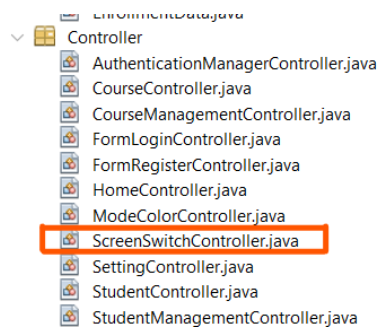
2.3.2. Xây dựng giao diện và các tương tác với giao diện.

a) Xây dựng giao diện

- Đây là file trong đó là các đoạn mã code để thiết kế ra 1 JFrame chính trong đó được chia là 2Jpanel : 1 Jpanel để thiết kế ra cột Menu và 1 Jpanel để chứa ra các giao diện của các trang.



b) Xây dựng tương tác



- Đây là File chứa các đoạn mã code để viết các tương tác của người dùng tương tác với thanh Menu.
- `setView` là đoạn mã code để set khi người dùng đăng nhập thành công vào app thì giao diện sẽ hiện mặc định là trang Home.

```
public void setView(JPanel jPanelItem, JLabel jLabelItem) {
    currentPage = "Home";
    jPanelItem.setBackground(bg: ColorBgMenuController);
    jLabelItem.setBackground(bg: ColorBgMenuController);

    jPanelRoot.removeAll();
    jPanelRoot.setLayout(new BorderLayout());
    jPanelRoot.add(new HomeJPanel());
    jPanelRoot.validate();
    jPanelRoot.repaint();
}
```

- Đoạn mã code bên dưới là set các sự kiện khi người dùng click vào Menu vào ô lựa chọn thì giao diện trang tương ứng với kind đó sẽ xuất hiện đồng thời các ô lựa chọn cũng sẽ đổi màu.


```

@Override
public void mouseClicked(MouseEvent e) {
    switch (kind) {
        case "Home" -> {
            node = new HomeJPanel();
        }
        case "Student" -> {
            node = new StudentJPanel();
        }
        case "Course" -> {
            node = new CourseJPanel();
        }
        case "Class" -> {
            node = new ClassJPanel();
        }
        case "StudentRestoration" -> {
            node = new StudentRestorationJPanel();
        }
        case "CourseRestoration" -> {
            node = new CourseRestorationJPanel();
        }
        case "Setting" -> {
            node = new SettingJPanel();
        }
        default -> {
        }
    }
    jPanelRoot.removeAll();
    jPanelRoot.setLayout(new BorderLayout());
    jPanelRoot.add(comp: node);
    jPanelRoot.validate();
    jPanelRoot.repaint();
    setChangeBackground(kind);
}

@Override
public void mouseReleased(MouseEvent e) {
    // Nếu currentPage trùng với kind hiển thị tại thì không cần thay đổi màu nút
    if (currentPage.equalsIgnoreCase(anotherString: kind)) {
        return;
    }
    jPanelItem.setBackground(bg: ColorBgMenu);
    jLabelItem.setBackground(bg: ColorBgMenu);
}

@Override
public void mouseEntered(MouseEvent e) {
    if (!currentPage.equalsIgnoreCase(anotherString: kind)) {
        jPanelItem.setBackground(bg: ColorBgMenuController);
        jLabelItem.setBackground(bg: ColorBgMenuController);
    }
}

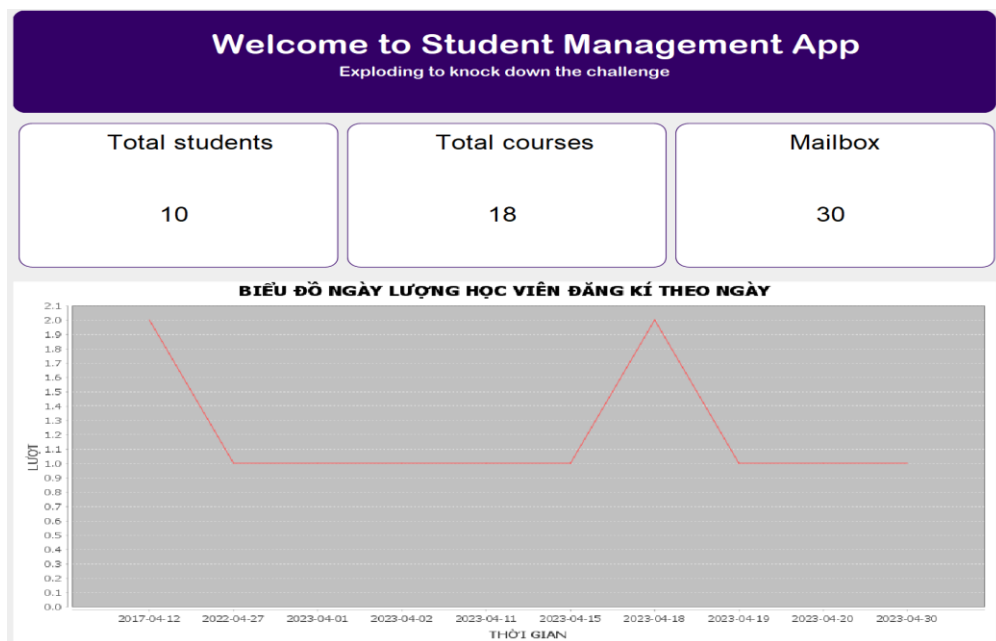
@Override
public void mouseExited(MouseEvent e) {
    if (!currentPage.equalsIgnoreCase(anotherString: kind)) {
        jPanelItem.setBackground(bg: ColorBgMenu);
        jLabelItem.setBackground(bg: ColorBgMenu);
    }
}

private void setChangeBackground(String kind) {
    for (CategoryBean item : listItem) {
        if (item.getKind().equalsIgnoreCase(anotherString: kind)) {
            item.getJPanel().setBackground(bg: ColorBgMenuController);
            item.getLabel().setBackground(bg: ColorBgMenuController);
        } else {
            item.getJPanel().setBackground(bg: ColorBgMenu);
            item.getLabel().setBackground(bg: ColorBgMenu);
        }
    }
}

```

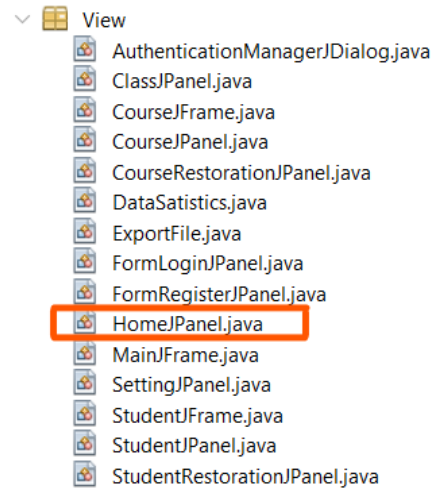
2.4. Trang Home

2.4.1. Giao diện



2.4.2. Xây dựng giao diện và tương tác

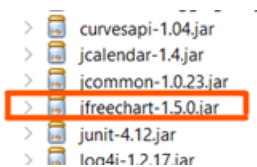
a) Xây dựng giao diện



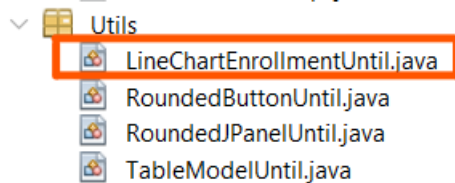
- Đây là File chứa các đoạn mã code để thiết kế lên giao diện của trang Home.
- Giao diện trang Home được chứa trong 1Jpanel với LayOut là BorderLayout. Và trong Jpanel lại được chia thành 3Jpanel nhỏ: 1Jpanel chứa text nằm ở phía North, 1Jpanel chứa tổng số học viên, khóa học, email nằm ở Center và 1Jpanel chứa biểu đồ lượng học viên đăng kí khóa học.

b) Xây dựng tương tác

- Để có thể vẽ được biểu đồ thì đầu tiên ta cần tải thư viện:



○ File LineChartEnrollmentUntil:



- Đoạn mã code bên dưới là để có thể lấy ra được dữ liệu học viên đăng kí khóa học và trả về dữ liệu các học viên đã đăng kí.

```

private DefaultCategoryDataset createDataset() {
    List<EnrollmentData> listItem = enrollmentService.getEnrollmentList();
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();

    if (listItem != null) {
        listItem.forEach((var item) -> {
            dataset.addValue(value: item.getEnrollmentCount(), rowKey: "Học viên", columnKey: item.getEnrollmentDate());
        });
    }
    return dataset;
}

```

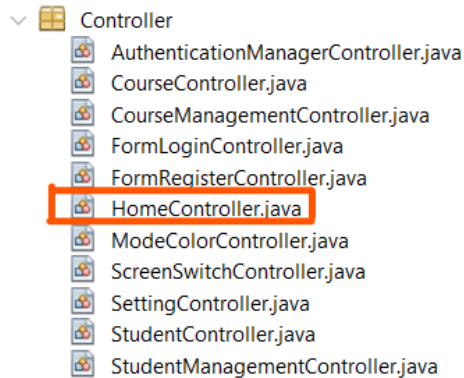
➤ Đoạn mã code bên dưới để tạo ra biểu đồ và hiện biểu đồ lên giao diện.

```

public void addLineChart(JPanel jpnItem) {
    JFreeChart lineChart = ChartFactory.createLineChart(
        title: "Biểu đồ ngày lượng học viên đăng kí theo ngày".toUpperCase(),
        categoryAxisLabel: "THỜI GIAN", valueAxisLabel: "LƯỢNG", dataset: createDataset(),
        orientation: PlotOrientation.VERTICAL, legend: false, tooltips: false, urls: false);
    ChartPanel chartPanel = new ChartPanel(chart: lineChart);
    chartPanel.setPreferredSize(new Dimension(width: jpnItem.getWidth(), height: jpnItem.getHeight()));
    jpnItem.removeAll();
    jpnItem.setLayout(new CardLayout());
    jpnItem.add(comp: chartPanel);
    jpnItem.validate();
    jpnItem.repaint();
    List<EnrollmentData> listData = enrollmentService.getEnrollmentList();
}

```

○ File HomeController



➤ Đoạn mã code dưới là để có thể lấy số học viên đã đăng kí khóa học và số khóa học ở trên database để có thể hiện ra số học viên và số khóa học ra màn hình.

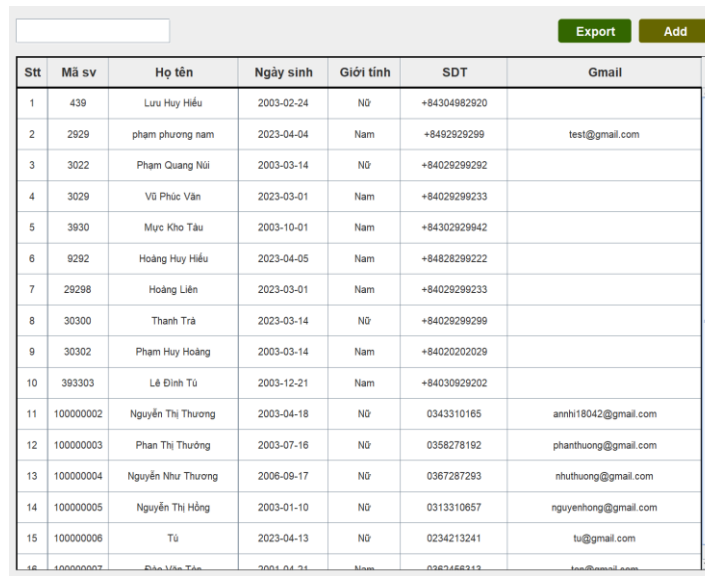
```

public void renderData() {
    int totalCourse = courseService.getTotalCourse();
    String totalCourseString = Integer.toString(i: totalCourse);
    jlbTotalCourse.setText(text: totalCourseString);
    int totalStudent = enrollmentService.getTotalStudent();
    String totalStudentString = Integer.toString(i: totalStudent);
    jlbTotalStudent.setText(text: totalStudentString);
    LineChartEnrollmentUntil lineChart = new LineChartEnrollmentUntil();
    lineChart.addLineChart(jpnItem: jpnLineChart);
}

```

2.5. Trang Student Management

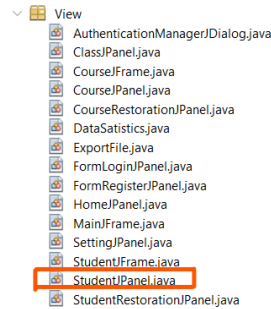
2.5.1. Giao diện



Stt	Mã sv	Họ tên	Ngày sinh	Giới tính	SĐT	Gmail
1	439	Lưu Huy Hiếu	2003-02-24	Nữ	+84304982920	
2	2929	phạm phương nam	2023-04-04	Nam	+8492929299	test@gmail.com
3	3022	Phạm Quang Núi	2003-03-14	Nữ	+84029299292	
4	3029	Vũ Phúc Văn	2023-03-01	Nam	+84029299233	
5	3930	Mục Kho Tàu	2003-10-01	Nam	+84302929942	
6	9292	Hoàng Huy Hiếu	2023-04-05	Nam	+84828299222	
7	29298	Hoàng Liên	2023-03-01	Nam	+84029299233	
8	30300	Thanh Trà	2023-03-14	Nữ	+84029299299	
9	30302	Phạm Huy Hoàng	2003-03-14	Nam	+84020202029	
10	393303	Lê Đình Tú	2003-12-21	Nam	+84030929202	
11	100000002	Nguyễn Thị Thương	2003-04-18	Nữ	0343310165	amthi18042@gmail.com
12	100000003	Phan Thị Thuởng	2003-07-16	Nữ	0358278192	phanthuong@gmail.com
13	100000004	Nguyễn Như Thương	2006-09-17	Nữ	0367287293	nhuthuong@gmail.com
14	100000005	Nguyễn Thị Hồng	2003-01-10	Nữ	0313310657	nguyenhong@gmail.com
15	100000006	Tú	2023-04-13	Nữ	0234213241	tu@gmail.com
16	100000007	Đào Văn Tân	2004-04-31	Nam	0363468343	tan@gmail.com

2.5.2. Xây dựng giao diện và tương tác.

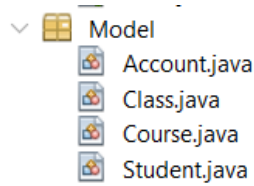
a) Xây dựng giao diện



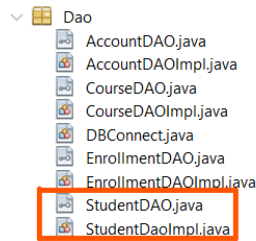
- Đây là File để thiết kế lên giao diện trang StudentManagement. Giao diện được chứa trong 1Jpanel và nó được setLayout là BorderLayout. Bên trong nó lại được chia làm 2Jpanel: 1 Jpanel nằm ở phí North chứa một ô tìm kiếm, 1 nút Export và 1 nút Add. 1 Jpanel nằm ở Center chứa bảng thông tin các học viên.

b) Xây dựng tương tác

- File Student: Nơi chứa các thuộc tính của học viên



○ File StudentDao / StudentDaoImpl



○ File Student Dao:

```
package Dao;

import java.util.List;

import Model.Student;

public interface StudentDAO {

    public List<Student> getList(boolean isDeleted);

    public int createOrUpdate(Student student);

    public int softDelete(Student student);

    public int hardDelete(Student student);

    public int recoverSoftDeletedStudent(Student student);

}
```

○ File StudentDaoImpl:

- Đoạn mã code dưới là để có thể lấy được danh sách học viên từ trên database xuống với biến boolean isDeleted ,nếu isDeleted = true thì getList sẽ trả về những sinh viên đã bị xóa mềm, và ngược lại

```
@Override
public List<Student> getList(boolean isDeleted) {
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    List<Student> list = new ArrayList<>();
    try {
        conn = DBConnect.getConnection();
        String sql = isDeleted ? "SELECT * FROM student WHERE status = false" : "SELECT * FROM student WHERE status = true";
        ps = conn.prepareStatement(string:sql);
        rs = ps.executeQuery();

        while (rs.next()) {
            Student student = new Student();
            student.setStudentCode(studentCode: rs.getInt(string:"student_code"));
            student.setFullName(fullName: rs.getString(string:"full_name"));
            student.setDateOfBirth(dateOfBirth: rs.getDate(string:"date_of_birth"));
            student.setStatus(status: rs.getBoolean(string:"status"));
            student.setSex(sex: rs.getBoolean(string:"sex"));
            student.setGmail(address: rs.getString(string:"gmail"));
            student.setPhoneNumber(phoneNumber: rs.getString(string:"phone_number"));
            list.add(e: student);
        }
        return list;
    } catch (SQLException ex) {
        System.out.println("Error occurred: " + ex.getMessage());
    } finally {
        try {
            if (ps != null) {
                ps.close();
            }
        } catch (SQLException ex) {
            System.out.println(ex);
        }
    }
    try {
        if (conn != null) {

```

- Đoạn code bên dưới để có thể thêm hoặc chỉnh sửa thông tin các học viên.

```
@Override
public int createOrUpdate(Student student) {
    String sql = "INSERT INTO student(student_code, full_name, date_of_birth, sex, phone_number, gmail, status) "
        + "VALUES(?, ?, ?, ?, ?, ?, ?) ON DUPLICATE KEY UPDATE"
        + " full_name = VALUES(full_name), date_of_birth = VALUES(date_of_birth), sex = VALUES(sex),"
        + " phone_number = VALUES(phone_number), gmail = VALUES(gmail), status = VALUES(status);";
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    int generatedKey = 0; // Khởi tạo giá trị mặc định cho generatedKey
    try {
        conn = DBConnect.getConnection();
        ps = conn.prepareStatement(string:sql, rs: PreparedStatement.RETURN_GENERATED_KEYS);
        ps.setInt(rs: 1, rs: student.getStudentCode());
        ps.setString(rs: 2, string:student.getFullName());
        ps.setDate(rs: 3, new Date(date: student.getDateOfBirth().getTime()));
        ps.setBoolean(rs: 4, bln: student.isSex());
        ps.setString(rs: 5, string:student.getPhoneNumber());
        ps.setString(rs: 6, string:student.getEmail());
        ps.setBoolean(rs: 7, bln: student.isStatus());
        int affectedRows = ps.executeUpdate();
        if (affectedRows > 0) {
            rs = ps.getGeneratedKeys();
            if (rs.next()) {
                generatedKey = rs.getInt(rs: 1);
                return generatedKey;
            }
        } else {
            System.err.println(x: "Không có dòng nào được thêm hoặc cập nhật trong cơ sở dữ liệu!");
        }
    } catch (SQLException ex) {
        System.err.println("Error occurred: " + ex.getMessage());
    } finally {
        try {
            if (ps != null) {
                ps.close();
            }
        }
    }
}
```

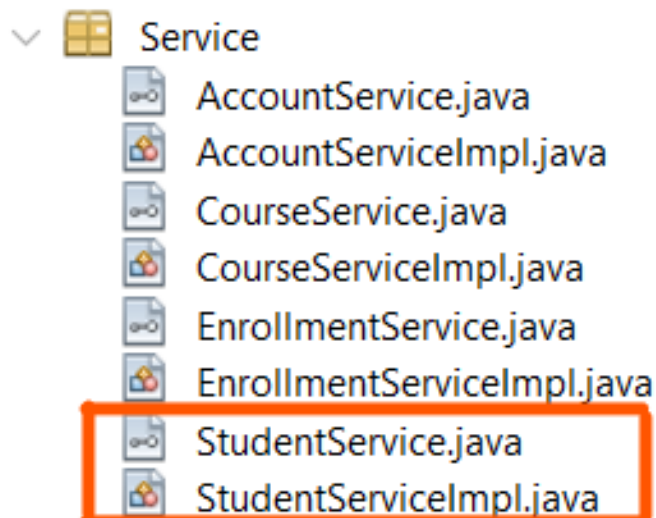
- Đoạn code bên dưới để xóa mềm (ẩn) các học viên và trên database vẫn còn và người dùng vẫn có thể khôi phục được.

```
@Override
public int softDelete(Student student) {
    String sql = "UPDATE student SET status = false WHERE student_code = ?";
    Connection conn = null;
    PreparedStatement ps = null;
    try {
        conn = DBConnect.getConnection();
        conn.setAutoCommit(bln:false); // Bắt đầu transaction
        ps = conn.prepareStatement(string:sql);
        ps.setInt(rs: 1, rs: student.getStudentCode());
        int rs = ps.executeUpdate();
        conn.commit(); // Kết thúc transaction và lưu các thao tác dữ liệu vào cơ sở dữ liệu
        return rs;
    } catch (SQLException ex) {
        System.err.println(x: ex);
        if (conn != null) {
            try {
                conn.rollback(); // Hủy bỏ transaction nếu có lỗi xảy ra
            } catch (SQLException e) {
                System.err.println(x: e);
            }
        }
    } finally {
        try {
            if (ps != null) {
                ps.close();
            }
        } catch (SQLException ex) {
            System.err.println(x: ex);
        }
        try {
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException ex) {
            System.err.println(x: ex);
        }
    }
}
```

- Đoạn code bên dưới là để người dùng có thể xóa cứng đi học viên và không thể khôi phục được học viên đó.

```
@Override
public int hardDelete(Student student) {
    String sql = "DELETE FROM student WHERE student_code = ?";
    Connection conn = null;
    PreparedStatement ps = null;
    try {
        conn = DBConnect.getConnection();
        conn.setAutoCommit(false); // Bắt đầu transaction
        ps = conn.prepareStatement(string:sql);
        ps.setInt(1, 1, 1: student.getStudentCode());
        int rs = ps.executeUpdate();
        conn.commit(); // Kết thúc transaction và lưu các thao tác dữ liệu vào cơ sở dữ liệu
        return rs;
    } catch (SQLException ex) {
        System.err.println(x: ex);
        if (conn != null) {
            try {
                conn.rollback(); // Hủy bỏ transaction nếu có lỗi xảy ra
            } catch (SQLException e) {
                System.err.println(x: e);
            }
        }
    } finally {
        try {
            if (ps != null) {
                ps.close();
            }
        } catch (SQLException ex) {
            System.err.println(x: ex);
        }
        try {
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException ex) {
            System.err.println(x: ex);
        }
    }
}
```

- File StudentService / StudentServiceImpl



- File StudentService:

```

package Service;

import java.util.List;

import Model.Student;

public interface StudentService {

    public List<Student> getList(boolean isDeleted);

    public int createOrUpdate(Student student);

    public int softDelete(Student student);

    public int hardDelete(Student student);

    public int recoverSoftDeletedStudent(Student student);

}

```

○ File StudentServiceImpl:

```

public class StudentServiceImpl implements StudentService {

    private StudentDAO studentDAO = null;

    public StudentServiceImpl() {
        studentDAO = new StudentDaoImpl();
    }

    @Override
    public List<Student> getList(boolean isDeleted) {
        return studentDAO.getList(isDeleted);
    }

    @Override
    public int createOrUpdate(Student student) {
        return studentDAO.createOrUpdate(student);
    }

    @Override
    public int softDelete(Student student) {
        return studentDAO.softDelete(student);
    }

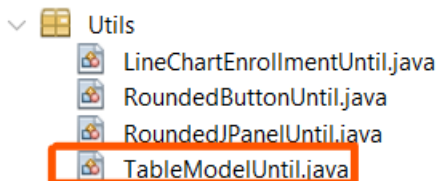
    @Override
    public int hardDelete(Student student) {
        return studentDAO.hardDelete(student);
    }

    @Override
    public int recoverSoftDeletedStudent(Student student) {
        return studentDAO.recoverSoftDeletedStudent(student);
    }

}

```

○ File TableModelUtil



- Đoạn mã code bên dưới là để lấy dữ liệu học viên và tạo ra model học viên.


```

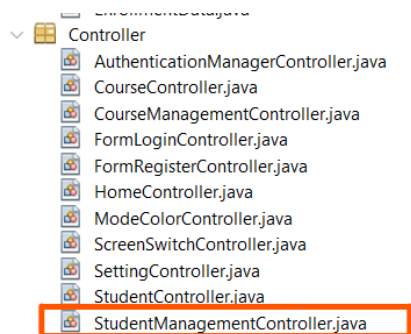
public DefaultTableModel setTableStudent(List<Student> listItem, String[] listColumn) {
    DefaultTableModel dtm = new DefaultTableModel() {
        @Override
        public boolean isCellEditable(int row, int column) {
            return false;
        }

        @Override
        public Class<?> getColumnClass(int columnIndex) {
            return switch (columnIndex) {
                case 0 -> Integer.class;
                case 3 -> Date.class;
                case 4 -> String.class;
                default -> super.getColumnClass(columnIndex);
            };
        }
    };

    dtm.setColumnIdentifiers(newIdentifiers: listColumn);
    int columns = listColumn.length;
    Object[] obj = null;
    int rows = listItem.size();
    if (rows > 0) {
        for (int i = 0; i < rows; i++) {
            Student student = listItem.get(index: i);
            obj = new Object[columns];
            obj[0] = (i + 1);
            obj[1] = student.getStudentCode();
            obj[2] = student.getFullName();
            obj[3] = student.getDateOfBirth();
            obj[4] = student.isSex() == true ? "Nam" : "Nữ";
            obj[5] = student.getPhoneNumber();
            obj[6] = student.getEmail();
            dtm.addRow(rowData: obj);
        }
    }
    return dtm;
}

```

- File StudentManagementController (file này quản lý cả trang Student Restoration)



- Đoạn mã code bên dưới là chỉnh sửa lại dữ liệu trong cột và add bảng vào giao diện với tham số boolean isDeleted (để phân biệt hàm sẽ quản lý trang Student hay Student Restoration)

```

public void renderData(boolean isDeleted) {
    this.isDeleted = isDeleted;
    List<Student> listItem = studentService.getList(isDeleted);
    if (listItem == null) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Không có dữ liệu để hiển thị.", title: "Thông báo", messageType: JOptionPane.INFORMATION_MESSAGE);
    } else {
        model = new TableModelUntil().setTableStudent(listItem, listColumn);
        tableStudent = new JTable(dm: model);

        // căn giữa text
        DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment(JLabel.CENTER);
        tableStudent.setDefaultRenderer(columnClass: Object.class, renderer: centerRenderer);
        //căn giữa cho cột 0 và 3
        tableStudent.getColumnModel().getColumn(columnIndex: 0).setCellRenderer(cellRenderer: centerRenderer);
        tableStudent.getColumnModel().getColumn(columnIndex: 3).setCellRenderer(cellRenderer: centerRenderer);
        // Set border color
        tableStudent.setBackground(bg: Color.WHITE);
        tableStudent.setBorder(border: BorderFactory.createLineBorder(color: Color.BLACK, thickness:2));
        tableStudent.getTableHeader().setBorder(border: BorderFactory.createLineBorder(color: Color.BLACK, thickness:2));
        // Set font
        tableStudent.setFont(
            new Font(name: "Times New Roman", style: Font.PLAIN, size: 16));
        tableStudent.getTableHeader()
            .setFont(new Font(name: "Times New Roman", style: Font.BOLD, size: 20));

        // Set font
        tableStudent.setFont(new Font(name: "Arial", style: Font.PLAIN, size: 16));
        tableStudent.getTableHeader().setFont(new Font(name: "Arial", style: Font.BOLD, size: 20));

        // Set chiều rộng và dài
        tableStudent.getTableHeader().setPreferredSize(new Dimension(width: 100, height:50));
        tableStudent.setRowHeight(rowHeight:50);
    }
}

```

```

public StudentJPanel() {
    this.init();
    pageStudentController = new StudentManagementController(jpnInfor, btnAdd, jTextFieldSearch:jtfSearch, btnExport);
    pageStudentController.renderData(isDeleted: false);
    pageStudentController.setEven();
}

```

➤ Ở đây chúng ta đang quản lý trang student nên sẽ chuyển vào tham số false.

- Đoạn mã bên dưới là sự kiện người dùng click vào nút Add thì sẽ hiện ra một JFrame để người dùng thêm học viên.

```

if (isDeleted == false) {
    btnAdd.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            StudentJFrame frame = new StudentJFrame(new Student());
            frame.setTitle(title: "Information Student");
            frame.setLocationRelativeTo(c: null);
            frame.setResizable(resizable: true);
            frame.setVisible(v: true);
        }
    });
}

```

➤ Ở đây chúng ta chỉ set nút add bấm được cho trang student (isDeleted = false)

- Đoạn mã bên dưới là sự kiện người dùng click vào nút Export thì sẽ xuất hiện bảng để chọn nơi người dùng muốn lưu file và xuất bảng học viên ra file Excel.

```

        Student student = listItem.get(index: i);

        row = sheet.createRow(i + 1);

        cell = row.createCell(columnIndex: 0, type: CellType.NUMERIC);
        cell.setCellValue(i + 1);

        cell = row.createCell(columnIndex: 1, type: CellType.NUMERIC);
        cell.setCellValue(d: student.getStudentCode());

        cell = row.createCell(columnIndex: 2, type: CellType.STRING);
        cell.setCellValue(string: student.getFullName());

        cell = row.createCell(columnIndex: 3, type: CellType.STRING);
        cell.setCellValue(string: student.getDateOfBirth().toString());

        cell = row.createCell(columnIndex: 4, type: CellType.STRING);
        cell.setCellValue(student.isSex() ? "Nam" : "Nữ");

        cell = row.createCell(columnIndex: 5, type: CellType.STRING);
        cell.setCellValue(string: student.getPhoneNumber());

        cell = row.createCell(columnIndex: 6, type: CellType.STRING);
        cell.setCellValue(string: student.getEmail());

        for (int j = 0; j < columns.length; j++) {
            CellStyle cellStyle = workbook.createCellStyle();
            cellStyle.setAlignment(ha: HorizontalAlignment.CENTER);
            row.getCell(column: j).setCellStyle(style: cellStyle);
            sheet.autoSizeColumn(column: j);
        }
    }
}

JFileChooser fileChooser = new JFileChooser();

// Thiết lập chế độ chỉ chọn đường dẫn để lưu file
fileChooser.setFileSelectionMode(mode: JFileChooser.DIRECTORIES_ONLY);

// Hiển thị hộp thoại chọn đường dẫn lưu file

btnExport.addMouseListener(new MouseAdapter() {
    public static FileOutputStream fis;

    @Override
    public void mouseClicked(MouseEvent e) {
        List<Student> listItem = studentService.getList(isDeleted);
        if (listItem.isEmpty()) {
            JOptionPane.showMessageDialog(parentComponent: null, message: "ds sinh vien rong!", title: "Notification", messageType: JOptionPane.
        } else {
            JButton btn = (JButton) e.getSource();
            btn.setBackground(bg: ColorBgExportController);
            //tạo 1 workbook mới của file excel
            XSSFWorkbook workbook = new XSSFWorkbook();
            XSSFSheet sheet = workbook.createSheet(sheetname: "Học viên");

            // Tạo dòng tiêu đề
            XSSFRow headerRow = sheet.createRow(rownum: 0);

            // Tạo các ô trong dòng tiêu đề
            String[] columns = {"STT", "Mã sv", "Họ tên", "Ngày sinh", "Giới tính", "SĐT", "Email"};
            for (int i = 0; i < columns.length; i++) {
                Cell headerCell = headerRow.createCell(columnIndex: i, type: CellType.STRING);
                headerCell.setCellValue(columns[i]);

                CellStyle headerCellStyle = workbook.createCellStyle();
                headerCellStyle.setAlignment(ha: HorizontalAlignment.CENTER);
                headerCell.setCellStyle(cs: headerCellStyle);
                sheet.autoSizeColumn(column: i);
            }

            XSSFRow row = null;
            Cell cell = null;

            int s = listItem.size();
            for (int i = 0; i < s; i++) {

```

```

int result = fileChooser.showSaveDialog(parent:null);
File fileToSave = null;
// Kiểm tra người dùng đã chọn OK hay Cancel
if (result == JFileChooser.APPROVE_OPTION) {
    // Lấy đường dẫn đã chọn
    File selectedFile = fileChooser.getSelectedFile();
    String filePath = selectedFile.getAbsolutePath();
    // Tạo tên file mới cho file Excel
    String excelFileName = "student" + ".xlsx";
    // Tạo đối tượng File để lưu file với tên mới
    fileToSave = new File(filePath + "/" + excelFileName);
    // Tạo đối tượng File để lưu file
    fileToSave = new File(filePath + "/student.xlsx");

    // Tiến hành lưu file
    // Ghi dữ liệu vào file và đóng file sau khi hoàn thành
    // FileOutputStream fis = null;
    try {
        fis = new FileOutputStream(file: fileToSave);
        SwingWorker<Void, Void> worker = new SwingWorker<Void, Void>() {
            @Override
            protected Void doInBackground() throws Exception {
                btnExport.setEnabled(b: false);
                workbook.write(stream: fis);
                return null;
            }
        };
        @Override
        protected void done() {
            try {
                JOptionPane.showMessageDialog(parentComponent: null, message: "File exported successfully!", title: "Not
                btnExport.setEnabled(b: true);
                fileChooser.setSelectedFile(file: null);
                Thread.currentThread().setName(name: "WorkerThreadExcel");
                // Không cần đóng luồng SwingWorker, nó sẽ tự động hoàn thành và kích hoạt done()
            } catch (Exception ex) {

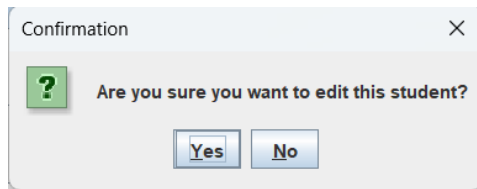
```

- o Đoạn mã code dưới đây là để khi người dùng click vào thông tin một học viên thì nó hiện ra ô thông báo là muốn chỉnh sửa hay xóa học viên hay không. Và nếu xác nhận đồng ý thì sẽ có 1Jframe hiện ra để người dùng thực hiện các thao tác chỉnh sửa hoặc xóa học viên.

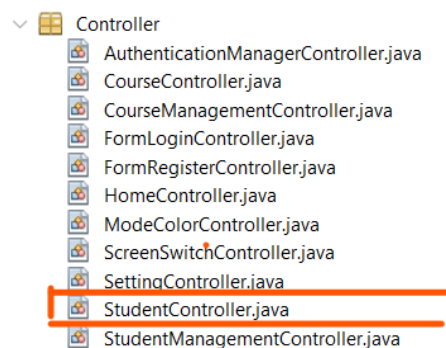
```

public void mouseClicked(MouseEvent e) {
    if (e.getClickCount() == 2 && tableStudent.getSelectedRow() != -1) {
        try {
            int selectedRowIndex = tableStudent.getSelectedRow();
            selectedRowIndex = tableStudent.convertRowIndexToModel(selectedRowIndex);
            Student student = createStudentFromTableData(model, rowIndex: selectedRowIndex);
            if (isDeleted) {
                đoạn xử lý cho student restoration
            } else {
                int dialogConfirm = JOptionPane.showConfirmDialog(parentComponent: null,
                    message: "Are you sure you want to edit this student?", title: "Confirmation", optionType: JOptionPane.YES_NO_OPTION);
                if (dialogConfirm == JOptionPane.YES_OPTION) {
                    StudentJFrame frame = new StudentJFrame(student);
                    frame.setLocationRelativeTo(c: null);
                    frame.setResizable(resizable:false);
                    frame.setTitle(title: "Thông tin sinh viên");
                    frame.setVisible(b: true);
                }
            }
        } catch (ParseException ex) {
            Logger.getLogger(name: StudentManagementController.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
        }
    }
}

```



○ File StudentController



- Đoạn mã code bên dưới là sự kiện khi ta click vào Submit thì toàn bộ các phần nhập sẽ được kiểm tra nếu nó thỏa mãn thì dữ liệu học viên sẽ được thêm hoặc cập nhật.

```
public void setEvent() {
    btnSubmit.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            // kiểm tra mã khóa học không được rỗng và phải là số
            if (jtfFullName.getText().trim().length() == 0
                || jtaGmail.getText().trim().length() == 0
                || jdcDateOfBirth.getDate() == null) {
                jlbMsg.setText(text: "Dữ liệu không được để trống.");
            } else if (!isValidEmail(email: jtaGmail.getText().trim())) {
                jlbMsg.setText(text: "định dạng không email hợp lệ");
            } else {
                student.setFullName(fullName: jtfFullName.getText().trim());
                student.setDateOfBirth(dateOfBirth: convertDateToDateTime(jdcDateOfBirth.getDate()));
                student.setPhoneNumber(phoneNumber: jtfNumber.getText());
                student.setGmail(address: jtaGmail.getText());
                student.setSex(sex: jrmMale.isSelected());
                student.setStatus(status: true);
                if (showDialog()) {
                    int lastId = studentService.createOrUpdate(student);
                    if (lastId != 0) {
                        StudentJPanel.pageStudentController.renderData(isDeleted: false);
                        StudentJPanel.pageStudentController.setEvent();
                        jlbMsg.setText(text: "Lưu dữ liệu thành công!");
                        StudentJFrame.dispose();
                    } else {
                        jlbMsg.setText(text: "Có lỗi xảy ra, vui lòng thử lại!");
                    }
                }
            }
        }
    });
}

private boolean isValidEmail(String email) {
    // Biểu thức chính quy kiểm tra định dạng email
    String emailPattern = "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";
}
```

- Đoạn mã code bên dưới là để thực hiện việc xóa mềm đi một học viên.

```
btnDelete.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        int dialogResult = JOptionPane.showConfirmDialog(parentComponent: null,
            message: "Are you sure you want to delete this student?", title: "Confirmation", optionType: JOptionPane.YES_NO_OPTION)
        if (dialogResult == JOptionPane.YES_OPTION) {
            int res = studentService.softDelete(student);
            if (res == 0) {
                jlbMsg.setText(text: "Error: student cannot be deleted!");
            } else {
                StudentJPanel.pageStudentController.renderData(isDeleted:false);
                studentRestorationPageController.setEven();
                jlbMsg.setText(text: "Student deleted successfully!");
                StudentJFrame.setVisible(b: false);
                StudentJFrame.dispose();
            }
        }
    }
});
```

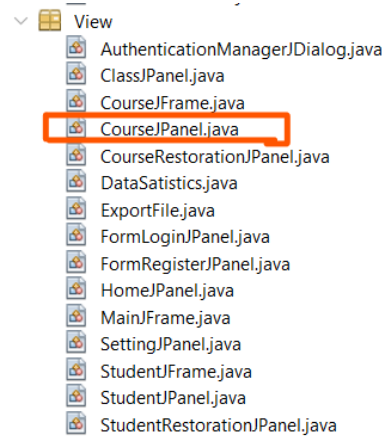
2.6. Trang Course Management

2.6.1. Giao diện

<input type="text"/> Export Add					
Mã khóa học	Tên khóa	Mô tả	Thời lượng(_h)	Giá tiền(_\$)	Ngày đăng
3	Vue Js cơ bản	Hỗ trợ xin việc	1d 6h 0m	2000	Apr 19, 2023
61	Typescript		2d 10h 20m	500	Apr 20, 2023
53	tailwind css		6h 40m	20	Apr 15, 2023
57	SQL		8d 8h 0m	57	Apr 17, 2023
4	Ruby on Rail Nâng cao	khó học	16d 16h 0m	4	Apr 28, 2023
1	Ruby On Rail		1d 9h 20m	3000	Apr 3, 2023
56	Python		2h	500	Apr 17, 2023
55	PHP		30m	200	Apr 17, 2023
52	Next js cơ bản		2d 2h 0m	100	Apr 2, 2023
58	Javascript		1d 6h 0m	2000	Apr 17, 2023
2	Java	Cấp chứng chỉ	2d 2h 0m	2	Apr 4, 2023
51	Html,css cơ bản	dành cho người mới	3d 11h 20m	50	Apr 26, 2023
7	Cấu trúc dữ liệu giải thuật	khó học	1d 9h 20m	100	Apr 10, 2023
5	Bootstrap4 cơ bản	miễn phí	2h	0	Apr 11, 2023
59	Bootstrap		12d 12h 0m	5000	Apr 17, 2023
50	Angular		1d 1h 0m	300	Apr 20, 2023

2.6.2 Xây dựng giao diện và tương tác

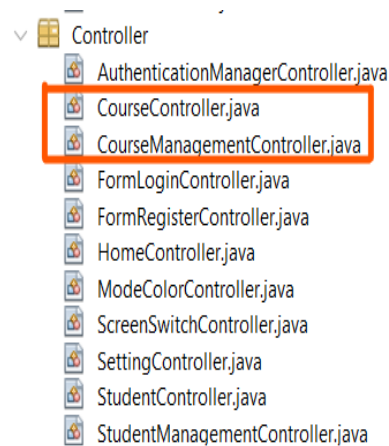
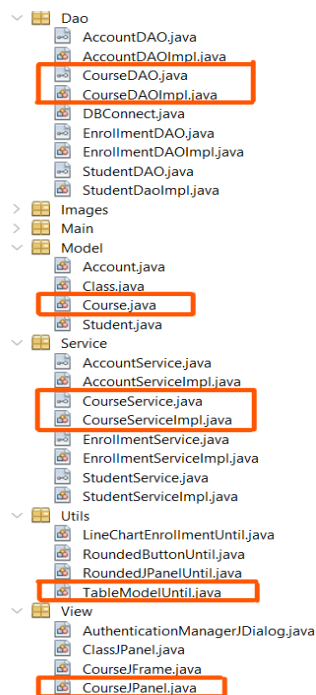
a) Xây dựng giao diện



- Trên là File thiết kế ra giao diện và cách thiết kế giống phần thiết kế giao diện của trang StudentManagement

b) Xây dựng tương tác

- Trang CourseManagement cũng được tổ chức và code tương tự trang StudentManagement. Nó bao gồm các File sau.



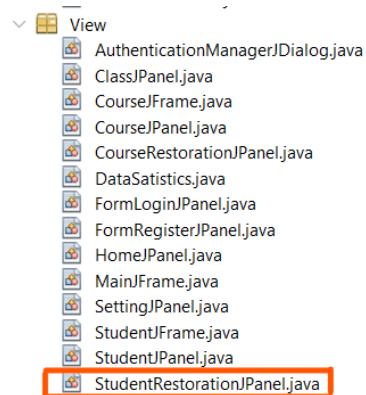
2.7. Trang Student Restoration

2.7.1. Giao diện

Stt	Mã sv	Họ tên	Ngày sinh	Giới tính	SDT	Gmail
1	292	Nguyễn Lệ Chi	2023-04-05	Nữ	+84292939393	
2	302	Vũ Văn Cường	2023-03-01	Nam	+84029299233	
3	494	Nguyễn Minh Châu	2023-04-05	Nữ	+84292939330	
4	838	Hoàng Hồ Gia Khánh	2023-03-14	Nam	+84020020222	
5	939	Vũ Trọng Hoàng	2023-04-12	Nam	+84292929292	
6	1393	Nguyễn Trà My	2023-04-05	Nam	+84292939393	
7	100000001	test 1	2023-04-06	Nữ	+840209292	test@gmail.com

2.7.2. Xây dựng giao diện và tương tác

a) Xây dựng giao diện



- File trên là nơi có các đoạn mã code xây dựng nên giao diện và giao diện nó tương tự giao diện của trang StudentManagement và chỉ khác nó bỏ nút Add

b) Xây dựng tương tác

- Sử dụng lại File StudentManagementController nhưng tham số truyền vào là true.

```
public StudentRestorationJPanel() {
    this.init();
    studentRestorationPageController = new StudentManagementController(jPanelView: jpnInfor, btnAdd, jTextFieldSearch: jtfSearch, btnExport);
    studentRestorationPageController.renderData(isDeleted:true);
    studentRestorationPageController.setEven();
}
```



```

public void mouseClicked(MouseEvent e) {
    if (e.getClickCount() == 2 && tableStudent.getSelectedRow() != -1) {
        try {
            int selectedRowIndex = tableStudent.getSelectedRow();
            selectedRowIndex = tableStudent.convertRowIndexToModel(selectedRowIndex);
            Student student = student = createStudentFromTableData(model, selectedRowIndex);
            if (isDeleted) {
                đoạn xử lý cho student restoration
            } else {
                int dialogConfirm = JOptionPane.showConfirmDialog(parentComponent: null,
                    message: "Are you sure you want to edit this student?", title: "Confirmation", optionType: JOptionPane.YES_NO_OPTION);
                if (dialogConfirm == JOptionPane.YES_OPTION) {
                    StudentJFrame frame = new StudentJFrame(student);
                    frame.setLocationRelativeTo(c: null);
                    frame.setResizable(resizable: false);
                    frame.setTitle(title: "Thông tin sinh viên");
                    frame.setVisible(v: true);
                }
            }
        } catch (ParseException ex) {
            Logger.getLogger(name: StudentManagementController.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
        }
    }
}

```

```

Object[] options = {"Recover", "Hard Delete", "Cancel"};
int dialogAction = JOptionPane.showOptionDialog(parentComponent: null,
    message: "Do you want to recover or hard delete this student?",
    title: "Confirmation",
    optionType: JOptionPane.YES_NO_CANCEL_OPTION,
    messageType: JOptionPane.QUESTION_MESSAGE,
    icon: null,
    options,
    options[2]);

```

```

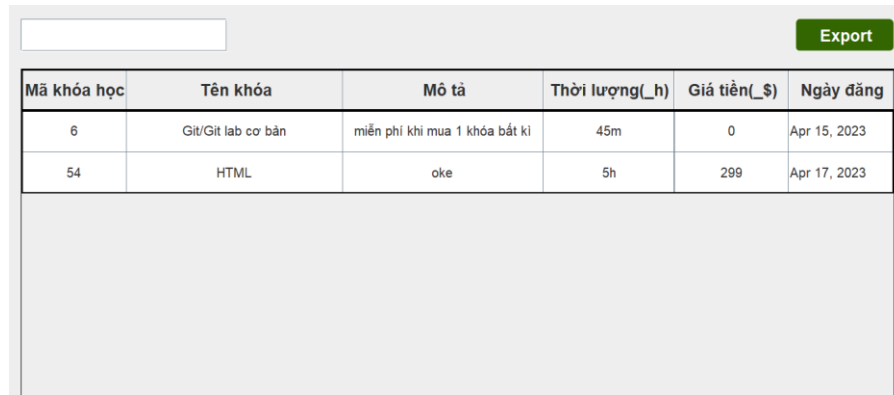
switch (dialogAction) {
    case JOptionPane.YES_OPTION -> {
        int dialogConfirmRestore = JOptionPane.showConfirmDialog(parentComponent: null,
            message: "Do you definitely want to restore?", title: "Confirmation", optionType: JOptionPane.YES_NO_OPTION);
        if (dialogConfirmRestore == JOptionPane.YES_OPTION) {
            int res = studentService.recoverSoftDeletedStudent(student);
            if (res == 1) {
                // recover is successful
                studentRestorationPageController.renderData(isDeleted: true);
                studentRestorationPageController.setEven();
                JOptionPane.showMessageDialog(parentComponent: null, message: "Restore successful", title: "Success", messageType: JOptionPane.INFORMATION_MESSAGE);
            } else {
                // recover is failed
                JOptionPane.showMessageDialog(parentComponent: null, message: "Restore failed", title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    case JOptionPane.NO_OPTION -> {
        int dialogConfirmHardDelete = JOptionPane.showConfirmDialog(parentComponent: null,
            message: "Note: After deletion, it will not be recoverable?", title: "Confirmation", optionType: JOptionPane.YES_NO_OPTION);
        if (dialogConfirmHardDelete == JOptionPane.YES_OPTION) {
            int res = studentService.hardDelete(student);
            if (res == 1) {
                // Hard delete is successful
                studentRestorationPageController.renderData(isDeleted: true);
                studentRestorationPageController.setEven();
                jpnView.removeAll();
                JOptionPane.showMessageDialog(parentComponent: null, message: "Deletion successful", title: "Success", messageType: JOptionPane.INFORMATION_MESSAGE);
            } else {
                // Hard delete failed
                JOptionPane.showMessageDialog(parentComponent: null, message: "Deletion failed", title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    default -> {
    }
}

```

- Phương thức `JOptionPane.showOptionDialog` hiển thị một hộp thoại với ba tùy chọn - "Phục hồi", "Xóa vĩnh viễn" và "Hủy". Phương thức trả về một giá trị số nguyên đại diện cho tùy chọn được chọn bởi người dùng. Sau đó, câu lệnh `switch` kiểm tra giá trị số nguyên được trả về bởi phương thức `showOptionDialog` và thực thi khối lệnh tương ứng với giá trị trả về.

2.8. Trang Course Restoration

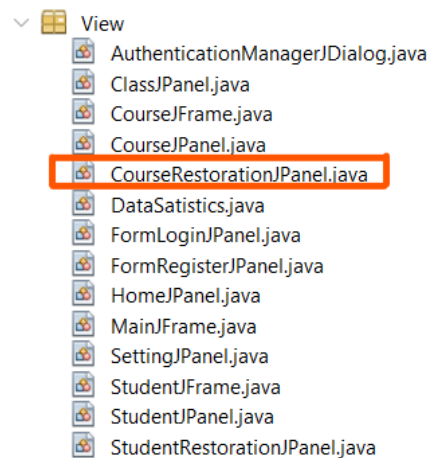
2.8.1. Giao diện



Mã khóa học	Tên khóa	Mô tả	Thời lượng(_h)	Giá tiền(_\$)	Ngày đăng
6	Git/Git lab cơ bản	miễn phí khi mua 1 khóa bất kì	45m	0	Apr 15, 2023
54	HTML	oke	5h	299	Apr 17, 2023

2.8.2 Xây dựng giao diện và tương tác

a) Xây dựng giao diện



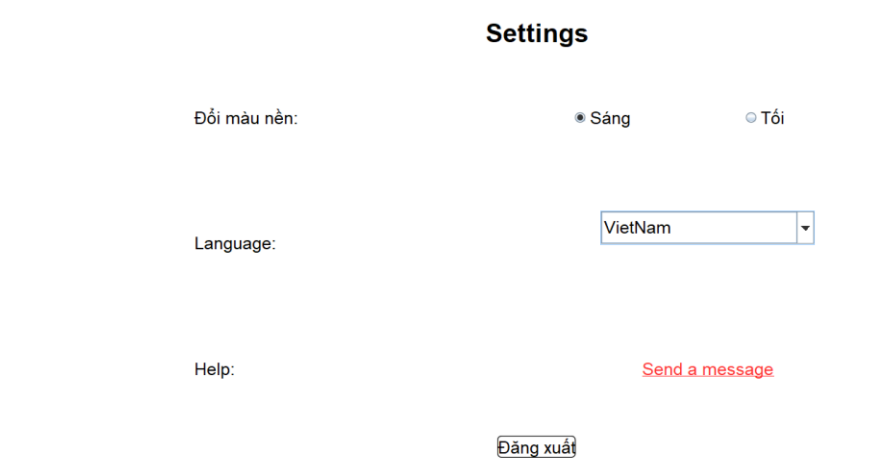
File trên là nơi có các đoạn mã code xây dựng nên giao diện và giao diện nó tương tự giao diện của trang CourseManagement và chỉ khác nó bỏ nút Add

b) Xây dựng tương tác

Sử dụng lại file CourseManagementController là logic tương tự như bên StudentManagementController.

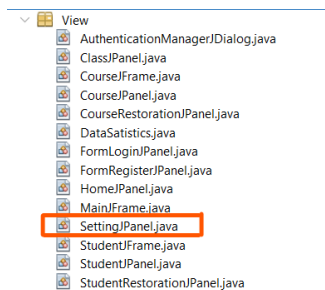
2.9. Trang Setting

2.9.1 Giao diện



2.9.2. Xây dựng giao diện và tương tác

a) Xây dựng giao diện

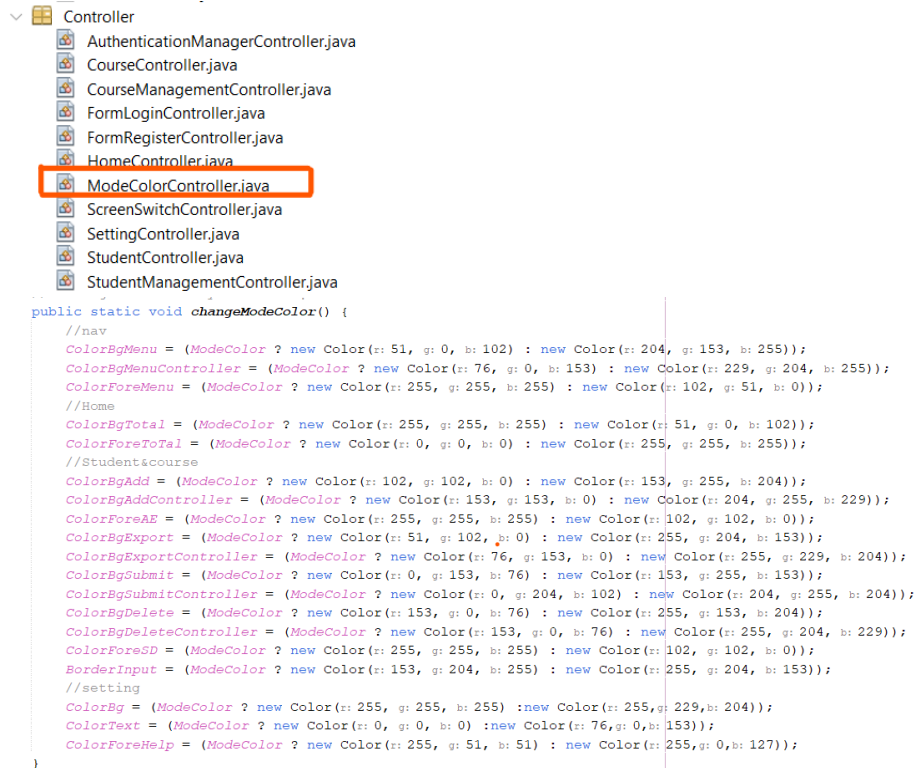


- Trên là File trong đó có các đoạn mã code để code lên giao diện trang Setting.

Trang Setting có một số chức năng chính là: đổi màu nền, đăng xuất

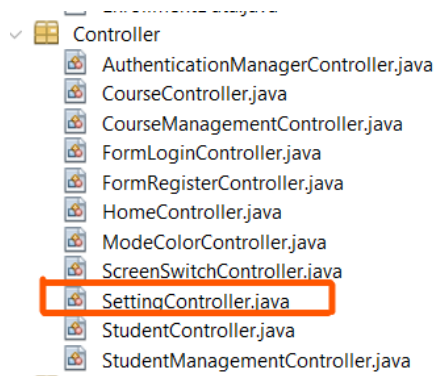
b) Xây dựng tương tác

- File ModeColorController:



- Đoạn mã code trên là để lưu màu hiện tại của các thành phần trong app và màu nó sẽ đổi sau khi đổi màu nền

○ File SettingController:



- Đoạn mã code dưới đây là xét sự kiện click vào 2 JradioButton Bright và Dark thì các thành phần trong app sẽ đổi màu

```

JrBright.addActionListener((ActionEvent e) -> {
    ModeColorController.ModeColor = true;
    changeModeColor();
    ScreenSwitchController.currentPage = "Setting";
    mainJFrame.resetFrame();
    mainJFrame.setExtendedState(state: JFrame.MAXIMIZED_BOTH);
    // Đặt JFrame không cho phép phóng to/thu nhỏ cửa sổ
});

JrDark.addActionListener((ActionEvent e) -> {
    ModeColorController.ModeColor = false;
    changeModeColor();
    ScreenSwitchController.currentPage = "Setting";
    mainJFrame.resetFrame();
    mainJFrame.setExtendedState(state: JFrame.MAXIMIZED_BOTH);
    // Đặt JFrame không cho phép phóng to/thu nhỏ cửa sổ
});

```

- Reload Frame hiện tại để cập nhật màu mới
- Đoạn mã code dưới là xét sự kiện khi người dùng click vào nút “Đăng xuất” thì nó trở về trang Đăng nhập

```

btnLogout.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        int dialogConfirmHardDelete = JOptionPane.showConfirmDialog(parentComponent: null, message: "Note: You want to sign out?", title
        if (dialogConfirmHardDelete == JOptionPane.YES_OPTION) {
            mainJFrame.dispose();
            AuthenticationManagerJDialog authenticationManagerJDialog = new AuthenticationManagerJDialog();
        }
    }
});

```

- Xóa Frame hiện tại và sổ ra trang login để đăng nhập lại từ đầu

Chương 3: Kết luận và hướng phát triển

Nhóm chúng em đã hoàn thành “**App quản lí học viên**”. Trong đó có một số ưu điểm và những nhược điểm sau :

+ Ưu điểm : App đã lấy dữ liệu trên database xuống có các trang học viên, khóa học (trong đó có các chức năng thêm, sửa, xóa mềm, tìm kiếm và xuất ra file excel). Ngoài ra app còn có các trang để khôi phục học viên và khóa học đã bị xóa. Thêm vào đó app còn có các chức năng để đổi màu, đăng xuất khỏi app. Giao diện app thân thiện, dễ dàng sử dụng.

Code dễ bảo trì do đặt class, biến chuẩn và 1 hàm chỉ làm 1 nhiệm vụ

+ Nhược điểm : chưa có hỗ trợ giao diện cho nhiều loại màn hình

- Nhóm chúng em sẽ có những cải tiến cho chương trình. Chúng em tin rằng những bản nâng cấp chương trình sau đó có thể ứng dụng vào việc quản lí các học viên ở các trung tâm dạy học

➤ Trong thời gian thực hiện dự án , nhờ sự chỉ bảo của thầy **Vũ Huân** và sự giúp đỡ của các bạn, nhóm chúng em đã thu được nhiều kết quả trong việc học lập trình Java.

➤ Do thời gian và khả năng có hạn nên bài tập lớn của nhóm chúng em còn

rất nhiều thiếu sót, chúng em rất mong nhận được sự góp ý, giúp đỡ của thầy và các bạn để bài tập của chúng em được hoàn thiện hơn. Chúng em xin cảm ơn !

Chương 4. Tài liệu tham khảo

1. [How to draw Line Charts in Java Swing - Stack Overflow](#)
2. <https://cuongquach.com/ebook-giao-trinh-sql-tran-nguyen-phong-pdf.html>
3. <https://chat.openai.com/>
4. [Giao diện app](#)
5. [Mô hình mvc](#)

