BAB IV

IMPLEMENTASI SISTEM

4.1 Implementasi Basis Data

Software pengolahan *database* yang digunakan dalam implementasi *database* yaitu *MySQL* dengan bahasa pemrograman java. Berikut merupakan tabel – tabel yang dibangun menjadi *database* pada sistem ini adalah :

1. Tabel admin

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	user_id	int(11)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
2	nama	varchar(255)	latin1_swedish_ci		Ya	*NULL*			Ubah Hapus Lainnya
3	pass	varchar(255)	latin1_swedish_ci		Ya	*NULL*			Ubah Hapus Lainnya

Gambar 4.1 Tabel admin

2. Tabel data uji

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	data_id	int(11)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
2	KET	varchar(255)	latin1_swedish_ci		Ya	*NULL*			Ubah Hapus Lainnya
3	data_x	double			Ya	*NULL*			Ubah Hapus Lainnya
4	data_x2	double			Ya	*NULL*			Ubah Hapus Lainnya
5	data_y	double			Ya	*NULL*			Ubah Hapus Lainnya

Gambar 4.2 Tabel data uji

3. Table data hasil prediksi

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	History_ID	int(11)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
2	History_Hasil	double			Ya	*NULL*			Ubah Hapus Lainnya
3	History_Ket	longtext	latin1_swedish_ci		Ya	*NULL*			Ubah Hapus Lainnya
4	History_Tanggal	datetime			Ya	*NULL*			Ubah Hapus Lainnya

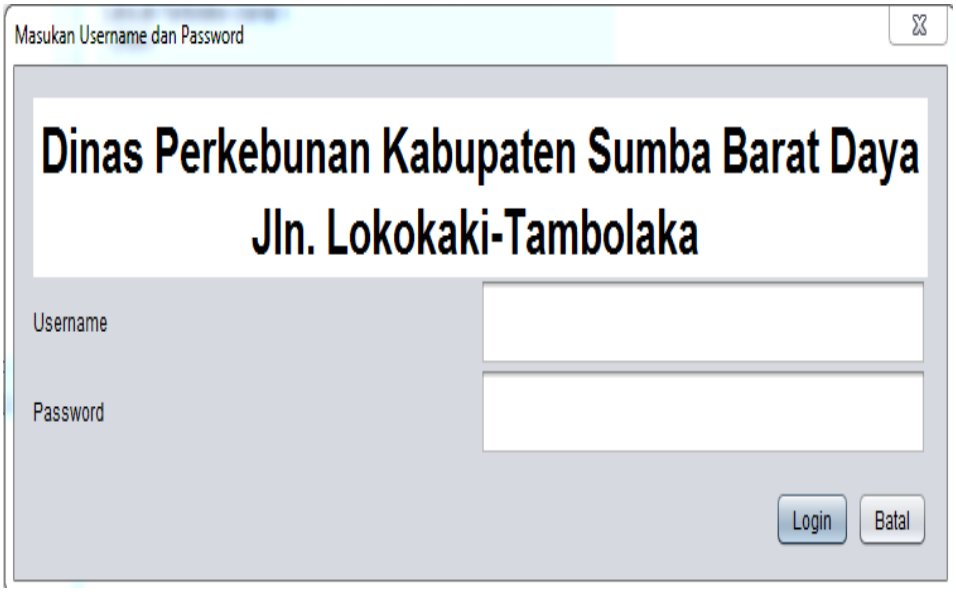
Gambar 4.3 Tabel hasil prediksi

4.2 Implementasi Sistem

Aplikasi prediksi hasil produksi tanaman vanili ini dibangun menggunakan bahasa pemrograman java dan dihubungkan dengan *database MySQL*. Tampilan dari sistem yang telah dibangun adalah sebagai berikut :

1. Tampilan *form login*

   Form login digunakan admin untuk melakukan proses *login* dengan memasukan *Username* dan *Password* dengan benar kemudian menekam tombol *login*.



Gambar 4.4 Tampilan *form login*

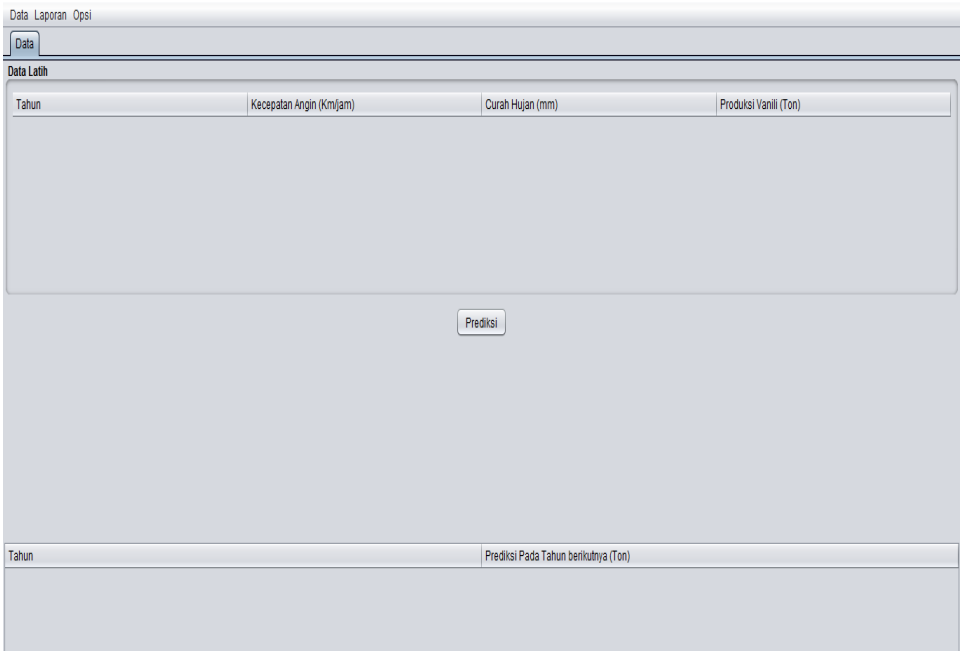
Tampilan ini muncul ketika pertama aplikasi di jalankan. Berikut adalah *source code* untuk membuat proses *login.*

```
public class loginPanel extends javax.swing.JPanel {

public loginPanel() {

Entity

Table(name = "user", catalog = "sederhana", schema = "")

NamedQueries({ NamedQuery(name = "User.findAll", query = "SELECT
u FROM User u"), NamedQuery(name = "User.findByUserId", query =
"SELECT u FROM User u WHERE u.userId = :userId"),
NamedQuery(name  = "User.findByNama", query = "SELECT u FROM
User u WHERE u.nama = :nama") NamedQuery(name =
"User.findByPass", query = "SELECT u FROM User u WHERE u.pass =
:pass")})
```

source code ini digunakan admin untuk melakukan proses *login* dengan memasukan *username* dan *password* dengan benar kemudian menekan tombol *login*.

2. Tampilan awal

Tampilan awal aplikasi ketika diakses, pada *form* ini admin dapat memilih menu data untuk memasukan data uji sebagai acuan data prediksi.



Gambar 4.5 Tampilan awal aplikasi

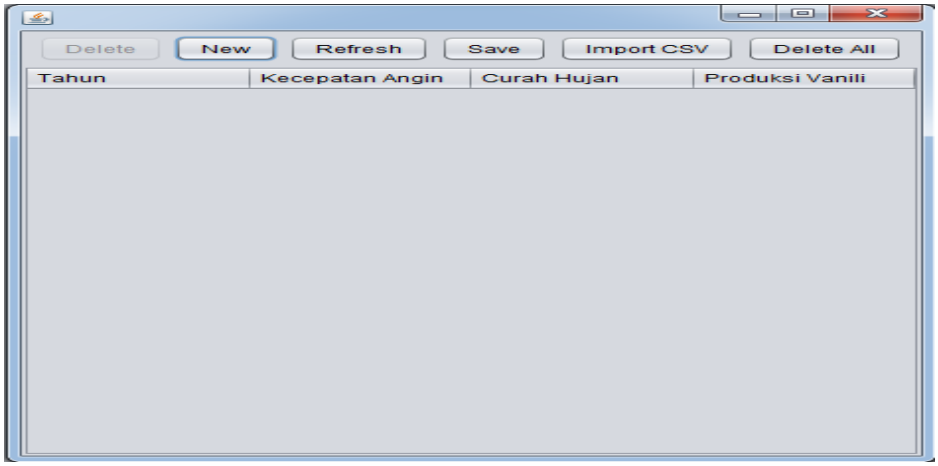
Tampilan ini merupakan tampilan awal dari aplikasi ketika *user* sebagai admin sukses melakukan *login*

```
public class MainFrame extends javax.swing.JFrame {
public MainFrame() {
        initComponents();
        initSetup();
        initGraphAsli();
 private void jButton1 Action Performed (java.awt.event.ActionEvent     evt) {jButton3ActionPerformed(evt);
private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) {      jDialog2.setLocationRelativeTo(null);
        this.jDialog2.setVisible(rootPaneCheckingEnabled);
private void jMenuItem8ActionPerformed(java.awt.event.ActionEvent evt) {
        jDialog3.setLocationRelativeTo(null);
        this.jDialog3.setVisible(rootPaneCheckingEnabled);
```

Source code ini merupakan fungsi untuk menampilkan menu utama pada aplikasi secara umum setelah admin melakukan proses *login* pada saat ketika aplikasi di *running*.

3. Tampilan *input* data uji

Setelah admin memilih menu data maka sistem akan menampilkan *form* untuk memasukan data uji.



4.6 Tampilan *input* data uji

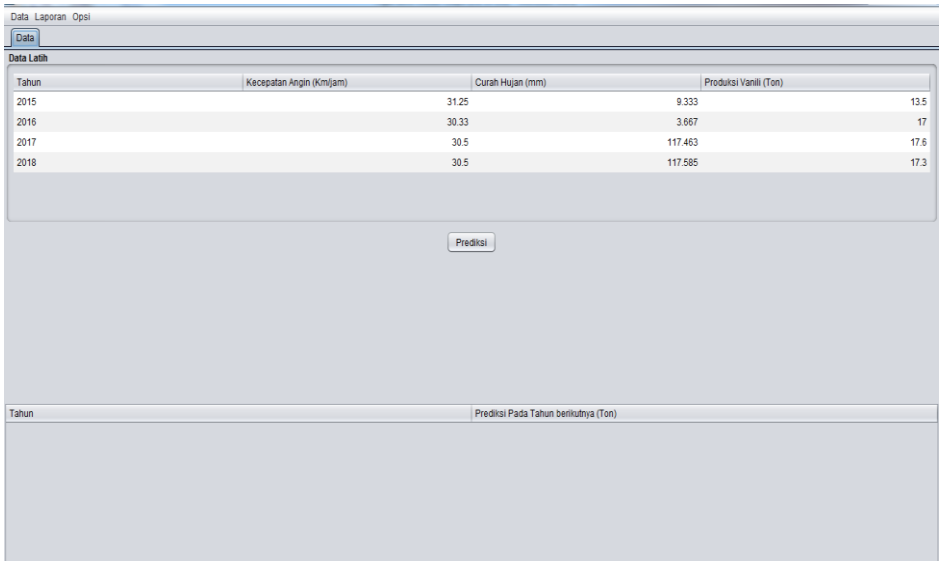
Tampilan *form* ini berfungsi bagi admin untuk melakukan pengolahan data prediksi seperti data kecepatan angin, curah hujan dan produksi vanili. Berikut adalah *source code* untuk *form* input data uji

```
private static File getFile(){
      JFileChooser fc = new JFileChooser();
      File file = null;
      fc.addChoosableFileFilter(new FileNameExtensionFilter("Data
CSV Files", "csv"));
      int returnVal = fc.showOpenDialog(null);
      if (returnVal == JFileChooser.APPROVE_OPTION) {
         file = fc.getSelectedFile();
      }
      return file;
```

Source code diatas merupakan sebuah perintah untuk memanggil data yang di uji yaitu data kecepatan angin, data curah hujan dan data hasil produksi tanaman vanili yang sudah yang sudah di siapkan oleh admin dalam bentuk file csv.

4. Tampilan hasil *input* data uji

Setelah admin meng-*input* data uji, maka admin harus melakukan *update* data.



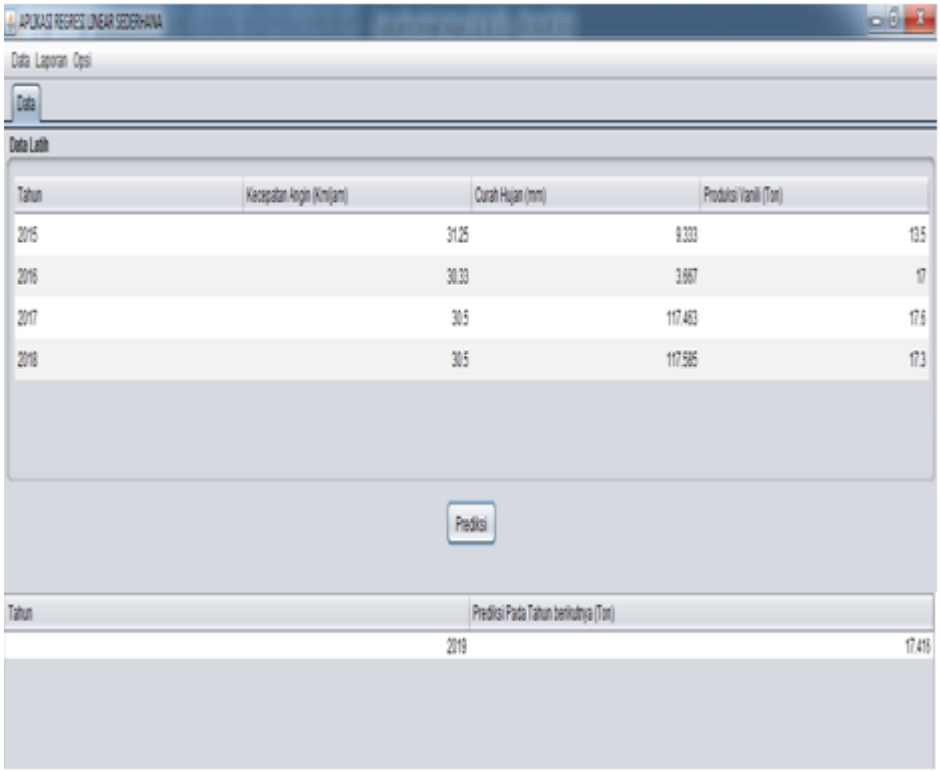
Gambar 4.7 Tampilan lihat data uji

```
 Entity
Table(name = "data", catalog = "sederhana", schema = "")
NamedQueries({
 NamedQuery(name = "Data.findAll", query = "SELECT d FROM Data
d")
 NamedQuery(name = "Data.findByDataId", query = "SELECT d FROM
Data d WHERE d.dataId = :dataId")
 NamedQuery(name = "Data.findByDataX", query = "SELECT d FROM
Data d WHERE d.dataX = :dataX")
 NamedQuery(name = "Data.findByDataY", query = "SELECT d FROM
Data d WHERE d.dataY = :dataY")})
 public class Data implements Serializable   @CsvBindByName(column =
"Ket")
    private String Ket;
```

Source code ini merupakan sebuah perintah untuk memanggil data uji yang telah di simpan pada database yang sudah di input oleh admin seperti data kecepatan angin, data curah hujan dan data hasil produksi tanaman vanili untuk melakukan prediksi.

5. Tampilan hasil prediksi

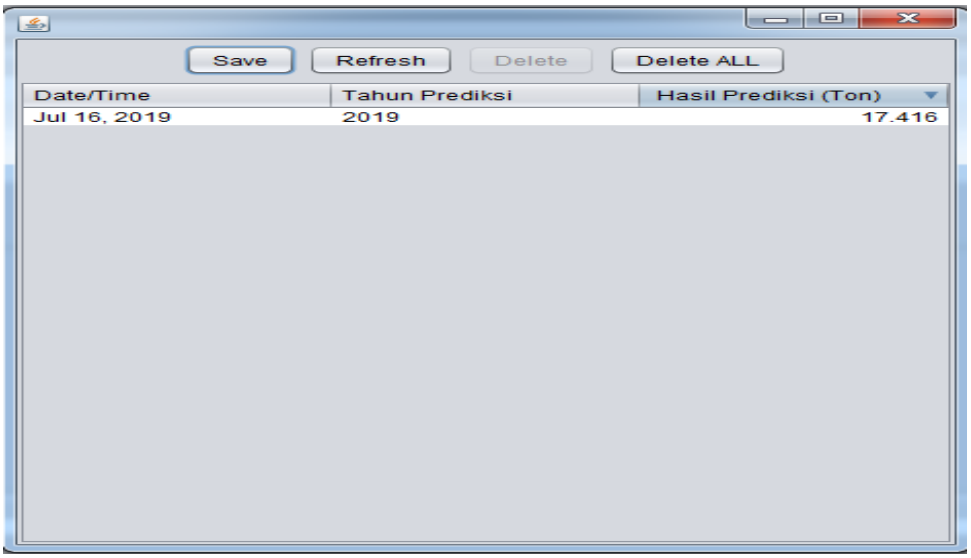
Jika admin ingin melakukan prediksi maka dapat mengklik pada pada tombol prediksi setelah *input* data uji dan aplikasi akan menampilkan hasil prediksi dengan menggunakan metode regresi linear berganda.



Gambar 4.8 Tampilan hasil prediksi

6. Tampilan data hasil prediksi

Berikut tampilan *form* data hasil prediksi.



4.9 Tampilan data hasil prediksi

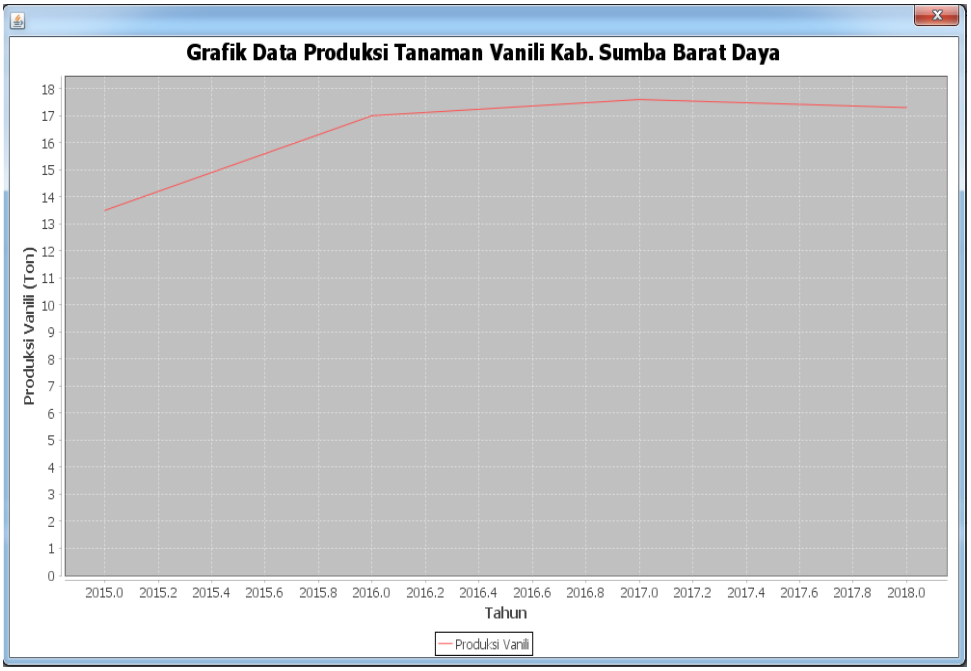
Berikut merupakan tampilan *form* hasil prediksi setelah admin melakukan prediksi.

```
public class HistoryForm extends JPanel {
    public HistoryForm() {
      initComponents();
      if (!Beans.isDesignTime()) {
         entityManager.getTransaction().begin();  }
  }private void deleteButtonActionPerformed(java.awt.event.ActionEvent
evt) {
      int[] selected = masterTable.getSelectedRows();
      List<linearsimple.model.History> toRemove = new
ArrayList<linearsimple.model.History>(selected.length);
      for (int idx = 0; idx < selected.length; idx++) {
         linearsimple.model.History h =
list.get(masterTable.convertRowIndexToModel(selected[idx]));
```

Source code diatas merupakan sebuah perintah pada aplikasi untuk menyimpan data hasil prediksi setelah melakukan proses prediksi dan mneyimpan pada database.

7. Tampilan laporan grafik data produksi

Grafik ini merupakan hasil dari *input* data produki vanili



Gambar 4.10 Grafik data produksi vanili

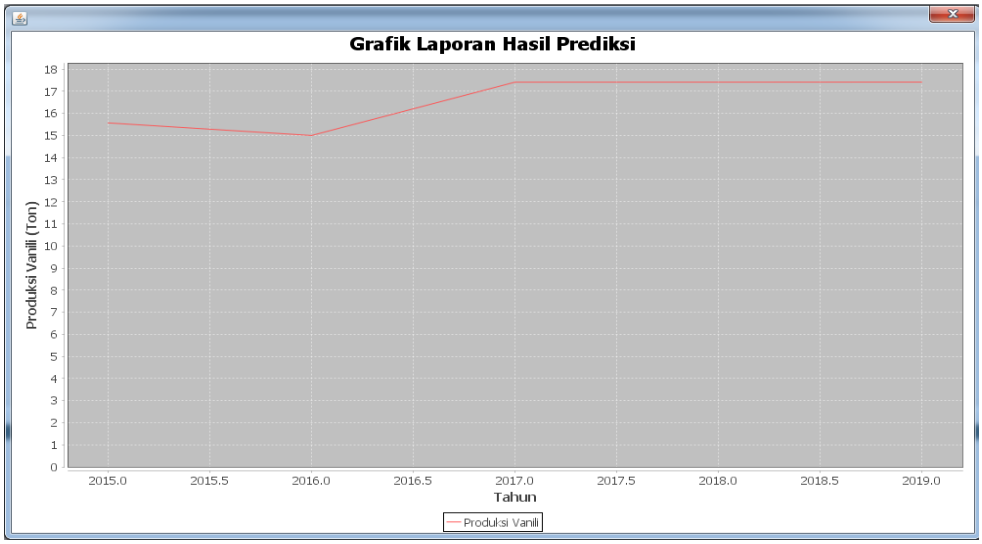
Grafik ini merupakan grafik saat admin melakukan *input* data produksi vanili. Berikut merupakan *Source code* grafik produksi vanili.

```
private XYDataset createDatasetUji() {
      XYSeriesCollection dataset = new XYSeriesCollection();
      XYSeries series = new XYSeries("Produksi Vanili");
      for (Data data : listDataUji) {
         series.add(data.getDataId(),data.getPrediksiY());
          dataset.addSeries(series);
      //dataset.addSeries(CreateSeriesAsli("Data Asli",listDataUji));
      return dataset;
   }
```

Source code diatas merupakan sebuah perintah pada aplikasi untuk membuat sebuag grafik setelah admin melakukan proses input data uji seperti data kecepata angin, data curah hujan dan data hasil produksi tanaman vanili.

8. Tampilan grafik laporan hasil prediksi

Setelah admin melakukan prediksi, maka grafik laporan hasil prediksi akan ditampilkan adalah sebagai berikut:



Gambar 4.11 Grafik laporan data hasil prediksi

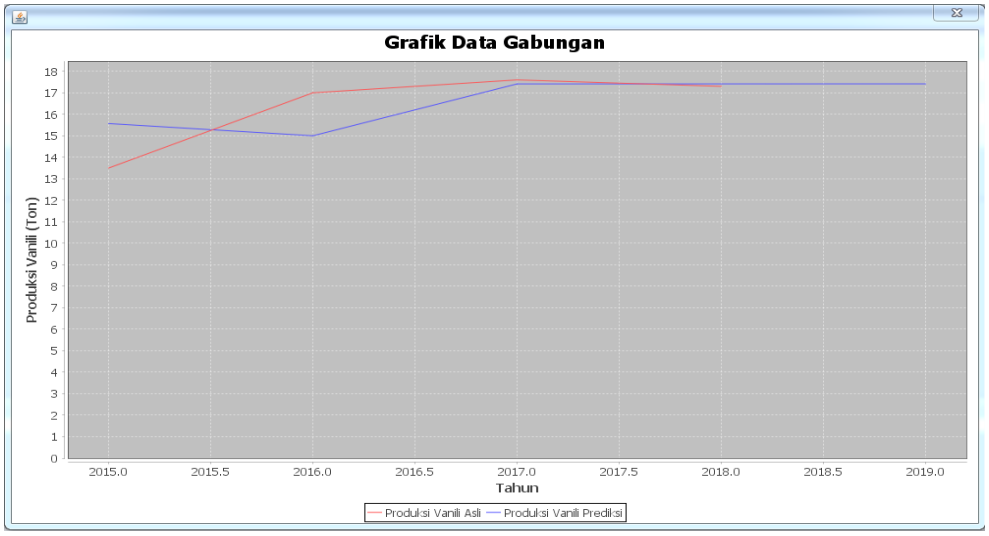
Berikut merupakan source code grafik hasil prediksi

```
private void initGraphUji() {
jPanelUji.removeAll();
jPanelUji.revalidate();
String chartTitle = "Grafik Laporan Hasil Prediksi";
String yAxisLabel = "Produksi Vanili (Ton)";
String xAxisLabel = "Tahun";
XYDataset dataset = createDatasetUji();
JFreeChart chart = ChartFactory.createXYLineChart(chartTitle,
 xAxisLabel, yAxisLabel, dataset);
```

Source code diatas merupakan sebuah perintah pada aplikasi untuk membuat sebuah grafik setelah admin melakukan proses prediksi.

9. Tampilan grafik gabungan

Pada grafik ini, menampilkan hasil grafik gabungan antara data produksi dan data hasil prediksi sebagai berikut:



Gambar 4.12 Tampilan grafik gabungan

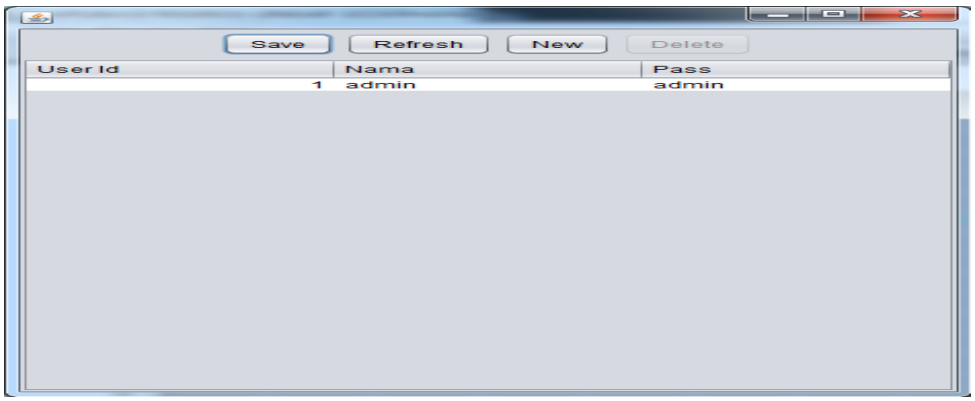
Berikut merupakan *source code* grafik gabungan yaitu grafik produksi dan grafik hasil prediksi.

```
private void initGraphGabungan()
jPanelGabungan.removeAll();
        jPanelGabungan.revalidate();
        String chartTitle = "Grafik Data Gabungan";
        String yAxisLabel = "Produksi Vanili (Ton)";
        String xAxisLabel = "Tahun";
        XYDataset dataset = createDatasetGabungan();
ChartPanel chartPanel = new ChartPanel( chart );
for (Data data : listData) {
            int parseInt = Integer.parseInt(data.getKet());
            series.add(parseInt,data.getDataY());
```

Source code diatas merupakan sebuah perintah pada aplikasi untuk membuat sebuah grafik gabungan antara grafik hasil produksi tanaman vanili dan grafik hasil prediksi.

1. Tampilan *form* data pengguna

Pada *form* data penggguna akan menampilkan jumlah admin yang melakukan *login* dengan entitas masing-masing.



Gambar 4.13 Tampilan *form* data pengguna

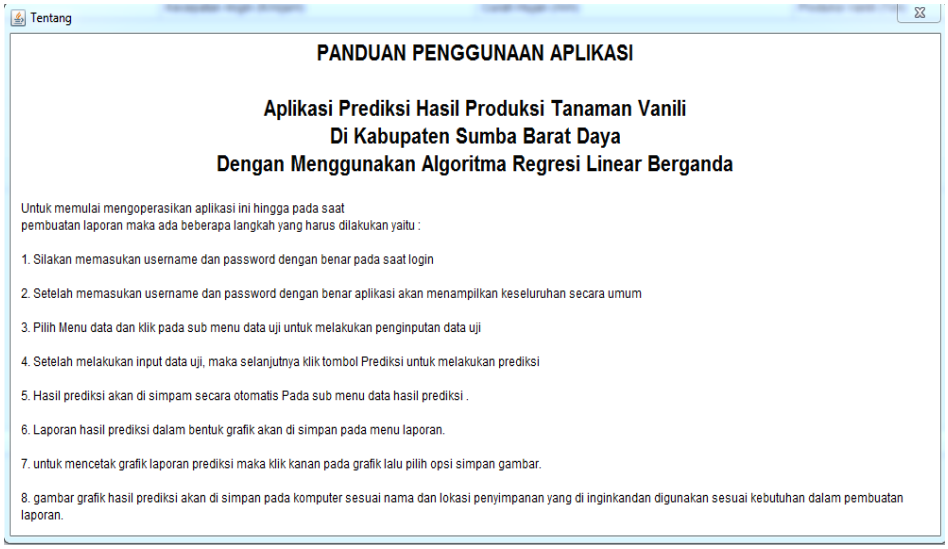
Berikut merupakan source code pada form data pengguna

```
public class UserForm extends JPanel {
        public UserForm() {
        initComponents();
        if (!Beans.isDesignTime()) {
            entityManager.getTransaction().begin();
private void refreshButtonActionPerformed(java.awt.event.ActionEvent
evt) {
          list.clear();
        list.addAll(data);
    private void
deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
        int[] selected = masterTable.getSelectedRows();
        List<linearsimple.model.User> toRemove = new
ArrayList<linearsimple.model.User>(selected.length);
        for (int idx = 0; idx < selected.length; idx++) {
            linearsimple.model.User u =
list.get(masterTable.convertRowIndexToModel(selected[idx]));
            toRemove.add(u);
            entityManager.remove(u);
```

Source code diatas merupakan sebuah perintah pada form data user, pada form data user admin dapat menambahkan data pengguna.

2. Tampilan tentang aplikasi

Pada *form* ini admin dapat menampilkan tentang tujuan dari aplikasi ini dibuat.



Gambar 4.14 Tampilan tentang aplikasi