

PRAKTIKUM REKAYASA PERANGKAT LUNAK 2



MANUAL BOOK

“Aplikasi Manajemen Buku Perpustakaan”

Nama Anggota :

- | | |
|--------------------------------|------------|
| 1. Ali Akbar Said | (50421119) |
| 2. Dewa Gede Budi Dharma Putra | (50421358) |
| 3. Ikbal Amin | (50421632) |

Kelas	: 4IA03
Fakultas	: Teknologi Industri
Jurusan	: Informatika
PJ	: Nurul Hidayatullah

Ditulis Guna Melengkapi Sebagian Syarat Praktikum

Rekayasa Perangkat Lunak 2

Universitas Gunadarma

2024

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perpustakaan memiliki peran penting sebagai pusat informasi yang menyediakan berbagai koleksi buku untuk mendukung pembelajaran, penelitian, dan pengembangan literasi masyarakat. Salah satu aspek utama dalam pengelolaan perpustakaan adalah pencatatan informasi buku, seperti judul, pengarang, penerbit, dan tahun terbit. Informasi ini sangat penting untuk mempermudah pengelolaan koleksi, pencarian, serta pengelompokan buku. Namun, metode pencatatan manual yang masih banyak digunakan di beberapa perpustakaan sering kali menghadapi kendala, seperti risiko kehilangan data, kesalahan pencatatan, dan sulitnya mengakses informasi dengan cepat.

Seiring perkembangan teknologi, aplikasi pencatatan informasi buku menjadi solusi yang efektif untuk menggantikan metode manual yang kurang efisien. Aplikasi ini dirancang khusus untuk mencatat dan mengelola data buku secara digital, sehingga pengelolaan informasi menjadi lebih terstruktur dan mudah diakses. Dengan aplikasi ini, pengelola perpustakaan dapat mencatat informasi penting buku dengan cepat dan akurat. Selain itu, data yang tersimpan secara digital lebih aman dibandingkan catatan fisik yang rentan terhadap kerusakan atau kehilangan.

Penggunaan aplikasi pencatatan buku memberikan banyak manfaat, seperti meningkatkan efisiensi kerja, mengurangi risiko kesalahan pencatatan, dan mempermudah proses pencarian data. Dengan fitur yang terintegrasi, aplikasi ini memungkinkan pengelola untuk mengakses informasi buku kapan saja tanpa perlu mencari secara manual. Hal ini juga membantu perpustakaan dalam memberikan layanan yang lebih responsif kepada pengguna.

Melalui digitalisasi proses pencatatan buku, perpustakaan dapat meningkatkan kualitas pengelolaan koleksi dan layanannya. Pengembangan aplikasi pencatatan informasi buku ini menjadi langkah penting untuk menjawab kebutuhan modern akan pengelolaan data yang efisien dan mendukung peran perpustakaan sebagai pusat informasi yang handal dan inovatif.

1.2 Tujuan

Dalam penulisan ini bertujuan untuk mempermudah proses pencatatan dan pengelolaan data buku perpustakaan agar lebih efisien, akurat, dan terstruktur. Aplikasi ini memungkinkan perpustakaan mencatat informasi penting, seperti judul buku, pengarang, penerbit, dan tahun terbit, dengan lebih mudah dan aman. Digitalisasi proses pencatatan ini dirancang untuk mengurangi risiko kesalahan, kehilangan data, serta mempercepat akses informasi. Selain itu, aplikasi ini bertujuan untuk mendukung pengelolaan koleksi buku secara modern, sehingga perpustakaan dapat beroperasi dengan lebih efektif sesuai dengan tuntutan perkembangan teknologi.

Dalam penulisan ini bertujuan untuk mempermudah akses data bagi pengelola dan pengguna perpustakaan. Melalui fitur pencarian cepat dan pengelolaan data yang terorganisir, aplikasi ini membantu pengelola menemukan informasi buku secara efisien serta memberikan layanan yang lebih baik kepada pengguna. Selain itu, aplikasi ini mendukung penyusunan laporan data yang relevan dan akurat untuk evaluasi pengelolaan koleksi. Dengan implementasi aplikasi ini, perpustakaan dapat meningkatkan produktivitas operasional sekaligus memenuhi kebutuhan informasi pengguna secara lebih profesional dan responsif.

BAB II

PEMBAHASAN

2.1 NetBeans

NetBeans adalah suatu *Integrated Development Environment* (IDE) atau serambi pengembangan perangkat lunak yang ditulis dalam bahasa pemrograman Java. IDE ini digunakan secara luas untuk membangun aplikasi berbasis Java serta aplikasi dalam berbagai bahasa pemrograman lainnya, seperti PHP, HTML5, C++, dan Python. Dengan antarmuka yang ramah pengguna serta dukungan fitur yang kaya, NetBeans menjadi pilihan populer bagi pengembang perangkat lunak untuk merancang, menulis, menguji, dan menyebarkan aplikasi.

2.2 Spring

Spring adalah sebuah *framework* pengembangan aplikasi yang bersifat open-source, dirancang untuk membantu pengembang membangun aplikasi berbasis Java secara lebih cepat dan efisien. Framework ini menyediakan berbagai fitur dan alat untuk menyederhanakan proses pengembangan aplikasi, terutama aplikasi berbasis enterprise. Dengan menggunakan pendekatan berbasis *inversion of control* (IoC) dan *dependency injection* (DI), Spring memungkinkan pengembang mengelola komponen-komponen aplikasi secara lebih modular dan terorganisir.

2.3 Hibernate

Hibernate adalah sebuah fitur pada sistem operasi (OS) yang terdapat pada perangkat laptop atau komputer yang memungkinkan perangkat masuk ke mode hemat daya tanpa menutup aplikasi atau dokumen yang sedang digunakan. Dalam mode Hibernate, seluruh data dari sesi kerja, termasuk aplikasi yang terbuka dan dokumen yang sedang dikerjakan, disimpan ke dalam disk (biasanya pada file khusus di penyimpanan) sebelum perangkat dimatikan sepenuhnya. Ketika perangkat dihidupkan kembali, sesi kerja akan dipulihkan persis seperti saat perangkat terakhir kali digunakan.

2.3.1.1 Fungsi Hibernate

Fungsi hibernate tentu sudah jelas yaitu menyimpan semua kerja memori atau menyimpan semua kegiatan yang dilakukan di laptop ketika laptop dimatikan, dan akan dikembalikan seperti semula ketika sebelum di hibernate.

2.3.1.2 Manfaat Hibernate

Ada beberapa manfaat hibernate yaitu :

1. Pengelolaan Koneksi Database yang Efisien

Hibernate memudahkan pengelolaan koneksi database dengan menyediakan *Object-Relational Mapping* (ORM), yang memungkinkan pengembang untuk bekerja dengan objek Java tanpa harus menulis kode SQL secara manual. Ini mengurangi kompleksitas pengelolaan koneksi dan meningkatkan efisiensi pengambilan data.

2. Penyederhanaan Kode

Dengan Hibernate, pengembang dapat fokus pada logika aplikasi dan tidak perlu menghabiskan banyak waktu menulis kode SQL untuk setiap interaksi dengan database. Hibernate otomatis mengonversi objek Java menjadi entitas database, menyederhanakan pengembangan aplikasi.

3. Portabilitas

Hibernate mendukung berbagai sistem manajemen basis data (DBMS), sehingga aplikasi yang dibangun menggunakan Hibernate lebih portabel. Pengembang tidak perlu mengubah banyak kode aplikasi saat berpindah ke database yang berbeda, karena Hibernate menangani perbedaan sintaksis SQL antara DBMS yang berbeda.

4. Manajemen Transaksi yang Lebih Baik

Hibernate menyediakan pengelolaan transaksi yang lebih baik dan mudah. Dengan menggunakan transaksi dalam Hibernate, pengembang dapat memastikan konsistensi data dan mengelola perubahan data secara atomik, yang mengurangi risiko inkonsistensi dan kerusakan data.

5. Caching untuk Kinerja yang Lebih Baik

Hibernate menyediakan fitur *caching*, yang memungkinkan data yang sering diakses untuk disimpan di dalam memori. Hal ini dapat mengurangi beban pada database dan meningkatkan kinerja aplikasi, terutama dalam aplikasi dengan volume data yang besar.

6. Pengelolaan Relasi Antar Tabel dengan Mudah

Hibernate memungkinkan pengembang untuk dengan mudah mengelola relasi antar entitas (tabel) menggunakan anotasi atau file konfigurasi. Ini termasuk pengelolaan relasi satu-ke-banyak, banyak-ke-banyak, dan satu-ke-satu tanpa perlu menulis banyak SQL untuk join.

7. Peningkatan Produktivitas

Dengan berbagai fitur otomatis seperti pengelolaan objek dan koneksi database, Hibernate dapat mengurangi waktu pengembangan aplikasi. Pengembang dapat lebih fokus pada logika aplikasi dan fitur lainnya, sehingga meningkatkan produktivitas secara keseluruhan.

Dengan semua manfaat ini, Hibernate menjadi pilihan utama dalam pengembangan aplikasi berbasis Java yang membutuhkan pengelolaan data yang efisien dan mudah dikelola.

2.3.2 Kerugian dan Masalah Hibernate

1. Kinerja yang Lebih Lambat pada Aplikasi Kecil

Meskipun Hibernate dapat meningkatkan kinerja pada aplikasi besar, pada aplikasi kecil atau aplikasi dengan volume data yang rendah, penggunaan Hibernate justru bisa memperlambat kinerja. Hal ini disebabkan oleh overhead yang terkait dengan konversi objek ke entitas database, pengelolaan sesi, dan transaksi yang lebih kompleks dibandingkan dengan menggunakan SQL langsung.

2. Penggunaan Memori yang Lebih Tinggi

Hibernate membutuhkan lebih banyak memori karena ia memuat seluruh objek yang terlibat dalam sesi dan menggunakan caching untuk meningkatkan kinerja. Pada aplikasi dengan data yang sangat besar, hal ini bisa mempengaruhi penggunaan memori dan dapat menyebabkan masalah kinerja.

3. Kurva Pembelajaran yang Cukup Curam

Meskipun Hibernate menawarkan banyak keuntungan, ia memiliki kurva pembelajaran yang cukup curam, terutama bagi pengembang yang belum berpengalaman dengan ORM atau konsep-konsep seperti *lazy loading*, *caching*, dan *transactions*. Untuk menggunakan Hibernate dengan efektif, pengembang perlu memahami arsitektur dan cara kerjanya dengan baik.

4. Masalah dengan Query Kompleks

Meskipun Hibernate menyediakan kemampuan untuk menjalankan query menggunakan HQL (Hibernate Query Language), query yang sangat kompleks atau sangat khusus sering kali lebih sulit dikelola. Dalam beberapa kasus, Hibernate tidak memberikan kontrol penuh terhadap SQL yang dihasilkan, yang dapat menyebabkan masalah kinerja atau kesalahan dalam pengambilan data.

5. **Overhead dalam Pengelolaan Transaksi**

Hibernate secara otomatis mengelola transaksi dan sesi, yang bisa menjadi keuntungan dalam banyak kasus, namun pada situasi tertentu, pengelolaan transaksi otomatis ini bisa menjadi lebih lambat atau lebih kompleks daripada menggunakan transaksi secara manual dalam kode SQL biasa, terutama pada aplikasi dengan beban transaksi yang sangat tinggi.

6. **Kesulitan dalam Debugging**

Karena Hibernate mengelola banyak aspek database dan konversi objek, debugging aplikasi yang menggunakan Hibernate bisa lebih rumit. Jika terjadi masalah dalam pengambilan data atau kinerja, melacak masalah di lapisan ORM bisa lebih sulit dibandingkan dengan men-debug kode SQL yang lebih sederhana dan langsung.

7. **Ketergantungan pada Hibernate Framework**

Menggunakan Hibernate berarti aplikasi menjadi bergantung pada framework tersebut. Jika suatu saat ada perubahan besar dalam versi Hibernate atau keputusan untuk menggantinya dengan teknologi lain, migrasi atau refactoring kode dapat menjadi pekerjaan yang cukup berat, terutama jika aplikasi sudah sangat bergantung pada fitur-fitur Hibernate.

2.4 **ORM(Object Relational Mapping)**

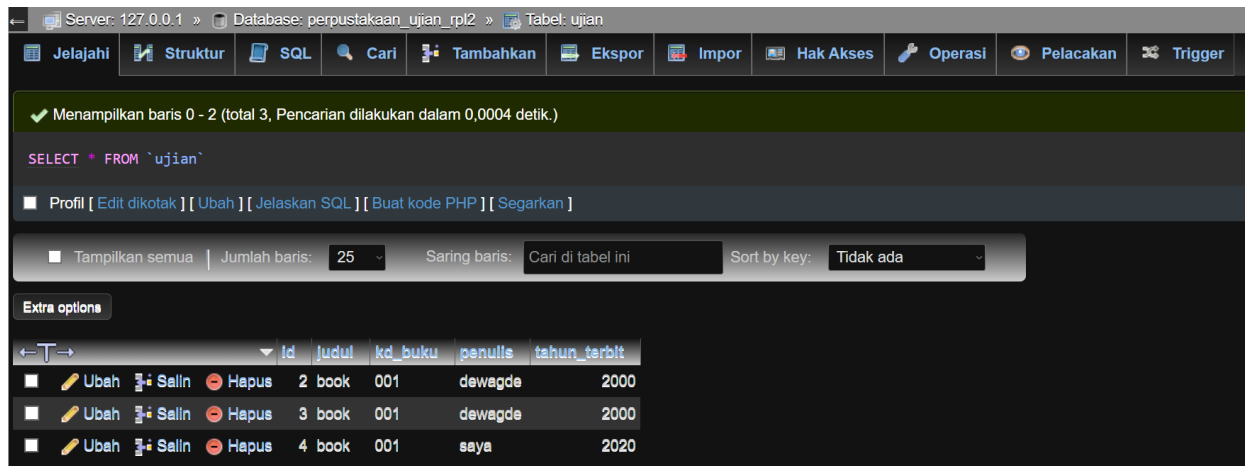
ORM (Object-Relational Mapping) merupakan metode pemrograman yang digunakan untuk mengkonversi data dari lingkungan bahasa pemrograman berorientasi objek (OOP) ke dalam format yang dapat disimpan dan dikelola dalam basis data relasional. Dengan ORM, objek-objek dalam aplikasi yang dikembangkan menggunakan bahasa pemrograman berorientasi objek, seperti Java, C#, atau Python, dapat dipetakan langsung ke dalam tabel-tabel pada database. Hal ini memungkinkan pengembang untuk berinteraksi dengan data menggunakan konsep objek, tanpa harus menulis SQL secara manual untuk melakukan operasi seperti menyimpan, mengambil, memperbarui, atau menghapus data.

ORM menyederhanakan proses komunikasi antara aplikasi berbasis objek dan database relasional, karena mengelola konversi antara objek dan tabel secara otomatis. Ini mengurangi kerumitan dan risiko kesalahan yang dapat terjadi dalam penulisan SQL manual. Hibernate, sebagai contoh populer dari ORM, memungkinkan pengembang untuk bekerja dengan objek Java dan secara otomatis mengelola pemetaan ke struktur database yang sesuai, serta menyediakan fitur-fitur seperti manajemen transaksi, pengambilan data yang efisien, dan dukungan terhadap hubungan antar entitas dalam database.

BAB III

ANALISA DAN PERANCANGAN

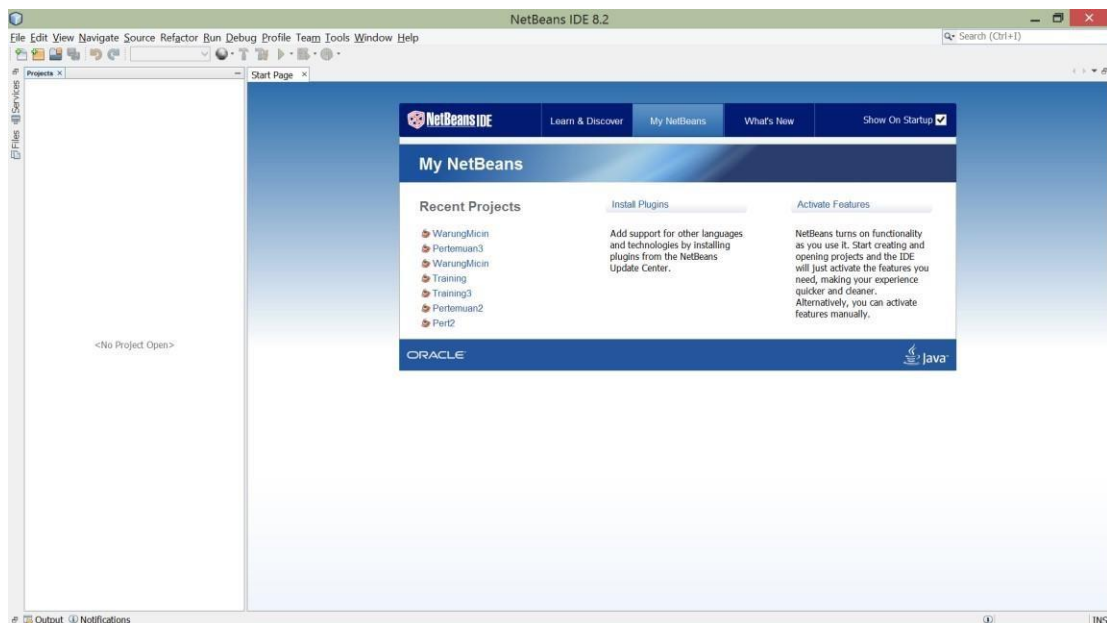
Project yang akan kita buat yaitu bertema “Aplikasi Manajemen Buku Perpustakaan”. Berikut ini merupakan database yang digunakan dalam project. Database yang digunakan bernama perpustakaan_ujian_rpl2 dengan tabel bernama ujian.



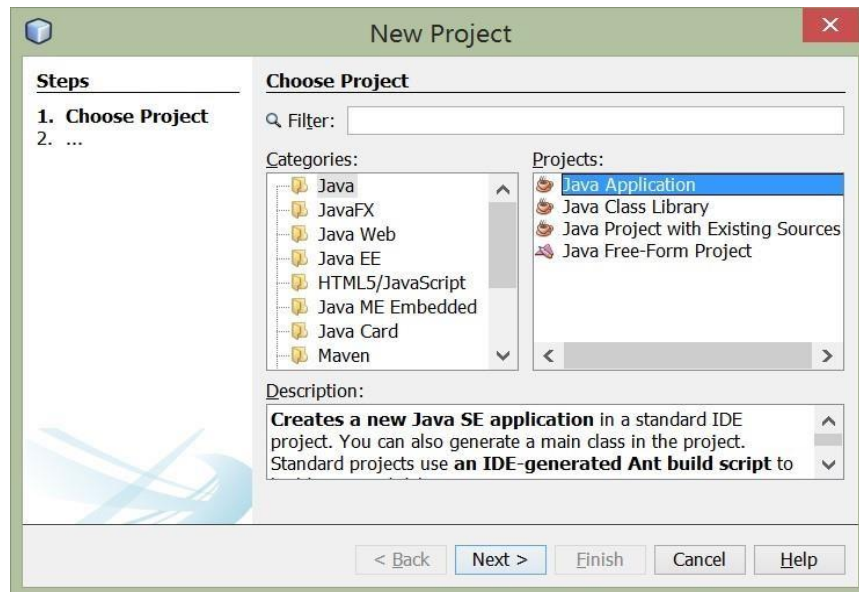
The screenshot shows a database management interface with a toolbar at the top containing buttons like 'Jelajahi', 'Struktur', 'SQL', 'Cari', 'Tambahkan', 'Ekspor', 'Impor', 'Hak Akses', 'Operasi', 'Pelacakan', and 'Trigger'. Below the toolbar, a status bar indicates 'Menampilkan baris 0 - 2 (total 3, Pencarian dilakukan dalam 0,0004 detik.)'. The main area displays a SQL query: `SELECT * FROM `ujian``. Below the query, there are links for 'Profil', 'Edit kotak', 'Ubah', 'Jelaskan SQL', 'Buat kode PHP', and 'Segarkan'. A filter bar shows 'Tampilkan semua', 'Jumlah baris: 25', 'Saring baris: Cari di tabel ini', and 'Sort by key: Tidak ada'. An 'Extra options' button is also present. The table data is as follows:

Ubah	Salin	Hapus	2	book	001	dewagde	2000
Ubah	Salin	Hapus	3	book	001	dewagde	2000
Ubah	Salin	Hapus	4	book	001	saya	2020

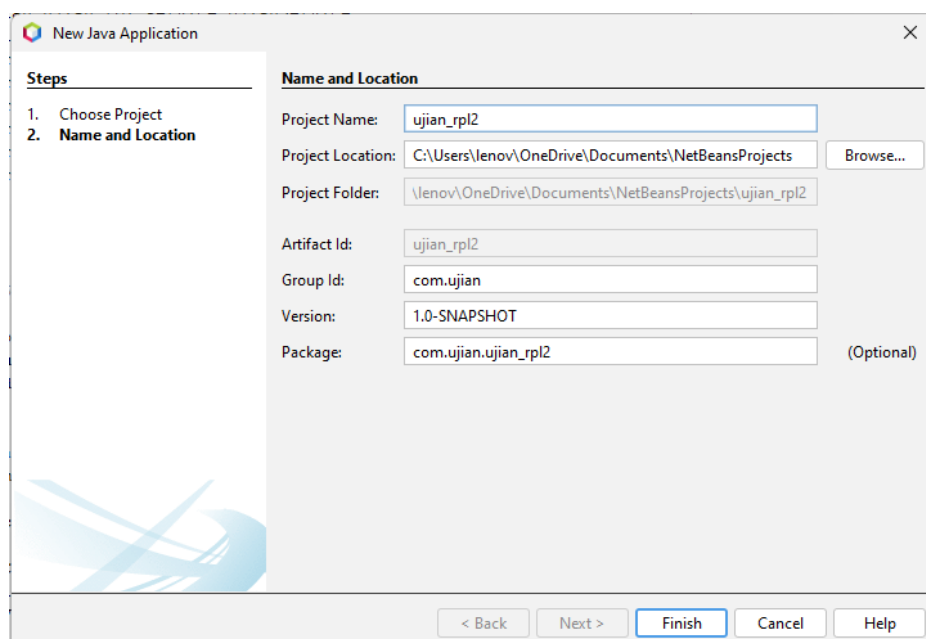
Dibawah ini merupakan tampilan awal pada Netbeans.



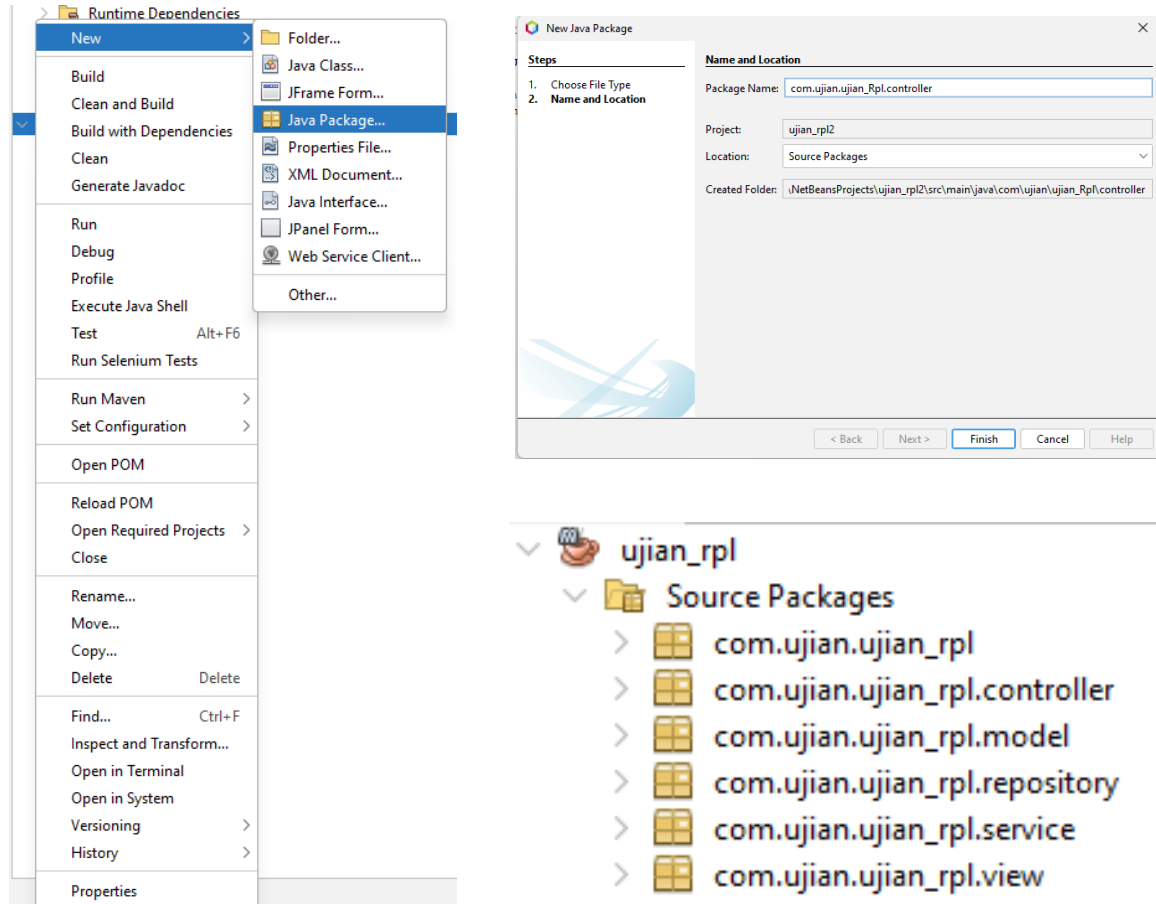
Langkah selanjutnya click “ new project “, pilih Java Application terus Next sperti gambar dibawah ini.



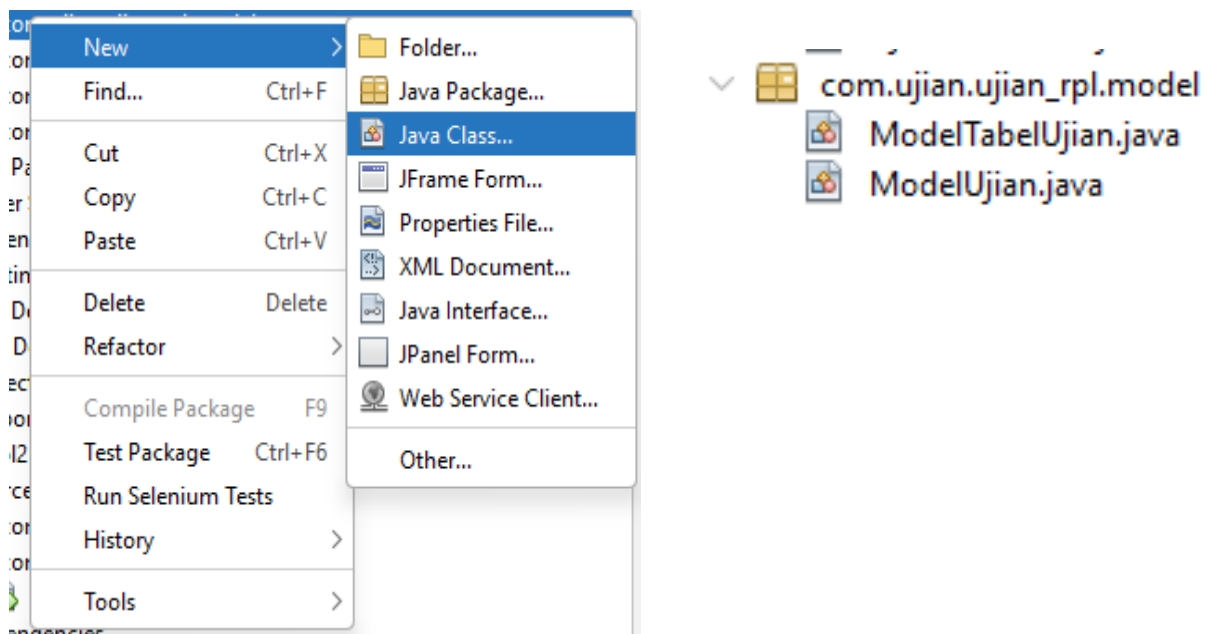
Kemudian beri nama project dan lokasi penyimpanan project, terus klik finish.

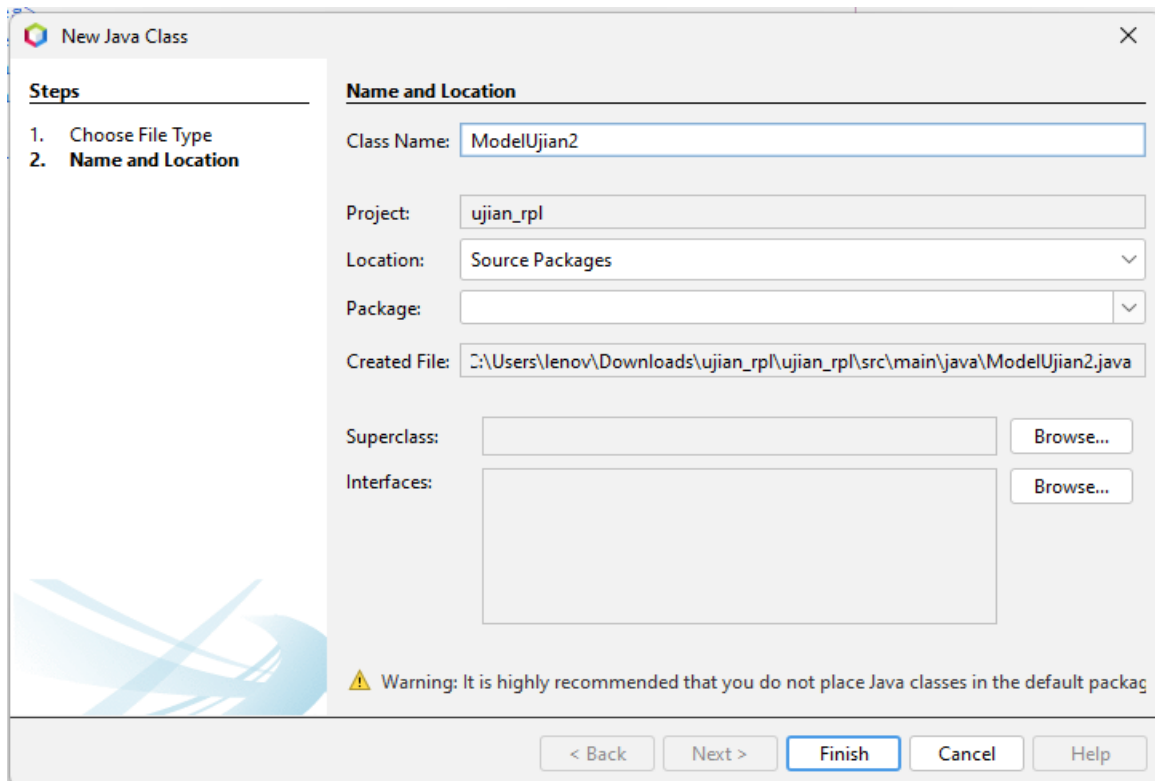


Selanjutnya membuat package untuk controller, model, repository service dan view.

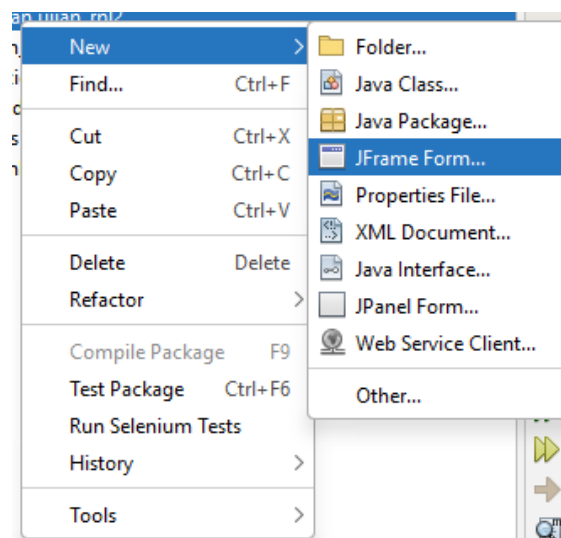


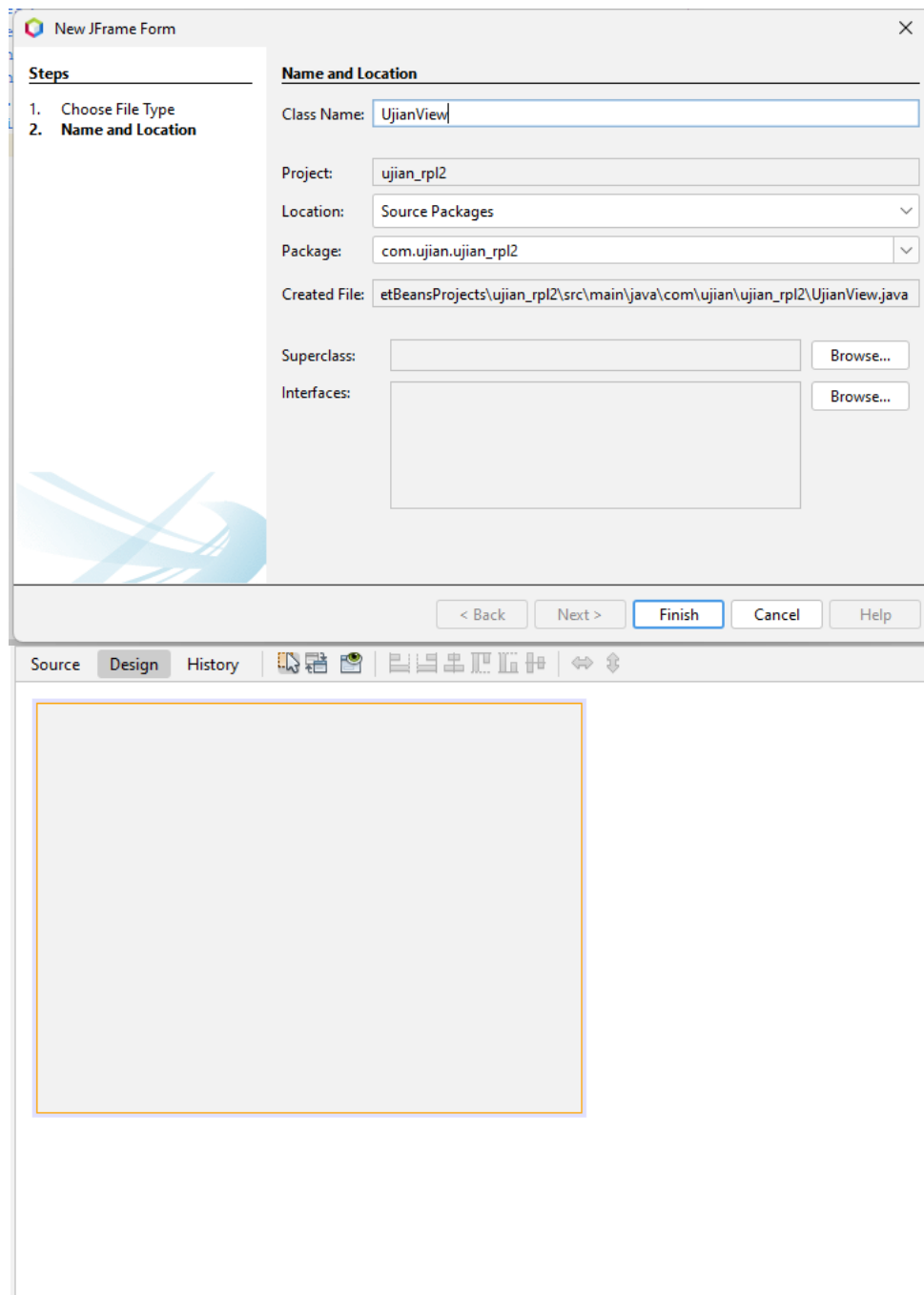
Selanjutnya yaitu membuat class ModelUjian dan ModelTabelUjian pada folder Model





Untuk membuat class pada folder controller, repository dan service sama seperti membuat class model. Selanjutnya yaitu membuat frameform pada folder view. Untuk membuat frameform kita klik kanan pilih new kemudian pilih JFrame Form lalu klik kemudian kita beri nama UjianView. Kemudian akan menampilkan halaman design untuk tampilan form yang nantinya akan kita edit dan sesuaikan.





Tahap selanjutnya yaitu kita mulai membuat logika code program untuk “Aplikasi manajemen buku Perpustakaan”. Tahap pertama yaitu kita mulai dari class ModelUjian.

- **ModelUjian**

```

Source History
1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this lic
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package com.ujian.ujian_rpl.model;
6   import jakarta.persistence.*;
7
8   /**
9    *
10   * @author 2060
11   */
12   @Entity
13   @Table (name = "Ujian")
14   public class ModelUjian {
15       @Id
16       @GeneratedValue(strategy = GenerationType.IDENTITY)
17       @Column (name = "id")
18       private int id;
19
20       @Column (name = "kd_buku", nullable = false, length = 8)
21       private String kd_buku;
22
23       @Column (name = "judul", nullable = false, length = 50)
24       private String judul;
25
26       @Column (name = "penulis")
27       private String penulis;
28
29       @Column (name = "tahun_terbit")
30       private int tahun_terbit;
31
32       public ModelUjian(int id, String kd_buku, String judul, String penulis, int tahun_terbit) {
33           this.id = id;
34           this.kd_buku = kd_buku;
35           this.judul = judul;
36           this.penulis = penulis;
37           this.tahun_terbit = tahun_terbit;
38       }
39
40       public ModelUjian() {
41       }
42
43       public int getId() {
44           return id;
45       }
46
47       public void setId(int id) {
48           this.id = id;
49       }
50
51       public String getkd_buku() {
52           return kd_buku;
53       }
54
55       public void setkd_buku(String kd_buku) {
56           this.kd_buku = kd_buku;
57       }
58
59       public String getjudul() {
60           return judul;
61       }
62
63       public void setjudul(String judul) {
64           this.judul = judul;
65       }
66
67       public String getpenulis() {
68           return penulis;
69       }
70
71       public void setpenulis(String penulis) {
72           this.penulis = penulis;
73       }
74
75       public int gettahun_terbit() {
76           return tahun_terbit;
77       }
78
79       public int settahun_terbit(int tahun_terbit) {
80           this.tahun_terbit = tahun_terbit;
81           return 0;
82       }
83
84   }
85

```

Kode di atas adalah kelas Java bernama ModelUjian yang digunakan untuk mewakili data di tabel database bernama Ujian. Dengan menggunakan anotasi @Entity, kelas ini dikenali sebagai entitas database oleh Hibernate. Properti id adalah kunci utama yang nilainya diatur secara otomatis oleh database, sedangkan properti lainnya seperti kd_buku, judul, penulis, dan tahun_terbit digunakan untuk menyimpan data buku. Setiap properti ini dihubungkan dengan kolom di tabel menggunakan anotasi seperti @Column, yang juga menentukan aturan tambahan seperti panjang maksimal dan apakah nilai boleh kosong atau tidak. Kelas ini memiliki dua jenis konstruktor: konstruktor lengkap untuk mengatur semua data saat membuat objek baru, dan konstruktor kosong yang diperlukan Hibernate untuk memproses data. Untuk setiap properti, ada metode getter dan setter yang digunakan untuk membaca dan mengubah nilai.

- **ModelTabelUjian**

```
Source History
4  /*
5  package com.ujian.ujian_rpl.model;
6
7  import java.util.List;
8  import javax.swing.table.AbstractTableModel;
9
10 /**
11  *
12  * @author 2060
13  */
14 public class ModelTabelUjian extends AbstractTableModel{
15
16     private List<ModelUjian> ujianList;
17     private String[] columnNames = {"ID", "Kode Buku", "Judul", "Penulis", "Tahun Terbit"};
18     public ModelTabelUjian(List<ModelUjian> ujianList) {
19         this.ujianList = ujianList;
20     }
21
22     @Override
23     public int getRowCount() {
24         return ujianList.size();
25     }
26
27     @Override
28     public int getColumnCount() {
29         return columnNames.length;
30     }
31
32     @Override
33     public Object getValueAt(int rowIndex, int columnIndex) {
34         ModelUjian ujian = ujianList.get(rowIndex);
35         return switch (columnIndex) {
36             case 0 -> ujian.getId();
37             case 1 -> ujian.getkd_buku();
38             case 2 -> ujian.getjudul();
39             case 3 -> ujian.getpenulis();
40             case 4 -> ujian.gettahun_terbit();
41             default -> null;
42         };
43     }
44
45     @Override
46     public String getColumnName(int column) {
47         return columnNames[column]; //Mengatur Nama Kolom
48     }
49
50     @Override
51     public boolean isCellEditable(int rowindex, int columnindex) {
52         return false; //Semua sel tidak dapat diedit
53     }
54
55     public void setUjianList(List<ModelUjian> ujianList) {
56         this.ujianList = ujianList;
57         fireTableDataChanged(); //Memberitahu JTable bahwa data telah berubah
58     }
59
60 }
```

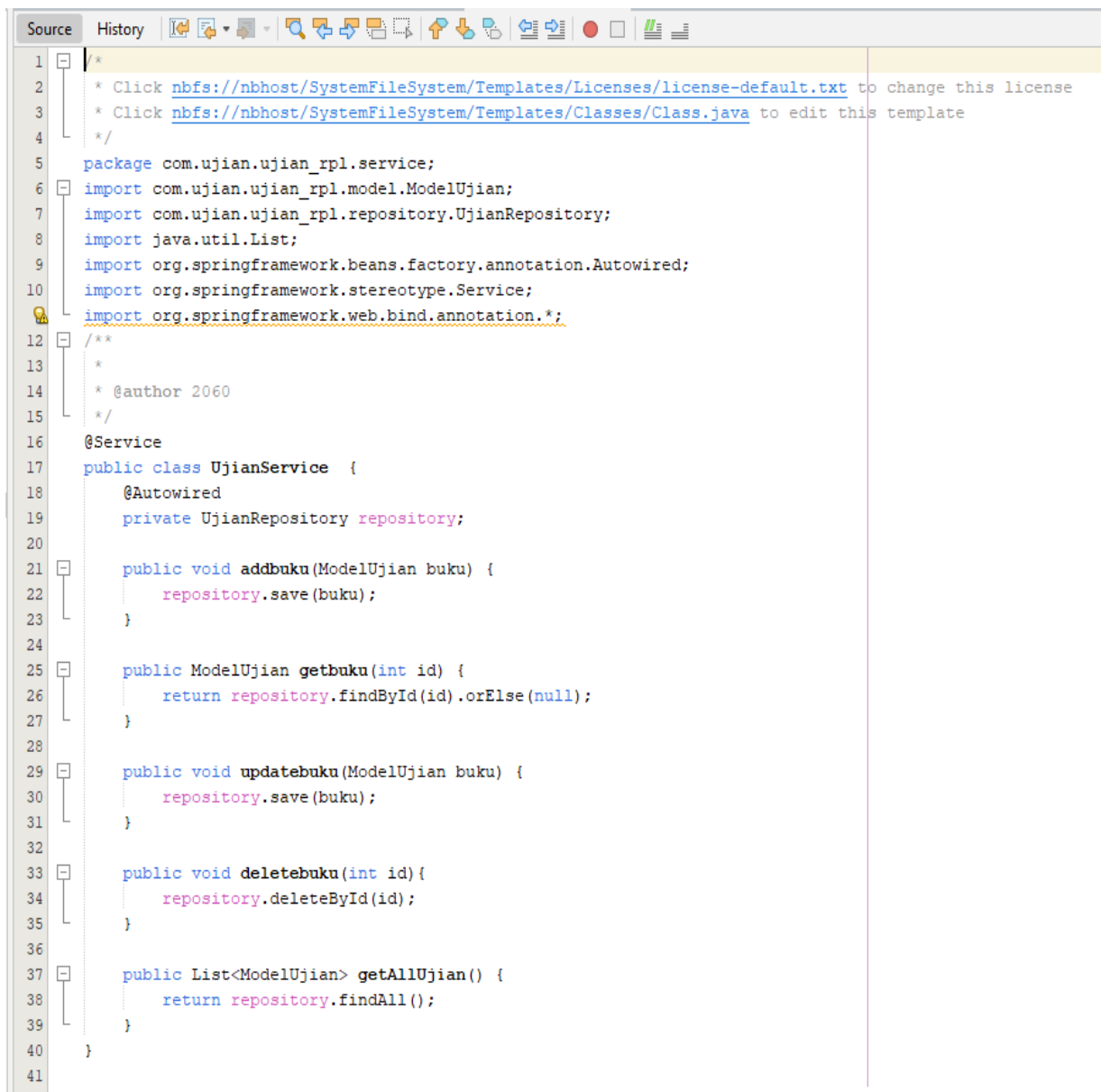
Kode diatas adalah kelas yang digunakan untuk menampilkan data dalam tabel menggunakan JTable pada aplikasi Java Swing. Data yang ditampilkan berasal dari daftar objek ModelUjian, yang berisi informasi seperti ID, kode buku, judul, penulis, dan tahun terbit. Nama kolom tabel ditentukan dalam array columnNames, dan jumlah baris serta kolom diatur menggunakan metode getRowCount dan getColumnCount. Metode getValueAt bertugas mengambil data dari objek ModelUjian berdasarkan baris dan kolom tertentu, sehingga setiap sel tabel dapat menampilkan data yang sesuai. Nama kolom diatur oleh metode getColumnName, sementara metode isCellEditable memastikan semua sel tabel tidak bisa diedit langsung. Selain itu, kelas ini memiliki metode setUjianList untuk mengganti data yang ditampilkan di tabel dan secara otomatis memperbarui tampilan tabel menggunakan fireTableDataChanged

- **UjianController**

```
4  */
5  package com.ujian.ujian_rpl.controller;
6
7  import org.springframework.web.bind.annotation.*;
8  import com.ujian.ujian_rpl.model.ModelUjian;
9  import com.ujian.ujian_rpl.service.UjianService;
10
11 import java.util.List;
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.stereotype.Controller;
14
15 /**
16  *
17  * @author 2060
18  */
19
20 @Controller
21 public class UjianController {
22     @Autowired
23     private UjianService ujianService;
24
25     //Untuk ADD
26     public String addUjian(@RequestBody ModelUjian buku) {
27         ujianService.addbuku(buku);
28         return "Buku added successfully";
29     }
30
31     //Get by ID
32     public ModelUjian getUjian(@PathVariable int id) {
33         return ujianService.getbuku(id);
34     }
35
36     // Update
37     public String updateUjian(@RequestBody ModelUjian buku) {
38         ujianService.updatebuku(buku);
39         return "Buku updated successfully";
40     }
41
42     // Delete By ID
43     public String deleteUjian(@PathVariable int id) {
44         ujianService.deletebuku(id);
45         return "Buku deleted successfully";
46     }
47
48     // Get All
49     public List<ModelUjian> getAllUjian() {
50         return ujianService.getAllUjian();
51     }
52 }
```

Kelas ini berfungsi sebagai penghubung antara antarmuka pengguna (UI) dan bagian aplikasi yang menangani proses pengelolaan data di dalam Spring Framework. Dengan menggunakan anotasi `@Controller`, kelas ini menangani permintaan HTTP dan meneruskannya ke lapisan layanan untuk memproses data. Anotasi `@Autowired` digunakan untuk secara otomatis menyuntikkan dependensi dari layanan `UjianService`, yang berisi fungsi-fungsi terkait pengelolaan data buku ujian. Metode-metode dalam kelas ini memungkinkan pengelolaan data buku, seperti menambah, memperbarui, menghapus, dan mengambil data buku. Metode `addUjian` digunakan untuk menambahkan buku baru, `getUjian` untuk mengambil data buku berdasarkan ID, `updateUjian` untuk memperbarui data buku yang ada, dan `deleteUjian` untuk menghapus buku berdasarkan ID. Sedangkan metode `getAllUjian` mengembalikan daftar semua buku yang tersedia.

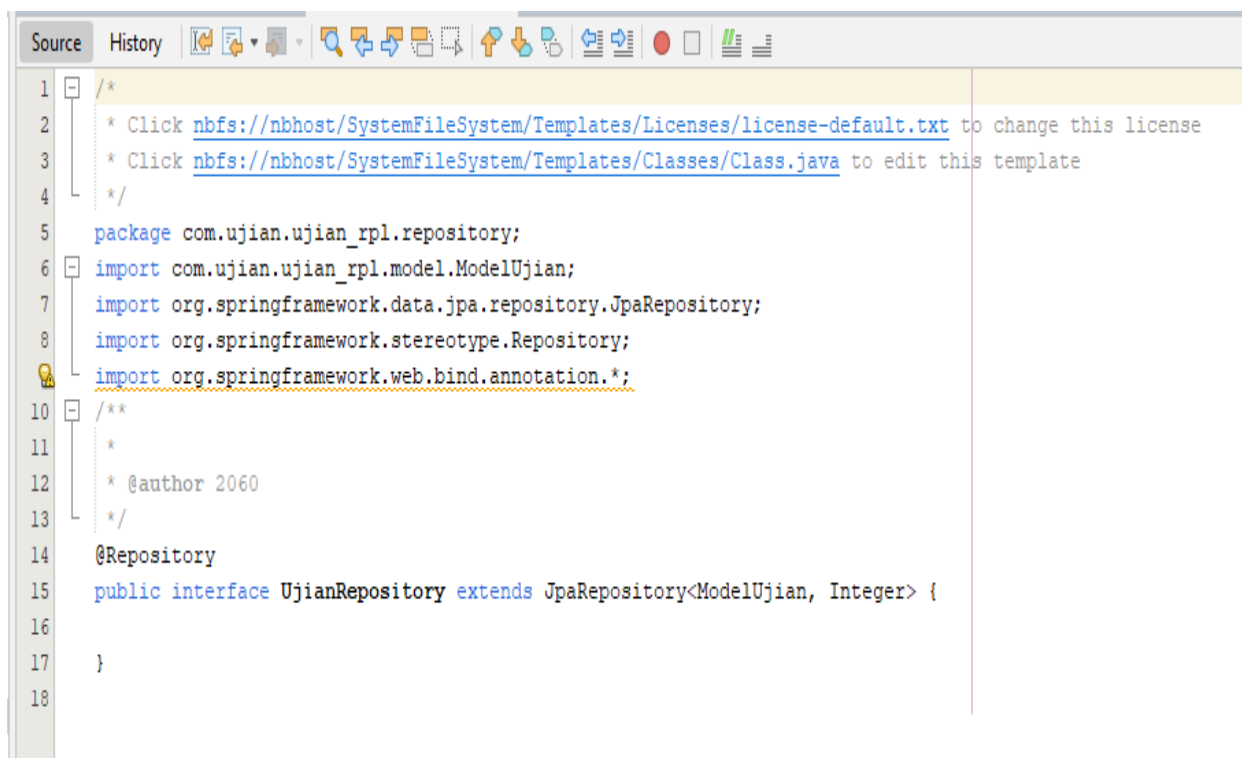
- **UjianService**



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package com.ujian.ujian_rpl.service;
6  import com.ujian.ujian_rpl.model.ModelUjian;
7  import com.ujian.ujian_rpl.repository.UjianRepository;
8  import java.util.List;
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.stereotype.Service;
11 import org.springframework.web.bind.annotation.*;
12 /**
13  *
14  * @author 2060
15  */
16 @Service
17 public class UjianService {
18     @Autowired
19     private UjianRepository repository;
20
21     public void addbuku(ModelUjian buku) {
22         repository.save(buku);
23     }
24
25     public ModelUjian getbuku(int id) {
26         return repository.findById(id).orElse(null);
27     }
28
29     public void updatebuku(ModelUjian buku) {
30         repository.save(buku);
31     }
32
33     public void deletebuku(int id) {
34         repository.deleteById(id);
35     }
36
37     public List<ModelUjian> getAllUjian() {
38         return repository.findAll();
39     }
40 }
41
```


Kelas `UjianService` berfungsi untuk mengelola operasi terkait data buku ujian, seperti menambah, memperbarui, menghapus, dan mengambil data dari database. Diberi anotasi `@Service`, kelas ini berperan sebagai layanan dalam aplikasi Spring, dengan `@Autowired` digunakan untuk menyuntikkan dependensi dari `UjianRepository`, yang mengatur interaksi dengan database. Metode-metode dalam kelas ini mencakup `addbuku` untuk menambahkan buku baru, `getbuku` untuk mengambil data buku berdasarkan ID, `updatebuku` untuk memperbarui data buku, dan `deletebuku` untuk menghapus buku berdasarkan ID. Selain itu, metode `getAllUjian` mengembalikan daftar semua buku ujian yang ada di database, memudahkan pengelolaan data buku ujian menggunakan repositori Spring Data JPA.

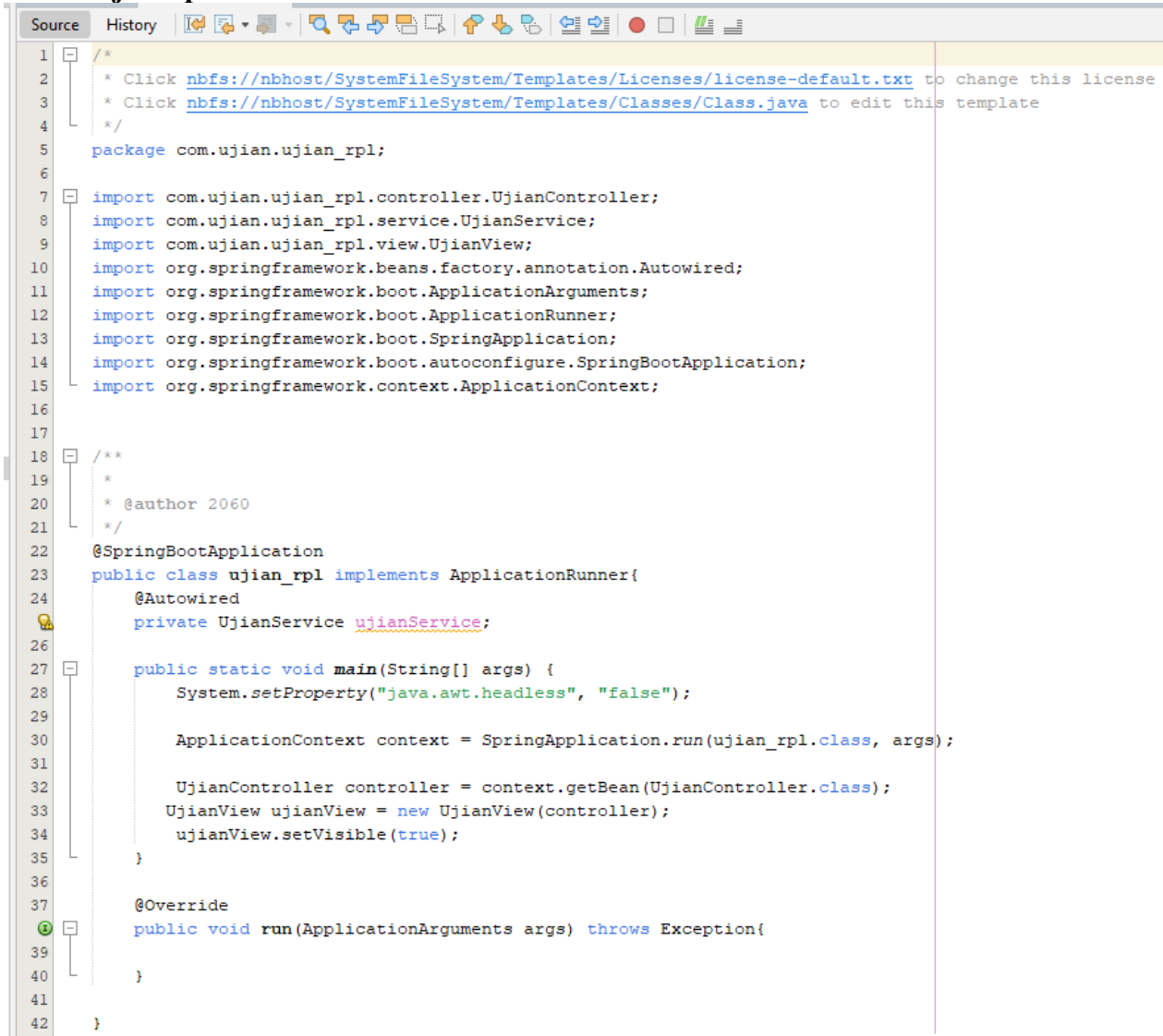
- **UjianRepository**



```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package com.ujian.ujian_rpl.repository;
6   import com.ujian.ujian_rpl.model.ModelUjian;
7   import org.springframework.data.jpa.repository.JpaRepository;
8   import org.springframework.stereotype.Repository;
9   import org.springframework.web.bind.annotation.*;
10  /**
11   *
12   * @author 2060
13   */
14  @Repository
15  public interface UjianRepository extends JpaRepository<ModelUjian, Integer> {
16
17  }
18
```

`UjianRepository` adalah sebuah interface yang digunakan untuk mengelola data buku ujian dalam database. Karena meng-extend `JpaRepository`, interface ini secara otomatis mendapatkan berbagai metode untuk melakukan operasi dasar seperti menambah, mengambil, memperbarui, dan menghapus data tanpa perlu menulis kode secara manual. Misalnya, metode `save` untuk menyimpan data, `findById` untuk mencari data berdasarkan ID, dan `deleteById` untuk menghapus data berdasarkan ID. Dengan anotasi `@Repository`, Spring tahu bahwa interface ini berfungsi untuk berinteraksi dengan database, sehingga memudahkan pengelolaan data buku ujian dalam aplikasi.

- **UjianRpl**

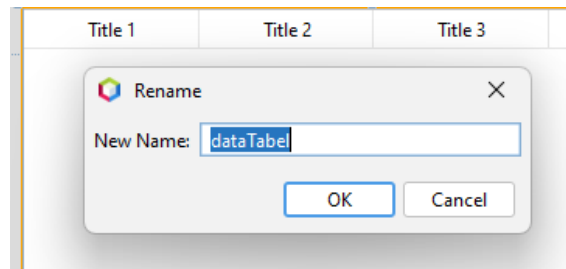
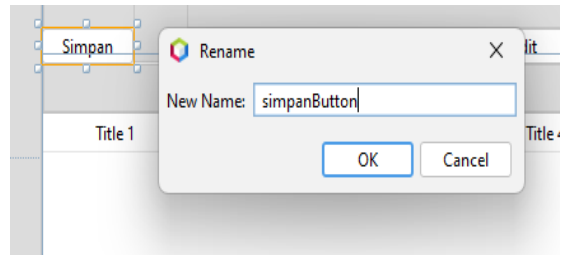
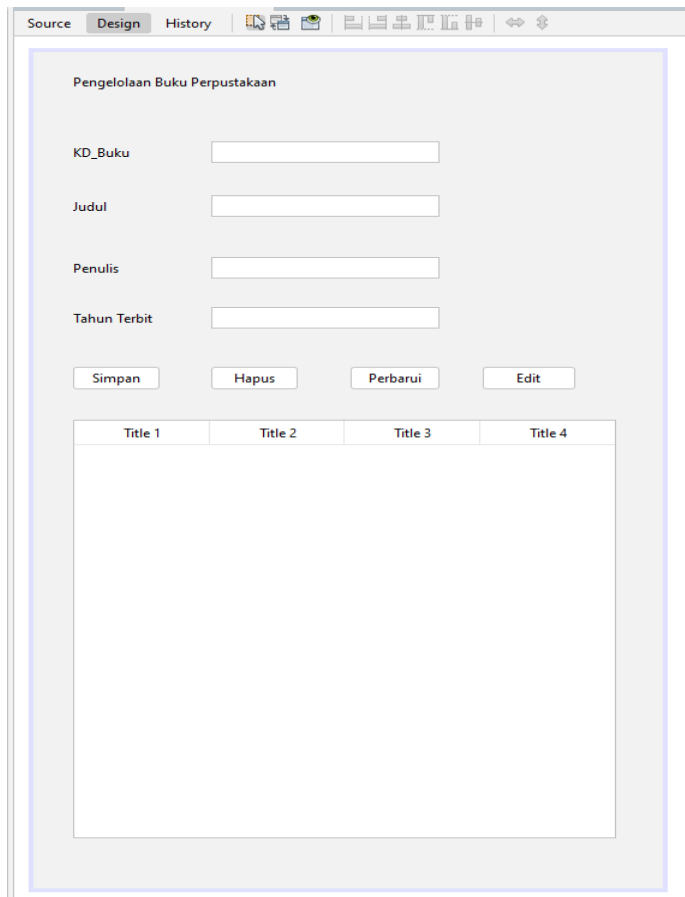


```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package com.ujian.ujian_rpl;
6
7  import com.ujian.ujian_rpl.controller.UjianController;
8  import com.ujian.ujian_rpl.service.UjianService;
9  import com.ujian.ujian_rpl.view.UjianView;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.boot.ApplicationArguments;
12 import org.springframework.boot.ApplicationRunner;
13 import org.springframework.boot.SpringApplication;
14 import org.springframework.boot.autoconfigure.SpringBootApplication;
15 import org.springframework.context.ApplicationContext;
16
17
18 /**
19 *
20 * @author 2060
21 */
22 @SpringBootApplication
23 public class ujian_rpl implements ApplicationRunner{
24     @Autowired
25     private UjianService ujianService;
26
27     public static void main(String[] args) {
28         System.setProperty("java.awt.headless", "false");
29
30         ApplicationContext context = SpringApplication.run(ujian_rpl.class, args);
31
32         UjianController controller = context.getBean(UjianController.class);
33         UjianView ujianView = new UjianView(controller);
34         ujianView.setVisible(true);
35     }
36
37     @Override
38     public void run(ApplicationArguments args) throws Exception{
39
40     }
41
42 }
```

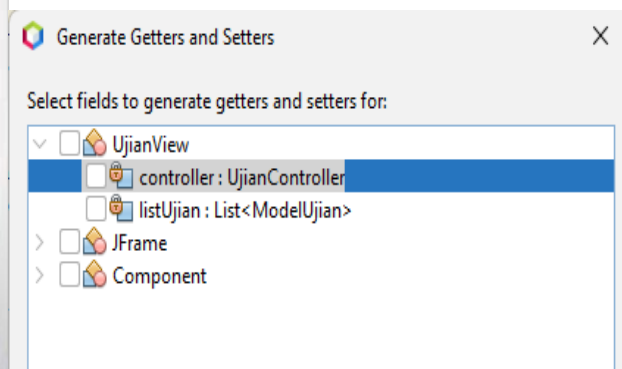
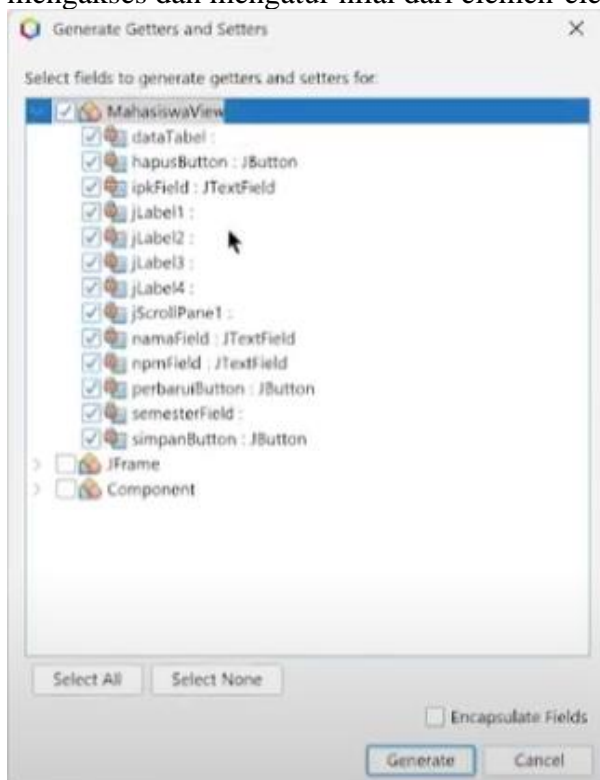
Kelas `ujian_rpl` adalah kelas utama yang menjalankan aplikasi Spring Boot. Aplikasi ini dimulai dari metode `main`, yang memulai aplikasi Spring dan mengatur semua komponen yang diperlukan, seperti `controller` dan tampilan (`view`). Di sini, `UjianController` digunakan untuk menangani logika aplikasi, sementara `UjianView` menampilkan antarmuka pengguna. Setelah aplikasi dimulai, tampilan pengguna akan muncul dengan perintah `ujianView.setVisible(true)`. Kelas ini juga mengimplementasikan `ApplicationRunner`, yang memungkinkan eksekusi kode setelah aplikasi berjalan.

- **UjianView**

Pada `UjianView`, langkah pertama adalah merancang tampilan antarmuka pengguna (UI) dengan elemen seperti tombol, tabel, dan input teks. Setelah itu, setiap elemen diberi nama variabel yang jelas, seperti `tableUjian` untuk tabel. Kemudian setelah selesai design ui kita berpindah ke bagian source untuk mengatur fungsi dan logika aplikasi.



Sebelum mulai membuat fungsi, langkah pertama yang perlu dilakukan adalah membuat getter dan setter untuk setiap elemen UI yang telah dirancang. Getter dan setter ini berfungsi untuk mengakses dan mengatur nilai dari elemen-elemen tersebut melalui kode program



langkah berikutnya adalah membuat fungsi. Fungsi ini digunakan untuk mengatur logika dan aksi pada setiap elemen UI, seperti menangani klik tombol, mengambil data dari tabel, atau memperbarui tampilan. Fungsi-fungsi ini memastikan setiap elemen UI dapat berfungsi sesuai dengan kebutuhan aplikasi.

```

1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to ch
3   * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
4   */
5   package com.ujian.ujian_rpl.view;
6
7   import com.ujian.ujian_rpl.controller.UjianController;
8   import com.ujian.ujian_rpl.model.ModelTabelUjian;
9   import com.ujian.ujian_rpl.model.ModelUjian;
10  import java.util.List;
11  import javax.swing.JButton;
12  import javax.swing.JLabel;
13  import javax.swing.JOptionPane;
14  import javax.swing.JPanel;
15  import javax.swing.JScrollPane;
16  import javax.swing.JTable;
17  import javax.swing.JTextField;
18  import javax.swing.table.TableModel;
19
20  /**
21   *
22   * @author 2060
23   */
24  public class UjianView extends javax.swing.JFrame {
25
26      /**
27       * Creates new form UjianView
28       */
29      private UjianController controller;
30      private List<ModelUjian> listUjian;
31      public UjianView(UjianController controller) {
32          this.controller = controller;
33          initComponents();
34          loadUjianTable();
35      }
36
37      public void loadUjianTable(){
38          List<ModelUjian> listUjian = controller.getAllUjian();
39          ModelTabelUjian ModelTabelUjiantableModel = new ModelTabelUjian(listUjian);
40          dataTabel.setModel(ModelTabelUjiantableModel);
41      }
42
43      private UjianView() {
44          throw new UnsupportedOperationException("Not supported yet");
45      }

```

```

54 @SuppressWarnings("unchecked")
55 Generated Code
210
211 private void simpanButtonActionPerformed(java.awt.event.ActionEvent evt) {
212
213     String kd_buku = kd_bukuField.getText(); // Access directly
214     String judul = judulField.getText(); // Access directly
215     String penulis = penulisField.getText(); // Access directly
216     int tahun_terbit = Integer.parseInt(tahun_terbitField.getText()); // Access directly
217
218     // Create ModelUjian object with the data
219     ModelUjian ujian = new ModelUjian(0, kd_buku, judul, penulis, tahun_terbit);
220
221     // Print the details
222     System.out.println(ujian.getkd_buku());
223     System.out.println(ujian.getjudul());
224     System.out.println(ujian.getpenulis());
225     System.out.println(ujian.gettahun_terbit());
226
227     // Add the new ujian and refresh the table
228     controller.addUjian(ujian);
229     loadUjianTable();
230 }
231
232 private void hapusButtonActionPerformed(java.awt.event.ActionEvent evt) {
233     // TODO add your handling code here:
234     JTextField idField = new JTextField(10);
235
236     // Membuat panel untuk menampung JTextField
237     JPanel panel = new JPanel();
238     panel.add(new JLabel("Masukkan ID yang ingin dihapus:"));
239     panel.add(idField);
240
241     // Menampilkan dialog box dengan JTextField, tombol OK, dan Cancel
242     int result = JOptionPane.showConfirmDialog(null, panel,
243         "Hapus Mahasiswa", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
244
245     // Jika tombol OK ditekan
246     if (result == JOptionPane.OK_OPTION) {
247         try {
248             // Mengambil input ID dan memanggil metode deleteMhs
249             int id = Integer.parseInt(idField.getText());
250             controller.deleteUjian(id);
251             JOptionPane.showMessageDialog(null, "Data berhasil dihapus.", "Sukses", JOptionPane.INFORMATION_MESSAGE);
252             loadUjianTable();
253         } catch (NumberFormatException e) {
254             // Menangani error jika ID yang dimasukkan bukan angka
255             JOptionPane.showMessageDialog(null, "ID harus berupa angka.", "Error", JOptionPane.ERROR_MESSAGE);
256         }
257     }
258 }
259
260 private void kd_bukuFieldActionPerformed(java.awt.event.ActionEvent evt) {
261     // TODO add your handling code here:
262 }
263
264 private void penulisFieldActionPerformed(java.awt.event.ActionEvent evt) {
265     // TODO add your handling code here:
266 }
267
268 private void tahun_terbitFieldActionPerformed(java.awt.event.ActionEvent evt) {
269     // TODO add your handling code here:
270 }
271
272 private void judulFieldActionPerformed(java.awt.event.ActionEvent evt) {
273     // TODO add your handling code here:
274 }

```

```

261 private void kd_bukuFieldActionPerformed(java.awt.event.ActionEvent evt) {
262     // TODO add your handling code here:
263 }
264
265 private void penulisFieldActionPerformed(java.awt.event.ActionEvent evt) {
266     // TODO add your handling code here:
267 }
268
269 private void tahun_terbitFieldActionPerformed(java.awt.event.ActionEvent evt) {
270     // TODO add your handling code here:
271 }
272
273 private void judulFieldActionPerformed(java.awt.event.ActionEvent evt) {
274     // TODO add your handling code here:
275 }
276
277 private void editButtonActionPerformed(java.awt.event.ActionEvent evt) {
278     int selectedRow = dataTabel.getSelectedRow(); // Mendapatkan baris yang dipilih
279
280     if (selectedRow != -1) { // Pastikan ada baris yang dipilih
281         TableModel model = dataTabel.getModel(); // Mendapatkan model tabel
282
283         // Mengisi field input dengan data dari baris yang dipilih
284         kd_bukuField.setText(model.getValueAt(selectedRow, 1).toString());
285         judulField.setText(model.getValueAt(selectedRow, 2).toString());
286         penulisField.setText(model.getValueAt(selectedRow, 3).toString());
287         tahun_terbitField.setText(model.getValueAt(selectedRow, 4).toString());
288
289         // Memperbarui data jika tombol perbarui diklik
290         perbaruiButton.addActionListener(new java.awt.event.ActionListener() {
291             public void actionPerformed(java.awt.event.ActionEvent evt) {
292                 String kd_buku = kd_bukuField.getText();
293                 String judul = judulField.getText();
294                 String penulis = penulisField.getText();
295                 int tahun_terbit = Integer.parseInt(tahun_terbitField.getText());
296
297                 // Memperbarui model ujian dengan data baru
298                 ModelUjian ujian = controller.getAllUjian().get(selectedRow);
299                 ujian.setkd_buku(kd_buku);
300                 ujian.setjudul(judul);
301                 ujian.setpenulis(penulis);
302                 ujian.settahun_terbit(tahun_terbit);
303
304                 // Update data di model dan tabel
305                 controller.updateUjian(ujian);
306                 loadUjianTable();
307
308                 JOptionPane.showMessageDialog(null, "Data berhasil diperbarui!");
309             }
310         });
311     } else {
312         JOptionPane.showMessageDialog(null, "Pilih baris data yang ingin diedit!", "Peringatan", JOptionPane.WARNING_MESSAGE);
313     }
314 }
315
316
317 /**
318  * @param args the command line arguments
319  */
320 public static void main(String args[]) {
321     /* Set the Nimbus look and feel */
322     /* Look and feel setting code (optional) */
323
324     /* Create and display the form */
325     java.awt.EventQueue.invokeLater(new Runnable() {
326         public void run() {
327             new UjianView().setVisible(true);
328         }
329     });
330 }
331
332 // Variables declaration - do not modify
333 private javax.swing.JTable dataTabel;
334 private javax.swing.JButton editButton;
335 private javax.swing.JButton hapusButton;
336 private javax.swing.JLabel jLabel1;
337 private javax.swing.JLabel jLabel2;
338 private javax.swing.JLabel jLabel3;
339 private javax.swing.JLabel jLabel4;
340 private javax.swing.JLabel jLabel5;
341 private javax.swing.JScrollPane jScrollPane1;
342 private javax.swing.JTextField judulField;
343 private javax.swing.JTextField kd_bukuField;
344 private javax.swing.JTextField penulisField;
345 private javax.swing.JButton perbaruiButton;
346 private javax.swing.JButton simpanButton;
347 private javax.swing.JTextField tahun_terbitField;
348 // End of variables declaration

```

```

369
370 public void setDataTable(JTable dataTable) {
371     this.dataTable = dataTable;
372 }
373
374 public void setHapusButton(JButton hapusButton) {
375     this.hapusButton = hapusButton;
376 }
377
378 public void setjLabel1(JLabel jLabel1) {
379     this.jLabel1 = jLabel1;
380 }
381
382 public void setjLabel2(JLabel jLabel2) {
383     this.jLabel2 = jLabel2;
384 }
385
386 public void setjLabel3(JLabel jLabel3) {
387     this.jLabel3 = jLabel3;
388 }
389
390 public void setjLabel4(JLabel jLabel4) {
391     this.jLabel4 = jLabel4;
392 }
393
394 public void setjLabel5(JLabel jLabel5) {
395     this.jLabel5 = jLabel5;
396 }
397
398 public void setjScrollPane1(JScrollPane jScrollPane1) {
399     this.jScrollPane1 = jScrollPane1;
400 }
401
402 public void setJudulField(JTextField judulField) {
403     this.judulField = judulField;
404 }
405
406 public void setKd_bukuField(JTextField kd_bukuField) {
407     this.kd_bukuField = kd_bukuField;
408 }
409
410 public void setPenulisField(JTextField penulisField) {
411     this.penulisField = penulisField;
412 }
413
414 public void setPerbaruiButton(JButton perbaruiButton) {
415     this.perbaruiButton = perbaruiButton;
416 }
417
418 public void setSimpanButton(JButton simpanButton) {
419     this.simpanButton = simpanButton;
420 }
421
422 public void setTahun_terbitField(JTextField tahun_terbitField) {
423     this.tahun_terbitField = tahun_terbitField;
424 }
425
426 private Object getkd_bukuField() {
427     throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/C
428 }
429
430 public JButton getEditButton() {
431     return editButton;
432 }
433
434 public void setEditButton(JButton editButton) {
435     this.editButton = editButton;
436 }
437
438 }

```

Kelas UjianView ini merupakan bagian dari tampilan antarmuka pengguna (GUI) pada aplikasi pengelolaan data buku perpustakaan yang menggunakan Java Swing. Program ini memanfaatkan JFrame sebagai jendela utama aplikasi, serta berbagai komponen GUI seperti JLabel, JTextField, JButton, JTable, dan JScrollPane untuk membangun tampilan yang interaktif dan mudah digunakan. Kelas utama dalam aplikasi ini adalah UjianView, yang bertugas untuk menampilkan informasi buku kepada pengguna. Kelas ini terhubung langsung dengan logika aplikasi melalui objek UjianController, yang berfungsi untuk mengelola dan memproses data buku sesuai dengan perintah yang diberikan oleh pengguna melalui tampilan antarmuka.

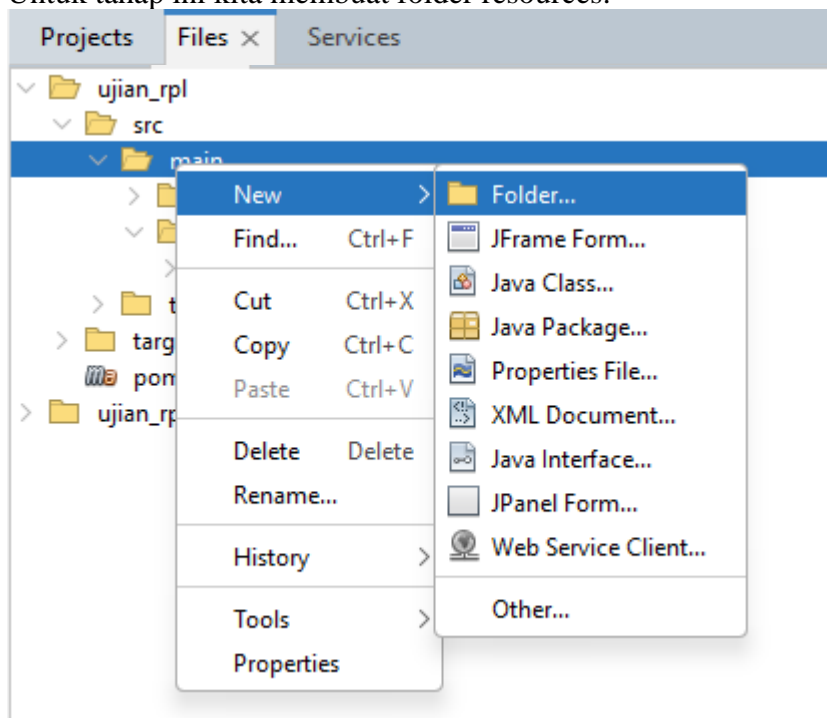
Pada form aplikasi ini, terdapat beberapa label yang menggambarkan kolom input untuk data buku, seperti "KD_Buku", "Judul", "Penulis", dan "Tahun Terbit". Setiap label disertai dengan komponen input berupa JTextField, yang memungkinkan pengguna untuk memasukkan informasi terkait buku yang akan ditambahkan atau diedit. Selain itu, terdapat beberapa tombol

aksi seperti "Simpan", "Hapus", dan "Perbarui", yang memungkinkan pengguna untuk melakukan operasi terkait pengelolaan data buku dalam sistem. Tombol-tombol ini akan memicu berbagai aksi di backend aplikasi yang berhubungan dengan penambahan, penghapusan, dan pembaruan data buku. Tabel JTable digunakan untuk menampilkan daftar buku yang telah ada dalam database atau yang sudah dikelola oleh aplikasi, memberikan gambaran langsung mengenai data buku yang ada.

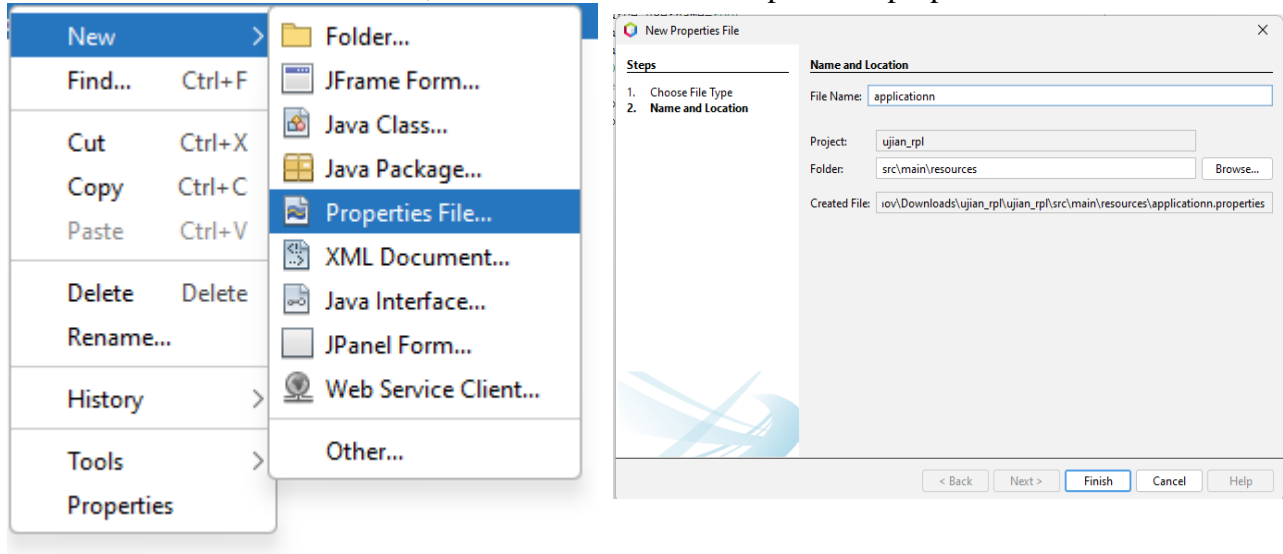
Fungsi dari tombol-tombol tersebut didefinisikan dalam metode event-handler tertentu, seperti `simpanButtonActionPerformed`, yang akan menambahkan buku baru ke dalam sistem berdasarkan informasi yang dimasukkan oleh pengguna. Ketika pengguna ingin menghapus buku, mereka dapat memilih tombol "Hapus" dan memasukkan ID buku yang ingin dihapus dari database. Selain itu, pengguna juga dapat mengedit data buku yang ada dengan memilih baris tabel yang sesuai dan mengklik tombol "Edit". Setelah itu, informasi buku yang dipilih akan dimuat ke dalam kolom input, dan pengguna dapat memperbarui informasi tersebut sebelum menekan tombol "Perbarui" untuk menyimpan perubahan. Semua perubahan yang dilakukan, baik penambahan, penghapusan, atau pembaruan, akan langsung memperbarui tampilan tabel dengan data terbaru.

- **Application.properties**

Untuk tahap ini kita membuat folder resources.



Setelah membuat folder resources, kemudian membuat file application.properties

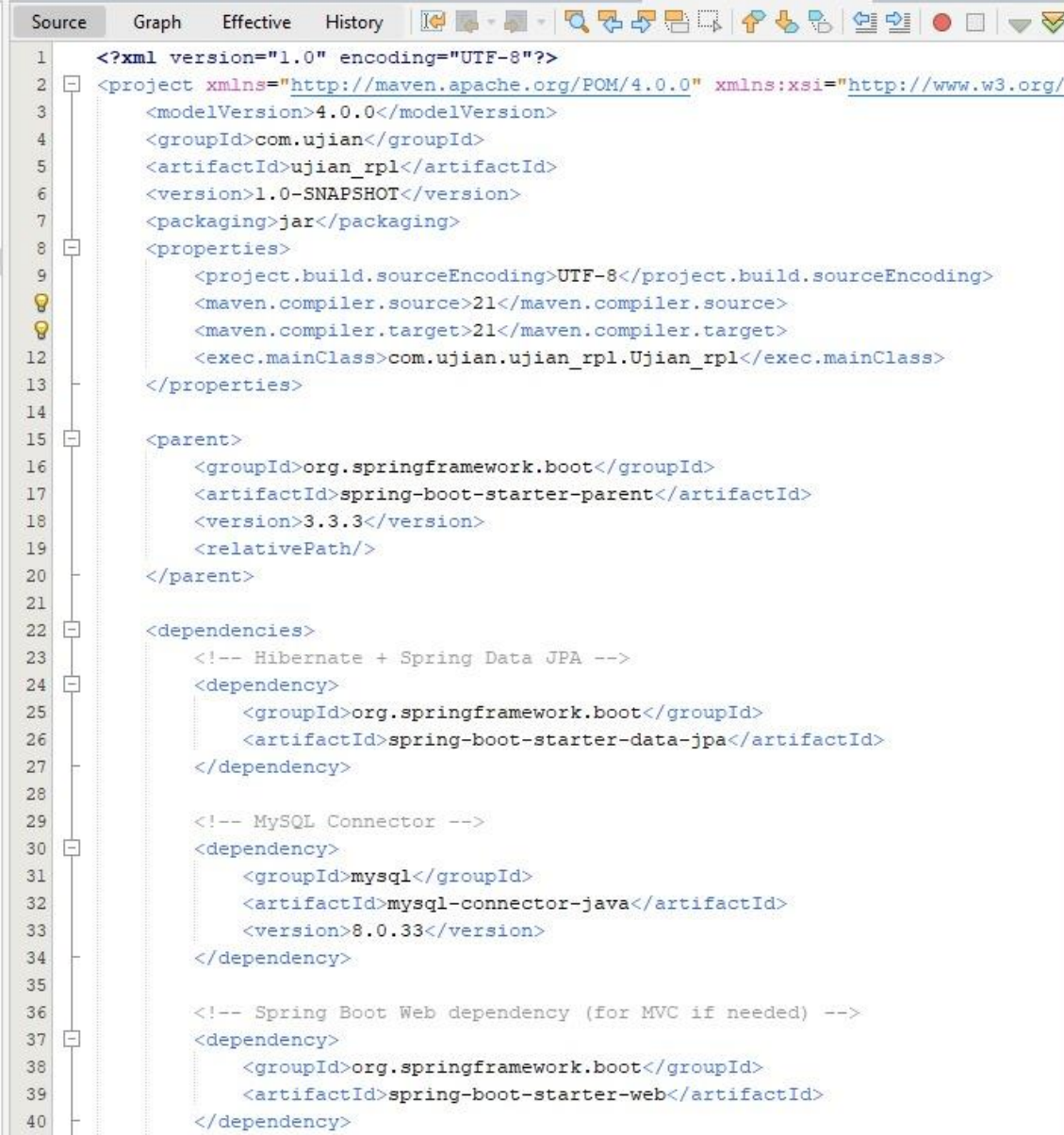


```
Source History
1
2 # Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3 # Click nbfs://nbhost/SystemFileSystem/Templates/Other/properties.properties to edit this template
4
5 # Konfigurasi MySQL Hibernate
6 # Konfigurasi MySQL Hibernate
7 spring.datasource.url=jdbc:mysql://localhost:3306/perpustakaan_ujian_rpl2?useSSL=false&serverTimezone=UTC
8 spring.datasource.username=root
9 spring.datasource.password=
10 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
11 server.port=8083
12 # Hibernate settings
13 spring.jpa.hibernate.ddl-auto=update
14 spring.jpa.show-sql=true
15
16
17
```

Kelas ini berfungsi untuk mengonfigurasi aplikasi Spring Boot agar terhubung dengan database MySQL menggunakan Hibernate ORM. Baris pertama menyertakan URL koneksi database, yang menunjukkan bahwa aplikasi akan terhubung ke database perpustakaan_ujian_rpl2 di localhost pada port 3306, dengan pengaturan tambahan untuk tidak menggunakan SSL dan menyesuaikan zona waktu server. Username yang digunakan adalah root dan password kosong. Selanjutnya, terdapat pengaturan untuk driver JDBC yang digunakan untuk koneksi ke MySQL. Konfigurasi server.port mengatur aplikasi agar berjalan pada port 8083. Pengaturan Hibernate spring.jpa.hibernate.ddl-auto=update memastikan schema database diperbarui secara otomatis sesuai dengan entitas yang ada, sementara spring.jpa.show-sql=true memungkinkan SQL yang dieksekusi ditampilkan di konsol untuk tujuan debugging.

- **Pom.xml**

File ini merupakan pengaturan Maven untuk proyek ini, yang menggunakan Spring Boot versi 3.3.3 sebagai kerangka kerja utama dan dirancang untuk berjalan di Java versi 21. Terdapat beberapa dependensi penting seperti Spring Data JPA untuk pengelolaan database, MySQL Connector untuk menghubungkan ke MySQL, dan Spring Boot Web untuk membuat aplikasi berbasis web. Selain itu, terdapat dependensi untuk pengujian aplikasi.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>com.ujian</groupId>
5   <artifactId>ujian_rpl</artifactId>
6   <version>1.0-SNAPSHOT</version>
7   <packaging>jar</packaging>
8   <properties>
9     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
10    <maven.compiler.source>21</maven.compiler.source>
11    <maven.compiler.target>21</maven.compiler.target>
12    <exec.mainClass>com.ujian.ujian_rpl.Ujian_rpl</exec.mainClass>
13  </properties>
14
15  <parent>
16    <groupId>org.springframework.boot</groupId>
17    <artifactId>spring-boot-starter-parent</artifactId>
18    <version>3.3.3</version>
19    <relativePath/>
20  </parent>
21
22  <dependencies>
23    <!-- Hibernate + Spring Data JPA -->
24    <dependency>
25      <groupId>org.springframework.boot</groupId>
26      <artifactId>spring-boot-starter-data-jpa</artifactId>
27    </dependency>
28
29    <!-- MySQL Connector -->
30    <dependency>
31      <groupId>mysql</groupId>
32      <artifactId>mysql-connector-java</artifactId>
33      <version>8.0.33</version>
34    </dependency>
35
36    <!-- Spring Boot Web dependency (for MVC if needed) -->
37    <dependency>
38      <groupId>org.springframework.boot</groupId>
39      <artifactId>spring-boot-starter-web</artifactId>
40    </dependency>
```

```

41
42     <!-- Testing dependencies -->
43     <dependency>
44         <groupId>org.springframework.boot</groupId>
45         <artifactId>spring-boot-starter-test</artifactId>
46         <scope>test</scope>
47     </dependency>
48 </dependencies>
49
50 <build>
51     <plugins>
52         <plugin>
53             <groupId>org.springframework.boot</groupId>
54             <artifactId>spring-boot-maven-plugin</artifactId>
55         </plugin>
56     </plugins>
57 </build>
58
59
60
61
62
63 </project>

```

- **Output**

Pengelolaan Buku Perpustakaan

KD_Buku

Judul

Penulis

Tahun Terbit

ID	Kode Buku	Judul	Penulis	Tahun Terbit

➤ **Simpan**

Pengelolaan Buku Perpustakaan

KD_Buku

Judul

Penulis

Tahun Terbit

ID	Kode Buku	Judul	Penulis	Tahun Terbit
8	AK212	Meraih Mimpi G..	Rangskuy	2045

➤ **Edit**

Pengelolaan Buku Perpustakaan

KD_Buku


Judul

Penulis

Tahun Terbit

ID	Kode Buku	Judul	Penulis	Tahun Terbit
8	AK212	Sang pemimpi	Dewa	2044

Message X

 **Data berhasil diperbarui!**

➤ **Hapus**

Pengelolaan Buku Perpustakaan

KD_Buku

Judul

Penulis


Tahun Terbit

ID	Kode Buku	Judul	Penulis	Tahun Terbit
8	AK212	Sang pemimpi	Dewa	2044

Hapus Mahasiswa ×

Masukkan ID yang ingin dihapus:

Sukses ×

 **Data berhasil dihapus.**

ID	Kode Buku	Judul	Penulis	Tahun Terbit

BAB IV

PENUTUP

4.1 Kesimpulan

Aplikasi manajemen buku perpustakaan berbasis digital yang telah dibuat memberikan kemudahan dalam pencatatan dan pengelolaan informasi buku, seperti judul, pengarang, penerbit, dan tahun terbit. Dengan menggunakan teknologi terkini, aplikasi ini berhasil mengurangi ketergantungan pada metode pencatatan manual yang sering menimbulkan kesalahan atau kehilangan data. Pengelolaan informasi yang lebih efisien memungkinkan pengelola perpustakaan untuk mengakses, memodifikasi, dan mencari data dengan lebih cepat, serta menyediakan laporan yang akurat untuk evaluasi pengelolaan koleksi buku.

Aplikasi ini juga meningkatkan produktivitas dan efisiensi operasional perpustakaan, memberikan solusi praktis bagi pengelola untuk menjaga dan memelihara koleksi buku mereka. Dengan antarmuka yang sederhana dan mudah digunakan, aplikasi ini dapat dioperasikan oleh pengguna dari berbagai latar belakang tanpa kesulitan. Secara keseluruhan, aplikasi manajemen buku ini tidak hanya mempermudah pengelolaan data perpustakaan, tetapi juga memastikan kualitas layanan yang lebih baik dan lebih responsif untuk pengguna perpustakaan.

4.2 Saran

Saran untuk pengembangan lebih lanjut aplikasi manajemen buku perpustakaan ini adalah untuk menambahkan fitur pencarian yang lebih canggih dan filter berdasarkan kategori, seperti genre buku atau tahun terbit, guna mempermudah pengguna dalam menemukan buku yang mereka butuhkan. Selain itu, integrasi dengan sistem peminjaman dan pengembalian buku secara otomatis dapat meningkatkan fungsionalitas aplikasi, menjadikannya lebih komprehensif dalam mendukung kegiatan operasional perpustakaan.