

- Java Aplikazioko Kodea

- DB_Konexioa.java
- Main.java
- AdministrariLangilea.java
- Akatsa.java
- Bezeroa.java
- Biltegia.java
- BiltegiLangilea.java
- Eramangarria.java
- Eskaera.java
- EskaeraLerroa.java
- FakturaPDF.java
- Fitxaketa.java
- Herria.java
- Hornitzaila.java
- Konponketa.java
- Langilea.java
- LangileSaila.java
- MahaiGainekoa.java
- Mugikorra.java
- Pantaila.java
- Pertsona.java
- Produktua.java
- ProduktuKategoria.java
- SalmentaLangilea.java
- Sarrera.java
- SarreraLerroa.java
- Softwarea.java
- Tableta.java
- TeknikariLangilea.java
- Zerbitzaria.java
- BezeroaDialog.java
- Combolitem.java
- EskaeraDialog.java
- FitxaketakKudeatu.java
- KonponketaXehetasunaElkarritzeta.java
- MenuAdministrazioa.java

- [MenuLogistika.java](#)
- [MenuSalmentak.java](#)
- [MenuTeknikoa.java](#)
- [MenuZuzendaritza.java](#)
- [NireDatuakDialog.java](#)
- [SaioaHastekoPanela.java](#)
- [SarreraBerriaDialog.java](#)
- [TaulaModelatzzailea.java](#)

Java Aplikazioko Kodea

DB_Konexioa.java

```
package db;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 * Datu-baserako konexioa kudeatzen duen klasea.
 * Singleton eredu erabiltzen du konexio bakarra bermatzeko.
 */
public class DB_Konexioa {

    // Konfigurazio datuak
    private static final String URLEA =
    "jdbc:mysql://192.168.115.155:3306/birtek_db";
    private static final String ERABILTZAILEA = "admin";
    private static final String PASAHITZA = "1234";

    /**
     * Connection instantzia estatikoa (Singleton).
     */
    private static Connection instance = null;

    /**
     * Datu-basera konektatzeko metodoa (Singleton Eredua).
     *
     * @return Connection objektua edo null errore bat egonez gero.
     */
    public static Connection konektatu() {
        try {
            if (instance == null || instance.isClosed()) {
                // Driver-a kargatu (aukerakoa JDBC 4.0+ bertsioetan, baina
                gomendagarria)
                Class.forName("com.mysql.cj.jdbc.Driver");
            }
        } catch (SQLException e) {
            System.out.println("Error connecting to the database: " + e.getMessage());
        }
    }
}
```

```

        instance = DriverManager.getConnection(URLEA, ERABILTZAILEA,
PASAHTZA);
    }
} catch (ClassNotFoundException e) {
    System.err.println("MySQL Driver-a ez da aurkitu: " + e.getMessage());
} catch (SQLException e) {
    System.err.println("Errorea datu-basera konektatzean: " +
e.getMessage());
}
return instance;
}
}

```

Main.java

```

package main;

import ui.SaioaHastekoPanela;
import java.awt.EventQueue;

/**
 * Main klasea.
 * Aplikazioaren sarrera puntuoa.
 */
public class Main {
    /**
     * Aplikazioa abiarazten duen metodo nagusia.
     *
     * @param args Komando lerroko argumentuak (ez dira erabiltzen).
     */
    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    SaioaHastekoPanela frame = new SaioaHastekoPanela();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
}

```

AdministrariLangilea.java

```

package model;

import db.DB_Konexioa;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.io.File;
import java.io.FileOutputStream;
import java.awt.Desktop;

/***
 * AdministrariLangilea klasea.
 * Langilea klasaren azpiklasea da, eta administrazio-lanak egiteko metodoak
 * ditu.
 * Langileak, sailak, fakturak eta bestelako entitateak kudeatzeko aukera ematen
 * du.
 */
public class AdministrariLangilea extends Langilea {

    /**
     * AdministrariLangileaeraikitzalea.
     * Langilea objektu batetik abiatuta AdministrariLangilea sortzen du.
     *
     * @param l Langilea objektua, oinarritzko datuekin.
     */
    public AdministrariLangilea(Langilea l) {
        super(l.getIdLangilea(), l.getIzena(), l.getAbizena(), l.getNan(),
l.getJaiotzaData(), l.getHerriaId(),
                l.getHelbidea(), l.getPostaKodea(), l.getTelefonoa(),
l.getEmaila(), l.getHizkuntza(),
                l.getPasahitza(), l.getSaltoTxartelaUid(), l.getAltaData(),
l.getEguneratzeData(),
                l.isAktibo(), l.getSailaId(), l.getIban(), l.getKurrikuluma());
    }

    /**
     * Langile berri bat sortzen du datu-basean.
     *
     * @param izena          Langilearen izena.
     * @param abizena        Langilearen abizena.
     * @param nan            Langilearen NAN zenbakia.
     * @param emaila         Langilearen email helbidea.
     * @param pasahitza     Langilearen pasahitza.
     * @param sailaId        Langilea dagokion sailaren IDa.
     * @param helbidea       Langilearen helbidea.
     * @param herriaId       Langilea bizi den herriaren IDa.
     * @param postaKodea     Posta kodea.
     * @param telefonoa      Harremanetarako telefonoa.
     * @param jaiotzaData   Jaiotza data (YYYY-MM-DD formatuan).
     * @param hizkuntza      Hizkuntza lehenetsia.
     * @param saltoTxartelaUid Salto sistemako txartelaren UIDa.
     * @param aktibo          Langilea aktibo dagoen edo ez.
     * @param iban            Bankuko kontuaren IBAN zenbakia.
     * @throws SQLException Datu-basean errorea gertatzen bada.
    */
}

```

```

        */
    public void langileaSortu(String izena, String abizena, String nan, String
email, String pasahitza, int sailaId,
                           String helbidea, int herriaId, String postaKodea, String telefonoa,
String jaiotzaData,
                           String hizkuntza, String saltoTxartelaUid, boolean aktibo, String iban)
throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "INSERT INTO langileak (izena, abizena, nan, emaila,
pasahitza, saila_id, helbidea, herria_id, posta_kodea, telefonoa, jaiotza_data,
hizkuntza, salto_txartela_uid, aktibo, iban)"
                    + "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setString(1, izena);
        pst.setString(2, abizena);
        pst.setString(3, nan);
        pst.setString(4, email);
        pst.setString(5, pasahitza);
        pst.setInt(6, sailaId);
        pst.setString(7, helbidea);
        pst.setInt(8, herriaId);
        pst.setString(9, postaKodea);
        pst.setString(10, telefonoa);
        pst.setString(11, jaiotzaData);
        pst.setString(12, hizkuntza);
        pst.setString(13, saltoTxartelaUid);
        pst.setBoolean(14, aktibo);
        pst.setString(15, iban);
        pst.executeUpdate();
    }
}

/**
 * Langile bat ezabatzen du datu-basetik.
 *
 * @param idLangilea Ezabatu nahi den langilearen IDa.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void langileaEzabatu(int idLangilea) throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "DELETE FROM langileak WHERE id_langilea = ?";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setInt(1, idLangilea);
        pst.executeUpdate();
    }
}

/**
 * Langile baten datuak eguneratzen ditu.
 *
 * @param idLangilea      Aldatu nahi den langilearen IDa.
 * @param izena            Izen berria.
 * @param abizena          Abizen berria.
 * @param nan              NAN berria.
 * @param emaila           Email berria.
 * @param sailaId          Sail berriaren IDa.
 * @param helbidea         Helbide berria.
 */

```

```

* @param herriaId      Herri berriaren IDa.
* @param postaKodea    Posta kode berria.
* @param telefonoa     Telefono berria.
* @param jaiotzaData   Jaiotza data berria.
* @param hizkuntza    Hizkuntza berria.
* @param pasahitzta   Pasahitz berria.
* @param saltoTxartelaUid Salto txartelaren UID berria.
* @param aktibo        Egoera (aktibo/ez-aktibo).
* @param iban          IBAN berria.
* @throws SQLException Datu-basean errorea gertatzen bada.
*/
public void langileaEditatu(int idLangilea, String izena, String abizena,
String nan, String emaila, int sailaId,
String helbidea, int herriaId, String postaKodea, String telefonoa,
String jaiotzaData,
String hizkuntza, String pasahitzta, String saltoTxartelaUid, boolean
aktibo, String iban)
throws SQLException {
try (Connection kon = DB_Konexioa.konektatu()) {
String sql = "UPDATE langileak SET izena = ?, abizena = ?, nan = ?,"
emaila = ?, saila_id = ?, helbidea = ?, herria_id = ?, posta_kodea = ?, telefonoa = ?
, jaiotza_data = ?, hizkuntza = ?, pasahitzta = ?, salto_txartela_uid = ?, aktibo = ?
, iban = ?, eguneratze_data = NOW() WHERE id_langilea = ?";
PreparedStatement pst = kon.prepareStatement(sql);
pst.setString(1, izena);
pst.setString(2, abizena);
pst.setString(3, nan);
pst.setString(4, emaila);
pst.setInt(5, sailaId);
pst.setString(6, helbidea);
pst.setInt(7, herriaId);
pst.setString(8, postaKodea);
pst.setString(9, telefonoa);
pst.setString(10, jaiotzaData);
pst.setString(11, hizkuntza);
pst.setString(12, pasahitzta);
pst.setString(13, saltoTxartelaUid);
pst.setBoolean(14, aktibo);
pst.setString(15, iban);
pst.setInt(16, idLangilea);
pst.executeUpdate();
}
}

/**
* Langile baten datuak lortzen ditu bere IDa erabiliz.
*
* @param idLangilea Bilatu nahi den langilearen IDa.
* @return Langilea objektua datuekin, edo null aurkitzen ez bada.
* @throws SQLException Datu-basean errorea gertatzen bada.
*/
public Langilea langileaIkusi(int idLangilea) throws SQLException {
Langilea langilea = null;
try (Connection kon = DB_Konexioa.konektatu()) {
String sql = "SELECT * FROM langileak WHERE id_langilea = ?";
PreparedStatement pst = kon.prepareStatement(sql);
pst.setInt(1, idLangilea);

```

```

        java.sql.ResultSet rs = pst.executeQuery();
        if (rs.next()) {
            langilea = new Langilea(
                rs.getInt("id_langilea"),
                rs.getString("izena"),
                rs.getString("abizena"),
                rs.getString("nan"),
                rs.getDate("jaiotza_data"),
                rs.getInt("herria_id"),
                rs.getString("helbidea"),
                rs.getString("posta_kodea"),
                rs.getString("telefonoa"),
                rs.getString("emaila"),
                rs.getString("hizkuntza"),
                rs.getString("pasahitza"),
                rs.getString("salto_txartela_uid"),
                rs.getTimestamp("alta_data"),
                rs.getTimestamp("eguneratze_data"),
                rs.getBoolean("aktibo"),
                rs.getInt("saila_id"),
                rs.getString("iban"),
                rs.getBytes("kurrikulum"));
        }
    }
    return langilea;
}

/**
 * Bezero baten fakturaren datuak ikusteko metooda.
 *
 * @param idFaktura Fakturaren IDa.
 * @return BezeroFaktura objektua, datuekin.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
/**
 * Eskaera bat ikusteko metooda (fakturaren datuekin).
 *
 * @param idEskaera Eskaeraren IDa.
 * @return Eskaera objektua (faktura zenbakia eta URL barne).
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public Eskaera eskaeraIkusi(int idEskaera) throws SQLException {
    Eskaera eskaera = null;
    String sql = "SELECT * FROM eskaerak WHERE id_eskaera = ?";
    try (Connection kon = DB_Konexioa.konektatu();
         PreparedStatement pst = kon.prepareStatement(sql)) {
        pst.setInt(1, idEskaera);
        try (java.sql.ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                eskaera = new Eskaera(
                    rs.getInt("id_eskaera"),
                    rs.getInt("bezeroa_id"),
                    (Integer) rs.getObject("langilea_id"),
                    rs.getTimestamp("data"),
                    rs.getTimestamp("eguneratze_data"),
                    rs.getBigDecimal("guztira_prezioa"),
                    rs.getString("faktura_zenbakia"));
            }
        }
    }
}

```

```

                rs.getString("faktura_url"),
                rs.getString("eskaera_egoera")));
            }
        }
    }
    return eskaera;
}

/**
 * Langile sail berri bat sortzen du.
 *
 * @param izena      Sailaren izena.
 * @param kokapena   Sailaren kokapena.
 * @param deskribapena Sailaren deskribapena.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void langileSailaSortu(String izena, String kokapena, String
deskribapena) throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "INSERT INTO langile_sailak (izena, kokapena,
deskribapena) VALUES (?, ?, ?)";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setString(1, izena);
        pst.setString(2, kokapena);
        pst.setString(3, deskribapena);
        pst.executeUpdate();
    }
}

/**
 * Langile sail bat ezabatzen du.
 *
 * @param idSaila Ezabatu nahi den sailaren IDa.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void langileSailaEzabatu(int idSaila) throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "DELETE FROM langile_sailak WHERE id_saila = ?";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setInt(1, idSaila);
        pst.executeUpdate();
    }
}

/**
 * Langile sail baten datuak eguneratzen ditu.
 *
 * @param idSaila      Aldatu nahi den sailaren IDa.
 * @param izena        Sailaren izen berria.
 * @param kokapena    Sailaren kokapen berria.
 * @param deskribapena Sailaren deskribapen berria.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void langileSailaEditatu(int idSaila, String izena, String kokapena,
String deskribapena)
    throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {

```

```

        String sql = "UPDATE langile_sailak SET izena = ?, kokapena = ?, deskribapena = ? WHERE id_saila = ?";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setString(1, izena);
        pst.setString(2, kokapena);
        pst.setString(3, deskribapena);
        pst.setInt(4, idSaila);
        pst.executeUpdate();
    }
}

/***
 * Sail baten informazioa lortzen du IDaren bidez.
 *
 * @param idSaila Sailaren IDa.
 * @return LangileSaila objektua.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public LangileSaila langileSailaikusi(int idSaila) throws SQLException {
    LangileSaila saila = null;
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "SELECT * FROM langile_sailak WHERE id_saila = ?";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setInt(1, idSaila);
        java.sql.ResultSet rs = pst.executeQuery();
        if (rs.next()) {
            saila = new LangileSaila(
                rs.getInt("id_saila"),
                rs.getString("izena"),
                rs.getString("kokapena"),
                rs.getString("deskribapena"));
        }
    }
    return saila;
}

/***
 * Sail guztiak zerrendatzen ditu.
 *
 * @return LangileSaila objektuen zerrenda.
 */
public java.util.ArrayList<LangileSaila> sailakGuztiakIkusi() {
    java.util.ArrayList<LangileSaila> zerrenda = new java.util.ArrayList<>();
    String sql = "SELECT * FROM langile_sailak";
    try (Connection kon = DB_Konexioa.konektatu();
         PreparedStatement pst = kon.prepareStatement(sql);
         java.sql.ResultSet rs = pst.executeQuery()) {
        while (rs.next()) {
            zerrenda.add(new LangileSaila(
                rs.getInt("id_saila"),
                rs.getString("izena"),
                rs.getString("kokapena"),
                rs.getString("deskribapena")));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

        return zerrenda;
    }

    /**
     * Fitxaketa guztiak ikusteko metodoa.
     *
     * @return Fitxaketa objektuen zerrenda.
     */
    public java.util.ArrayList<Fitxaketa> fitxaketaGuztiakIkusi() {
        java.util.ArrayList<Fitxaketa> zerrenda = new java.util.ArrayList<>();
        String galdera = "SELECT * FROM fitxaketak ORDER BY id_fitxaketa DESC";
        try (Connection kon = DB_Konexioa.konektatu()) {
            PreparedStatement pst = kon.prepareStatement(galdera);
            java.sql.ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                zerrenda.add(new Fitxaketa(
                    rs.getInt("id_fitxaketa"),
                    rs.getInt("langilea_id"),
                    rs.getDate("data"),
                    rs.getTime("ordua"),
                    rs.getString("mota"),
                    rs.getTimestamp("eguneratze_data")));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return zerrenda;
    }

    /**
     * Fitxaketa bat ezabatzen du.
     *
     * @param idFitxaketa Ezabatu nahi den fitxaketaren IDa.
     * @throws SQLException Datu-basean errorea gertatzen bada.
     */
    public void fitxaketaEzabatu(int idFitxaketa) throws SQLException {
        try (Connection kon = DB_Konexioa.konektatu()) {
            String sql = "DELETE FROM fitxaketak WHERE id_fitxaketa = ?";
            PreparedStatement pst = kon.prepareStatement(sql);
            pst.setInt(1, idFitxaketa);
            pst.executeUpdate();
        }
    }

    /**
     * Fitxaketa baten datuak egunearatzen ditu.
     *
     * @param idFitxaketa Aldatu nahi den fitxaketaren IDa.
     * @param data          Data berria.
     * @param ordua         Ordu berria.
     * @param mota          Mota berria (SARRERA/IRTEERA).
     * @throws SQLException Datu-basean errorea gertatzen bada.
     */
    public void fitxaketaEditatu(int idFitxaketa, java.sql.Date data, java.sql.Time
ordua, String mota)
        throws SQLException {
        try (Connection kon = DB_Konexioa.konektatu()) {

```

```

        String sql = "UPDATE fitxaketak SET data = ?, ordua = ?, mota = ?,
egunerazte_data = NOW() WHERE id_fitxaketa = ?";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setDate(1, data);
        pst.setTime(2, ordua);
        pst.setString(3, mota);
        pst.setInt(4, idFitxaketa);
        pst.executeUpdate();
    }
}

/***
 * Fitxaketa berri bat sortzen du.
 *
 * @param langileaId Fitxaketa egin duen langilearen IDa.
 * @param data Fitxaketaren data.
 * @param ordua Fitxaketaren ordua.
 * @param mota Fitxaketaren mota.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void fitxaketaSortu(int langileaId, java.sql.Date data, java.sql.Time
ordua, String mota)
        throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "INSERT INTO fitxaketak (langilea_id, data, ordua, mota)
VALUES (?, ?, ?, ?)";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setInt(1, langileaId);
        pst.setDate(2, data);
        pst.setTime(3, ordua);
        pst.setString(4, mota);
        pst.executeUpdate();
    }
}

/***
 * Hornitzaire baten datuak ikusten ditu.
 *
 * @param idHornitzairea Hornitzairearen IDa.
 * @return Hornitzairea objektua.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public Hornitzairea hornitzaireaIkusi(int idHornitzairea) throws SQLException {
    Hornitzairea hornitzairea = null;
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "SELECT * FROM hornitzaireak WHERE id_hornitzairea = ?";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setInt(1, idHornitzairea);
        java.sql.ResultSet rs = pst.executeQuery();
        if (rs.next()) {
            hornitzairea = new Hornitzairea(
                rs.getInt("id_hornitzairea"),
                rs.getString("izena_soziala"),
                rs.getString("nan_ifz"),
                rs.getString("kontaktu_pertsona"),
                rs.getString("helbidea"),
                rs.getInt("herria_id"),

```

```

                rs.getString("posta_kodea"),
                rs.getString("telefonoa"),
                rs.getString("emaila"),
                rs.getString("hizkuntza"),
                rs.getString("pasahitza"),
                rs.getBoolean("aktibo"),
                rs.getTimestamp("eguneratze_data"));
            }
        }
        return hornitzalea;
    }

    /**
     * Hornitzale bat ezabatzen du.
     *
     * @param idHornitzalea Ezabatu nahi den hornitzalearen IDa.
     * @throws SQLException Datu-basean errorea gertatzen bada.
     */
    public void hornitzaleaEzabatu(int idHornitzalea) throws SQLException {
        try (Connection kon = DB_Konexioa.konektatu()) {
            String sql = "DELETE FROM hornitzaleak WHERE id_hornitzalea = ?";
            PreparedStatement pst = kon.prepareStatement(sql);
            pst.setInt(1, idHornitzalea);
            pst.executeUpdate();
        }
    }

    /**
     * Hornitzale baten datuak eguneraztzen ditu.
     *
     * @param idHornitzalea Aldatu nahi den hornitzalearen IDa.
     * @param izenaSoziala Izen sozial berria.
     * @param nan NAN edo IFZ berria.
     * @param kontaktuPertsona Kontaktu pertsona berria.
     * @param helbidea Helbide berria.
     * @param herriaId Herriaren ID berria.
     * @param postaKodea Posta kode berria.
     * @param telefonoa Telefono berria.
     * @param emaila Email berria.
     * @param hizkuntza Hizkuntza berria.
     * @throws SQLException Datu-basean errorea gertatzen bada.
     */
    public void hornitzaleaEditatu(int idHornitzalea, String izenaSoziala, String
nan, String kontaktuPertsona,
                                    String helbidea, int herriaId, String postaKodea, String telefonoa,
String emaila, String hizkuntza)
        throws SQLException {
        try (Connection kon = DB_Konexioa.konektatu()) {
            String sql = "UPDATE hornitzaleak SET izena_soziala = ?, nan_ifz = ?, kontaktu_pertsona = ?, helbidea = ?, herria_id = ?, posta_kodea = ?, telefonoa = ?, emaila = ?, hizkuntza = ?, eguneratze_data = NOW() WHERE id_hornitzalea = ?";
            PreparedStatement pst = kon.prepareStatement(sql);
            pst.setString(1, izenaSoziala);
            pst.setString(2, nan);
            pst.setString(3, kontaktuPertsona);
            pst.setString(4, helbidea);
            pst.setInt(5, herriaId);

```

```

        pst.setString(6, postaKodea);
        pst.setString(7, telefonoa);
        pst.setString(8, emaila);
        pst.setString(9, hizkuntza);
        pst.setInt(10, idHornitzalea);
        pst.executeUpdate();
    }
}

/***
 * Herri guztien zerrenda lortzen du.
 *
 * @return Herria objektuen zerrenda.
 */
public java.util.ArrayList<Herria> herriakIkusi() {
    java.util.ArrayList<Herria> zerrenda = new java.util.ArrayList<>();
    String sql = "SELECT * FROM herriak";
    try (Connection kon = DB_Konexioa.konektatu()) {
        PreparedStatement pst = kon.prepareStatement(sql);
        java.sql.ResultSet rs = pst.executeQuery();
        while (rs.next()) {
            zerrenda.add(new Herria(
                rs.getInt("id_herria"),
                rs.getString("izena"),
                rs.getString("lurraldea"),
                rs.getString("nazioa")));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return zerrenda;
}

/***
 * Herri berri bat sortzen du.
 *
 * @param izena      Herriaren izena.
 * @param lurraldea Lurraldea (Probintzia).
 * @param nazioa     Nazioa.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void herriBerriaSortu(String izena, String lurraldea, String nazioa)
throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "INSERT INTO herriak (izena, lurraldea, nazioa) VALUES (?, ?, ?)";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setString(1, izena);
        pst.setString(2, lurraldea);
        pst.setString(3, nazioa);
        pst.executeUpdate();
    }
}

/***
 * Herri bat ezabatzen du.
 *
 */

```

```

* @param idHerria Ezabatu nahi den herriaren IDa.
* @throws SQLException Datu-basean errorea gertatzen bada.
*/
public void herriaEzabatu(int idHerria) throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "DELETE FROM herriak WHERE id_herria = ?";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setInt(1, idHerria);
        pst.executeUpdate();
    }
}

/***
 * Herri baten datuak eguneratzen ditu.
*
* @param idHerria Aldatu nahi den herriaren IDa.
* @param izena Izen berria.
* @param lurrealdea Lurralde berria.
* @param nazioa Nazio berria.
* @throws SQLException Datu-basean errorea gertatzen bada.
*/
public void herriaEditatu(int idHerria, String izena, String lurrealdea, String nazioa) throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        String sql = "UPDATE herriak SET izena = ?, lurrealdea = ?, nazioa = ? WHERE id_herria = ?";
        PreparedStatement pst = kon.prepareStatement(sql);
        pst.setString(1, izena);
        pst.setString(2, lurrealdea);
        pst.setString(3, nazioa);
        pst.setInt(4, idHerria);
        pst.executeUpdate();
    }
}

/***
 * Langile baten kurrikuluma (PDF BLOB) lortzen du eta irekitzen du.
*
* @param idLangilea Langilearen IDa.
* @throws Exception Errorea datuak lortzean edo fitxategia irekitzean.
*/
public void kurrikulumaIkusi(int idLangilea) throws Exception {
    String sql = "SELECT kurrikuluma FROM langileak WHERE id_langilea = ?";
    try (Connection kon = DB_Konexioa.konektatu()) {
        PreparedStatement pst = kon.prepareStatement(sql));
        pst.setInt(1, idLangilea);
        try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                byte[] pdfBytes = rs.getBytes("kurrikuluma");
                if (pdfBytes != null & pdfBytes.length > 0) {
                    File tempFile = File.createTempFile("kurrikuluma_" +
idLangilea + "_", ".pdf");
                    tempFile.deleteOnExit();
                    try (FileOutputStream fos = new FileOutputStream(tempFile)) {
                        fos.write(pdfBytes);
                    }
                }
            }
        }
    }
}

```

```
        if (Desktop.isDesktopSupported()) {
            Desktop.getDesktop().open(tempFile);
        } else {
            throw new Exception(
                "Sistemak ez du fitxategiak irekitzeko aukera
onartzen (Desktop ez da bateragarria).");
        }
    } else {
        throw new Exception("Ez dago kurrikulumik igota");
    }
} else {
    throw new Exception("Ez da langilea aurkitu.");
}
}
}
```

Akatsa.java

```
package model;

/**
 * Akatsa klasea.
 * Sisteman gertatzen diren akatsak edo erroreak kudeatzeko klasea.
 * Akatsaren IDa, izena eta deskribapena gordetzen ditu.
 */
public class Akatsa {
    private int idAkatsa;
    private String izena;
    private String deskribapena;

    /**
     * Akatsa eraikitzalea.
     *
     * @param idAkatsa      Akatsaren IDa.
     * @param izena         Akatsaren izena.
     * @param deskribapena Akatsaren deskribapena.
     */
    public Akatsa(int idAkatsa, String izena, String deskribapena) {
        this.idAkatsa = idAkatsa;
        this.izena = izena;
        this.deskribapena = deskribapena;
    }

    /**
     * Akatsaren IDa lortzen du.
     *
     * @return Akatsaren IDa.
     */
    public int getIdAkatsa() {
```

```

        return idAkatsa;
    }

    /**
     * Akatsaren IDa ezartzen du.
     *
     * @param idAkatsa Akatsaren ID berria.
     */
    public void setIdAkatsa(int idAkatsa) {
        this.idAkatsa = idAkatsa;
    }

    /**
     * Akatsaren izena lortzen du.
     *
     * @return Akatsaren izena.
     */
    public String getIzena() {
        return izena;
    }

    /**
     * Akatsaren izena ezartzen du.
     *
     * @param izena Akatsaren izen berria.
     */
    public void setIzena(String izena) {
        this.izena = izena;
    }

    /**
     * Akatsaren deskribapena lortzen du.
     *
     * @return Akatsaren deskribapena.
     */
    public String getDeskribapena() {
        return deskribapena;
    }

    /**
     * Akatsaren deskribapena ezartzen du.
     *
     * @param deskribapena Akatsaren deskribapen berria.
     */
    public void setDeskribapena(String deskribapena) {
        this.deskribapena = deskribapena;
    }
}

```

Bezeroa.java

```

package model;

import java.sql.Date;
import java.sql.Timestamp;

/**
 * Bezeroa klasea.
 * Pertsona klasaren azpiklasea da, eta bezeroen informazio espezifikoa
 * gordetzen du.
 * Bezeroen sexua eta ordainketa txartela kudeatzen ditu.
 */
public class Bezeroa extends Pertsona {
    private String sexua;
    private String bezeroOrdainketaTxartela;

    /**
     * Bezeroa eraikitzalea.
     * Bezero berri bat sortzen du emandako datuekin.
     *
     * @param idBezeroa           Bezeroaren IDa.
     * @param izenaEdoSoziala     Bezeroaren izena edo izen soziala.
     * @param abizena              Bezeroaren abizena.
     * @param ifzNan                NAN edo IFZ zenbakia.
     * @param jaiotzaData          Jaiotza data.
     * @param sexua                  Bezeroaren sexua.
     * @param bezeroOrdainketaTxartela Ordainketa txartelaren zenbakia.
     * @param helbidea              Bezeroaren helbidea.
     * @param herriaId               Herriaren IDa.
     * @param postaKodea             Posta kodea.
     * @param telefonoa              Telefono zenbakia.
     * @param emaila                  Email helbidea.
     * @param hizkuntza              Hizkuntza lehenetsia.
     * @param pasahitza              Pasahitza.
     * @param altaData                 Alta data.
     * @param egeneratzeData          Azken egeneratze data.
     * @param aktibo                   Bezeroa aktibo dagoen edo ez.
     */
    public Bezeroa(int idBezeroa, String izenaEdoSoziala, String abizena, String
ifzNan, Date jaiotzaData,
                  String sexua, String bezeroOrdainketaTxartela, String helbidea, int
herriaId, String postaKodea,
                  String telefonoa, String emaila, String hizkuntza, String pasahitza,
Timestamp altaData,
                  Timestamp egeneratzeData, boolean aktibo) {
        super(idBezeroa, izenaEdoSoziala, abizena, ifzNan, jaiotzaData, helbidea,
herriaId, postaKodea,
                  telefonoa, emaila, hizkuntza, pasahitza, aktibo, altaData,
egeneratzeData);
        this.sexua = sexua;
        this.bezeroOrdainketaTxartela = bezeroOrdainketaTxartela;
    }

    /**
     * Bezeroaren IDa lortzen du.
     *
     * @return IDa.

```

```
/*
public int getIdBezeroa() {
    return this.id;
}

/** 
 * Bezeroaren IDa ezartzen du.
 *
 * @param idBezeroa ID berria.
 */
public void setIdBezeroa(int idBezeroa) {
    this.id = idBezeroa;
}

/** 
 * Bezeroaren izena edo izen soziala lortzen du.
 *
 * @return Izena edo Izen Soziala.
 */
public String getIzenaEdoSoziala() {
    return this.izena;
}

/** 
 * Bezeroaren izena edo izen soziala ezartzen du.
 *
 * @param izenaEdoSoziala Izen berria.
 */
public void setIzenaEdoSoziala(String izenaEdoSoziala) {
    this.izena = izenaEdoSoziala;
}

/** 
 * NAN edo IFZ lortzen du.
 *
 * @return NAN edo IFZ.
 */
public String getIfzNan() {
    return this.nanIfz;
}

/** 
 * NAN edo IFZ ezartzen du.
 *
 * @param ifzNan NAN edo IFZ berria.
 */
public void setIfzNan(String ifzNan) {
    this.nanIfz = ifzNan;
}

/** 
 * Bezeroaren sexua lortzen du.
 *
 * @return Sexua.
 */
public String getSexua() {
    return sexua;
}
```

```

}

/**
 * Bezeroaren sexua ezartzen du.
 *
 * @param sexua Sexu berria.
 */
public void setSexua(String sexua) {
    this.sexua = sexua;
}

/**
 * Bezeroaren ordainketa txartela lortzen du.
 *
 * @return Ordainketa txartela.
 */
public String getBezeroOrdainketaTxartela() {
    return bezeroOrdainketaTxartela;
}

/**
 * Bezeroaren ordainketa txartela ezartzen du.
 *
 * @param bezeroOrdainketaTxartela Txartel zenbaki berria.
 */
public void setBezeroOrdainketaTxartela(String bezeroOrdainketaTxartela) {
    this.bezeroOrdainketaTxartela = bezeroOrdainketaTxartela;
}

}

```

Biltegia.java

```

package model;

/**
 * Biltegia klasea.
 * Biltegien informazioa kudeatzeko klasea.
 * Biltegiaren IDa, izena eta SKU kodea gordetzen ditu.
 */
public class Biltegia {
    private int idBiltegia;
    private String izena;
    private String biltegiSku;

    /**
     * Biltegia eraikitzalea.
     *
     * @param idBiltegia Biltegiaren IDa.
     * @param izena Biltegiaren izena.
     * @param biltegiSku Biltegiaren SKU kodea.
     */
}
```

```

/*
public Biltegia(int idBiltegia, String izena, String biltegiSku) {
    this.idBiltegia = idBiltegia;
    this.izena = izena;
    this.biltegiSku = biltegiSku;
}

/**
 * Biltegiaren IDa lortzen du.
 *
 * @return IDa.
 */
public int getIdBiltegia() {
    return idBiltegia;
}

public void setIdBiltegia(int idBiltegia) {
    this.idBiltegia = idBiltegia;
}

/**
 * Biltegiaren izena lortzen du.
 *
 * @return Izena.
 */
public String getIzena() {
    return izena;
}

public void setIzena(String izena) {
    this.izena = izena;
}

/**
 * Biltegiaren SKU kodea lortzen du.
 *
 * @return SKU kodea.
 */
public String getBiltegiSku() {
    return biltegiSku;
}

public void setBiltegiSku(String biltegiSku) {
    this.biltegiSku = biltegiSku;
}
}

```

BiltegiLangilea.java

```
package model;
```

```

import db.DB_Konexioa;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 * BiltegiLangilea klasea.
 * Langilea klasaren azpiklasea da, eta biltegiko langileen funtzioak ditu.
 * Produktuak, sarrerak, biltegiak kudeatzeko metodoak eskaintzen ditu.
 */
public class BiltegiLangilea extends Langilea {

    /**
     * BiltegiLangilea eraikitzailea.
     * Langilea objektu batetik abiatuta BiltegiLangilea sortzen du.
     *
     * @param l Langilea objektua.
     */
    public BiltegiLangilea(Langilea l) {
        super(l.getIdLangilea(), l.getIzena(), l.getAbizena(), l.getNan(),
l.getJaiotzaData(), l.getHerriaId(),
                l.getHelbidea(), l.getPostaKodea(), l.getTelefonoa(),
l.getEmaila(), l.getHizkuntza(),
                l.getPasahitza(), l.getSaltoTxartelaUid(), l.getAltaData(),
l.getEgeneratzeData(),
                l.isAktibo(), l.getSailaId(), l.getIban(), l.getKurrikuluma());
    }

    /**
     * Biltegi berri bat sortzen du.
     *
     * @param izena Biltegiaren izena.
     * @param sku   Biltegiaren SKU kodea.
     * @throws SQLException Datu-basean errorea gertatzen bada.
     */
    public void biltegiaSortu(String izena, String sku) throws SQLException {
        String sql = "INSERT INTO biltegiak (izena, biltegi_sku) VALUES (?, ?)";
        try (Connection kon = DB_Konexioa.konektatu();
             PreparedStatement pst = kon.prepareStatement(sql)) {
            pst.setString(1, izena);
            pst.setString(2, sku);

            pst.executeUpdate();
        }
    }

    /**
     * Biltegi bat ezabatzen du.
     *
     * @param idBiltegia Ezabatu nahi den biltegiaren IDa.
     * @throws SQLException Datu-basean errorea gertatzen bada edo biltegiak
     *                     produktuak baditu.
     */
    public void biltegiaEzabatu(int idBiltegia) throws SQLException {
        try (Connection kon = DB_Konexioa.konektatu()) {
            PreparedStatement pstCheck = kon.prepareStatement("SELECT COUNT(*) FROM

```

```

produktuak WHERE biltegi_id = ?");
        pstCheck.setInt(1, idBiltegia);
        ResultSet rs = pstCheck.executeQuery();
        if (rs.next() && rs.getInt(1) > 0) {
            throw new SQLException("EZIN DA EZABATU: Produktuak ditu
barruan.");
        }
        String sql = "DELETE FROM biltegiak WHERE id_biltegia = ?";
        try (PreparedStatement pst = kon.prepareStatement(sql)) {
            pst.setInt(1, idBiltegia);

            pst.executeUpdate();
        }
    }

/**
 * Biltegi baten datuak eguneratzen ditu.
 *
 * @param idBiltegia Aldatu nahi den biltegiaren IDa.
 * @param izena Izen berria.
 * @param sku SKU kode berria.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void biltegiaEditatu(int idBiltegia, String izena, String sku) throws
SQLException {
    String sql = "UPDATE biltegiak SET izena = ?, biltegi_sku = ? WHERE
id_biltegia = ?";
    try (Connection kon = DB_Konexioa.konektatu();
         PreparedStatement pst = kon.prepareStatement(sql)) {
        pst.setString(1, izena);
        pst.setString(2, sku);
        pst.setInt(3, idBiltegia);
        pst.executeUpdate();
    }
}

/**
 * Hornitzaire berri bat sortzen du.
 *
 * @param izena Hornitzairearen izena.
 * @param ifz Hornitzairearen IFZ.
 * @param emaila Hornitzairearen emaila.
 * @return Sortutako hornitzairearen IDa.
 * @throws SQLException Datu-basean errorea gertatzen bada edo hornitzairea
existitzen bada.
 */
public int hornitzaireBerriaSortu(String izena, String ifz, String emaila)
throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        PreparedStatement pstCheck = kon
            .prepareStatement("SELECT COUNT(*) FROM hornitzaireak WHERE
emaila = ? OR ifz_nan = ?");
        pstCheck.setString(1, emaila);
        pstCheck.setString(2, ifz);
        ResultSet rsCheck = pstCheck.executeQuery();
        if (rsCheck.next() && rsCheck.getInt(1) > 0) {

```

```

        throw new SQLException("ERROREA: Hornitzale hori existitzen da
jada.");
    }

    String sql = "INSERT INTO hornitzaileak (izena_soziala, ifz_nan,
emaila, pasahitza, helbidea, herria_id, posta_kodea) VALUES (?, ?, ?, ?, '1234',
'Zehaztugabea', 1, '00000')";

    try (PreparedStatement pst = kon.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {
        pst.setString(1, izena);
        pst.setString(2, ifz);
        pst.setString(3, emaila);
        pst.executeUpdate();
        ResultSet rsKey = pst.getGeneratedKeys();
        if (rsKey.next()) {
            return rsKey.getInt(1);
        }
    }
}

throw new SQLException("Errorea hornitzalea sortzean.");
}

/**
 * Produktuen sarrera berri bat sortzen du.
 * Transaction bidez kudeatzen da osotasuna bermatzeko.
 *
 * @param hornitzaileId Hornitzalearen IDa.
 * @param produktuak Sartuko diren produktuen zerrenda.
 * @param lerroak Sarrera lerroen zerrenda.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void produktuSarreraBerriaSortu(int hornitzaileaId, List<Produktua>
produktuak, List<SarreraLerroa> lerroak)
    throws SQLException {
    Connection con = null;
    try {
        con = DB_Konexioa.konektatu();
        con.setAutoCommit(false);

        // 1. Sarrera sortu
        String sqlSarrera = "INSERT INTO sarrerak (hornitzalea_id,
langilea_id, sarrera_egoera) VALUES (?, ?, 'Bidean')";
        PreparedStatement pstSarrera = con.prepareStatement(sqlSarrera,
Statement.RETURN_GENERATED_KEYS);
        pstSarrera.setInt(1, hornitzaileaId);
        pstSarrera.setInt(2, this.getIdLangilea());
        pstSarrera.executeUpdate();
        ResultSet rsKeys = pstSarrera.getGeneratedKeys();
        int sarreraId = -1;
        if (rsKeys.next())
            sarreraId = rsKeys.getInt(1);
        else
            throw new SQLException("Ez da sarrera produktu IDrik sortu.");

        // 2. Sortu produktuak eta lerroak
        String sqlProd = "INSERT INTO produktuak (izena, marka, kategoria_id,
mota, biltegi_id, hornitzale_id, stock, produktu_egoera, deskribapena, irudia_url,
produktu_egoera_oharra, salgai) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, 'Zehazteko', ?, ?, ?, ?)";

```

```

0)");
    PreparedStatement pstProd = con.prepareStatement(sqlProd,
Statement.RETURN_GENERATED_KEYS);
    String sqlLerroa = "INSERT INTO sarrera_lerroak (sarrera_id,
produktua_id, kantitatea, sarrera_lerro_egoera) VALUES (?, ?, ?, 'Bidean')";
    PreparedStatement pstLerroa = con.prepareStatement(sqlLerroa);

    for (int i = 0; i < produktuak.size(); i++) {
        Produktua p = produktuak.get(i);
        SarreraLerroa l = lerroak.get(i);

        pstProd.setString(1, p.getIzena());
        pstProd.setString(2, p.getMarka());
        pstProd.setInt(3, p.getKategoriaId());
        pstProd.setString(4, p.getMota());
        pstProd.setInt(5, p.getBiltegiId());
        pstProd.setInt(6, hornitzailleaId);
        pstProd.setInt(7, p.getStock()); // Initial stock = quantity in
line? Or 0? Logic in MenuLogistika uses
                                            // stock = kanti
        pstProd.setString(8, p.getDeskribapena());
        pstProd.setString(9, p.getIrudiaUrl());
        pstProd.setString(10, p.getProduktuEgoeraOharra());
        pstProd.executeUpdate();

        ResultSet rsProdKey = pstProd.getGeneratedKeys();
        int prodId = -1;
        if (rsProdKey.next())
            prodId = rsProdKey.getInt(1);

        pstLerroa.setInt(1, sarreraId);
        pstLerroa.setInt(2, prodId);
        pstLerroa.setInt(3, l.getKantitatea());
        pstLerroa.executeUpdate();
    }

    con.commit();
} catch (SQLException e) {
    if (con != null)
        con.rollback();
    throw e;
} finally {
    if (con != null) {
        con.setAutoCommit(true);
        con.close();
    }
}
}

/**
 * Produktu baten egoeraren oharra eguneratzen du.
 *
 * @param idProduktua Produktuaren IDa.
 * @param oharra      Ohar berria.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void produktuEgoeraOharraJarri(int idProduktua, String oharra) throws

```

```

SQLException {
    // Funtzionalitate hau UI-ko sorkuntzaren zati zen, baina beharbada
bereizita
    // behar da?
    // UI-ak ez zuen "Gehitu Oharra" botoi esplizitrik lehendik zeuden
    // produktuentzat.
    String sql = "UPDATE produktuak SET produktu_egoera_oharra = ? WHERE
id_produktua = ?";
    try (Connection kon = DB_Konexioa.konektatu();
        PreparedStatement pst = kon.prepareStatement(sql)) {
        pst.setString(1, oharra);
        pst.setInt(2, idProduktua);
        pst.executeUpdate();
    }
}

/**
 * Produktu bat biltegi batetik bestera mugitzen du.
 *
 * @param idProduktua Produktuaren IDa.
 * @param idBiltegia Biltegi berriaren IDa.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void produktuarenBiltegiaAldatu(int idProduktua, int idBiltegia) throws
SQLException {
    String sql = "UPDATE produktuak SET biltegi_id = ? WHERE id_produktua = ?";
    try (Connection kon = DB_Konexioa.konektatu();
        PreparedStatement pst = kon.prepareStatement(sql)) {
        pst.setInt(1, idBiltegia);
        pst.setInt(2, idProduktua);
        pst.executeUpdate();
    }
}
// Object[] zerrenda bat itzultzea hobeto izan daiteke UI-rako, baina saia
// gaitezten itzultzen
// Sarrera objektuak.
// UI-ak taulak lotzen ditu Hornitzairearen Izena erakusteko. Sarrera modeloak
// ez du
// Hornitzairearen Izenik.
// Barne-klase berri baten zerrenda bat edo DTO bat itzuliko dut, ala Sarrera
// soilik eta
// informazio gehigarria geroago lortu?
// Hasi beharrezkoa denarekin soilik. UI-ak konsulta pertsonalizatu bat
// erabiltzen du.
// Konsulta exekutatuko dut eta ResultSet logika itzuliko dut, edo beharbada
// hobeto,
// TableModel egiturarekin bat datorren Object[] zerrenda bat itzuli.

/**
 * Produktu sarrerak ikusten ditu, egoeraren arabera iragazita.
 *
 * @param egoeraIragazkia Iragazkia ("Bidean", "Jasota" edo null denak
 *                         ikusteko).
 * @return Objektu array zerrenda bat sarreren informazioarekin.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public List<Object[]> produktuSarrerakIkusi(String egoeraIragazkia) throws

```

```

SQLException {
    String baseSql = "SELECT s.id_sarrera, h.izena_soziala AS Hornitzalea,
s.data, s.sarrera_egoera FROM sarrerak s JOIN hornitzaleak h ON s.hornitzalea_id
= h.id_hornitzalea ";
    String sql = baseSql;
    if ("Bidean".equals(egoeraIragazkia))
        sql += " WHERE s.sarrera_egoera = 'Bidean' ";
    else if ("Jasota".equals(egoeraIragazkia))
        sql += " WHERE s.sarrera_egoera = 'Jasota' ";
    sql += " ORDER BY s.data DESC";

    List<Object[]> emaitza = new ArrayList<>();
    try (Connection kon = DB_Konexioa.konektatu()) {
        PreparedStatement pst = kon.prepareStatement(sql)) {
        ResultSet rs = pst.executeQuery();
        ResultSetMetaData meta = rs.getMetaData();
        int colCount = meta.getColumnCount();
        while (rs.next()) {
            Object[] row = new Object[colCount];
            for (int i = 0; i < colCount; i++) {
                row[i] = rs.getObject(i + 1);
            }
            emaitza.add(row);
        }
    }
    return emaitza;
}

/**
 * Sarrera baten egoera eguneratzen du eta bere lerro guztienarekin batera.
 *
 * @param idSarrera Sarreraren IDa.
 * @param egoera Egoera berria.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void produktuSarreraEditatu(int idSarrera, String egoera) throws
SQLException {
    // Cascading: Update Parent -> Update All Lines
    String sql = "UPDATE sarrerak SET sarrera_egoera = ? WHERE id_sarrera = ?";
    String sqlLerroak = "UPDATE sarrera_lerroak SET sarrera_lerro_egoera = ?
WHERE sarrera_id = ?";

    Connection kon = null;
    try {
        kon = DB_Konexioa.konektatu();
        kon.setAutoCommit(false);

        try (PreparedStatement pst = kon.prepareStatement(sql)) {
            pst.setString(1, egoera);
            pst.setInt(2, idSarrera);
            pst.executeUpdate();
        }

        try (PreparedStatement pstLerroak = kon.prepareStatement(sqlLerroak)) {
            pstLerroak.setString(1, egoera);
            pstLerroak.setInt(2, idSarrera);
            pstLerroak.executeUpdate();
        }
    }
}

```

```

        }

        kon.commit();
    } catch (SQLException e) {
        if (kon != null)
            kon.rollback();
        throw e;
    } finally {
        if (kon != null) {
            kon.setAutoCommit(true);
            kon.close();
        }
    }
}

/**
 * Sarrera lerro baten egoera aldatzen du.
 * Lerro guztiak jasota badaude, sarrera osoaren egoera ere eguneratzen da.
 *
 * @param idSarreraLerroa Sarrera lerroaren IDa.
 * @param egoera          Egoera berria.
 * @param idSarrera       Sarreraren IDa (gurasoa).
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void produktuSarreraEgoeraAldatu(int idSarreraLerroa, String egoera, int
idSarrera) throws SQLException {
    try (Connection con = DB_Konexioa.konektatu()) {
        PreparedStatement pst = con.prepareStatement(
            "UPDATE sarrera_lerroak SET sarrera_lerro_egoera = ? WHERE
id_sarrera_lerroa = ?");
        pst.setString(1, egoera);
        pst.setInt(2, idSarreraLerroa);
        if (pst.executeUpdate() > 0) {
            // guraso egoera egiaztatu
            PreparedStatement pstCheck = con.prepareStatement(
                "SELECT COUNT(*) FROM sarrera_lerroak WHERE sarrera_id = ?
AND sarrera_lerro_egoera != 'Jasota'");
            pstCheck.setInt(1, idSarrera);
            ResultSet rs = pstCheck.executeQuery();
            if (rs.next()) {
                String egoeraBerria = (rs.getInt(1) == 0) ? "Jasota" :
"Bidean";
                produktuSarreraEditatu(idSarrera, egoeraBerria);
            }
        }
    }
}

/**
 * Eskaera baten egoera aldatzen du eta bere lerro guztienarekin bai.
 *
 * @param idEskaera Eskaeraren IDa.
 * @param egoera    Egoera berria.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void produktuEskaeraEgoeraAldatu(int idEskaera, String egoera) throws
SQLException {

```

```

// Cascading: Update Parent -> Update All Lines
String sql = "UPDATE eskaerak SET eskaera_egoera = ? WHERE id_eskaera = ?";
String sqlLerroak = "UPDATE eskaera_lerroak SET eskaera_lerro_egoera = ?
WHERE eskaera_id = ?";

Connection kon = null;
try {
    kon = DB_Konexioa.konektatu();
    kon.setAutoCommit(false);

    try (PreparedStatement pst = kon.prepareStatement(sql)) {
        pst.setString(1, egoera);
        pst.setInt(2, idEskaera);
        pst.executeUpdate();
    }

    try (PreparedStatement pstLerroak = kon.prepareStatement(sqlLerroak)) {
        pstLerroak.setString(1, egoera);
        pstLerroak.setInt(2, idEskaera);
        pstLerroak.executeUpdate();
    }

    kon.commit();
} catch (SQLException e) {
    if (kon != null)
        kon.rollback();
    throw e;
} finally {
    if (kon != null)
        kon.setAutoCommit(true);
    kon.close();
}
}

/**
 * Sarrera baten lerroak ikusten ditu.
 *
 * @param idSarrera Sarreraren IDa.
 * @return Objektu array zerrenda sarrera lerroen informazioarekin.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public List<Object[]> produktuSarreraLerroakIkusi(int idSarrera) throws
SQLException {
    String sql = "SELECT sl.id_sarrera_lerroa, p.izena, p.marka, sl.kantitatea,
sl.sarrera_lerro_egoera " +
        "FROM sarrera_lerroak sl " +
        "JOIN produktuak p ON sl.produktua_id = p.id_produktua " +
        "WHERE sl.sarrera_id = ?";
    List<Object[]> emaitza = new ArrayList<>();
    try (Connection kon = DB_Konexioa.konektatu()) {
        PreparedStatement pst = kon.prepareStatement(sql)) {
            pst.setInt(1, idSarrera);
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                emaitza.add(new Object[] {
                    rs.getInt(1),

```

```

                rs.getString(2),
                rs.getString(3),
                rs.getInt(4),
                rs.getString(5)
            });
        }
    }
    return emaitza;
}

/**
 * Sarrera lerro baten egoera aldatzen du eta guraso sarreraren egoera
 * egiaztatzen du.
 *
 * @param idSarreraLerroa Sarrera lerroaren IDa.
 * @param egoera          Egoera berria.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void produktuSarreraLerroEgoeraAldatu(int idSarreraLerroa, String
egoera) throws SQLException {
    // Update Line -> Check Peers -> Update Parent (Internal - No Cascade)
    String sql = "UPDATE sarrera_lerroak SET sarrera_lerro_egoera = ? WHERE
id_sarrera_lerroa = ?";

    try (Connection kon = DB_Konexioa.konektatu()) {
        // 1. Lerroaren egoera aldatu
        try (PreparedStatement pst = kon.prepareStatement(sql)) {
            pst.setString(1, egoera);
            pst.setInt(2, idSarreraLerroa);
            pst.executeUpdate();
        }

        // 2. Gurasoaren ID-a lortu
        int idSarrera = -1;
        try (PreparedStatement pstGetId = kon
                .prepareStatement("SELECT sarrera_id FROM sarrera_lerroak WHERE
id_sarrera_lerroa = ?")) {
            pstGetId.setInt(1, idSarreraLerroa);
            ResultSet rs = pstGetId.executeQuery();
            if (rs.next())
                idSarrera = rs.getInt(1);
        }

        if (idSarrera != -1) {
            // 3. Egiaztatu gainerako lerroak - 'Ezabatua' lerroak ez kontuan
hartu
            boolean allJasota = true;
            try (PreparedStatement pstCheck = kon.prepareStatement(
                    "SELECT COUNT(*) FROM sarrera_lerroak WHERE sarrera_id = ?
AND sarrera_lerro_egoera != 'Jasota' AND sarrera_lerro_egoera != 'Ezabatua'")) {
                pstCheck.setInt(1, idSarrera);
                ResultSet rs = pstCheck.executeQuery();
                if (rs.next() && rs.getInt(1) > 0) {
                    allJasota = false;
                }
            }
        }
    }
}

```

```

        // 4. Gurasoaren egoera aldatu (Infinitu kaskada saihesteko)
        String newStatus = allJasota ? "Jasota" : "Bidean";
        try (PreparedStatement pstUpdateParent = kon
                .prepareStatement("UPDATE sarrerak SET sarrera_egoera = ?
WHERE id_sarrera = ?")) {
            pstUpdateParent.setString(1, newStatus);
            pstUpdateParent.setInt(2, idSarrera);
            pstUpdateParent.executeUpdate();
        }
    }
}

/**
 * Sarrera bat eta bere lerroak ezabatzen ditu.
 *
 * @param idSarrera Ezabatu nahi den sarreraren IDa.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void produktuSarreraEzabatu(int idSarrera) throws SQLException {
    String sqlLerroak = "DELETE FROM sarrera_lerroak WHERE sarrera_id = ?";
    String sqlSarrera = "DELETE FROM sarrerak WHERE id_sarrera = ?";

    Connection kon = null;
    try {
        kon = DB_Konexioa.konektatu();
        kon.setAutoCommit(false);

        try (PreparedStatement pstLerroak = kon.prepareStatement(sqlLerroak)) {
            pstLerroak.setInt(1, idSarrera);
            pstLerroak.executeUpdate();
        }

        try (PreparedStatement pstSarrera = kon.prepareStatement(sqlSarrera)) {
            pstSarrera.setInt(1, idSarrera);
            pstSarrera.executeUpdate();
        }

        kon.commit();
    } catch (SQLException e) {
        if (kon != null)
            kon.rollback();
        throw e;
    } finally {
        if (kon != null) {
            kon.setAutoCommit(true);
            kon.close();
        }
    }
}

/**
 * Eskaerak ikusten ditu, egoeraren arabera iragazita.
 *
 * @param egoeraIragazkia Iragazkia (Egoera zehatza edo "Denak").
 * @return Objektu array zerrenda eskaeren informazioarekin.
 * @throws SQLException Datu-basean errorea gertatzen bada.

```

```

/*
public List<Object[]> produktuEskaerakIkusi(String egoeraIragazkia) throws
SQLException {
    String sql = "SELECT e.id_eskaera, b.izena_edo_soziala, e.data,
e.guztira_prezioa, e.eskaera_egoera " +
        "FROM eskaerak e " +
        "JOIN bezeroak b ON e.bezeroa_id = b.id_bezeroa ";

    if (egoeraIragazkia != null && !egoeraIragazkia.isEmpty() &&
!"Denak".equals(egoeraIragazkia)) {
        sql += " WHERE e.eskaera_egoera = ?";
    }
    sql += " ORDER BY e.data DESC";

    List<Object[]> emaitza = new ArrayList<>();
    try (Connection kon = DB_Konexioa.konektatu();
         PreparedStatement pst = kon.prepareStatement(sql)) {

        if (egoeraIragazkia != null && !egoeraIragazkia.isEmpty() &&
!"Denak".equals(egoeraIragazkia)) {
            pst.setString(1, egoeraIragazkia);
        }

        ResultSet rs = pst.executeQuery();
        while (rs.next()) {
            emaitza.add(new Object[] {
                rs.getInt(1),
                rs.getString(2),
                rs.getTimestamp(3),
                rs.getBigDecimal(4),
                rs.getString(5)
            });
        }
    }
    return emaitza;
}

/**
 * Eskaera baten lerroak ikusten ditu.
 *
 * @param idEskaera Eskaeraren IDa.
 * @return Objektu array zerrenda eskaera lerroen informazioarekin.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public List<Object[]> produktuEskaeraLerroakIkusi(int idEskaera) throws
SQLException {
    String sql = "SELECT el.id_eskaera_lerroa, p.izena, el.kantitatea,
el.unitate_precioa, el.eskaera_lerro_egoera "
        +
        "FROM eskaera_lerroak el " +
        "JOIN produktuak p ON el.productua_id = p.id_productua " +
        "WHERE el.eskaera_id = ?";

    List<Object[]> emaitza = new ArrayList<>();
    try (Connection kon = DB_Konexioa.konektatu();
         PreparedStatement pst = kon.prepareStatement(sql)) {
        pst.setInt(1, idEskaera);
        ResultSet rs = pst.executeQuery();

```

```

        while (rs.next()) {
            emaitza.add(new Object[] {
                rs.getInt(1),
                rs.getString(2),
                rs.getInt(3),
                rs.getBigDecimal(4),
                rs.getString(5)
            });
        }
    }
    return emaitza;
}

/**
 * Eskaera lerro baten egoera aldatzen du eta eskaera osoaren egoera
 * eguneratzen
 * du.
 *
 * @param idEskaeraLerroa Eskaera lerroaren IDa.
 * @param egoera          Egoera berria.
 * @param idEskaera        Eskaeraren IDa (gurasoa).
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void produktuEskaeraLerroEgoeraAldatu(int idEskaeraLerroa, String
egoera, int idEskaera)
    throws SQLException {
    try (Connection kon = DB_Konexioa.konektatu()) {
        // 1. Lerroaren egoera egunerau
        String sql = "UPDATE eskaera_lerroak SET eskaera_lerro_egoera = ? WHERE
id_eskaera_lerroa = ?";
        try (PreparedStatement pst = kon.prepareStatement(sql)) {
            pst.setString(1, egoera);
            pst.setInt(2, idEskaeraLerroa);
            pst.executeUpdate();
        }

        // 2. Eskaera osoaren egoera egunerau
        // LOGIKA ZUZENKETA:
        // - Lerroren bat "Prestatzen" badago -> Eskaera "Prestatzen"
        // - Bestela (denak "Osatua/Bidalita" edo "Ezabatua" badira) ->
        // "Osatua/Bidalita"

        boolean dagoPrestatzen = false;
        String sqlCheck = "SELECT COUNT(*) FROM eskaera_lerroak WHERE
eskaera_id = ? AND eskaera_lerro_egoera = 'Prestatzen'";
        try (PreparedStatement pstABC = kon.prepareStatement(sqlCheck)) {
            pstABC.setInt(1, idEskaera);
            ResultSet rs = pstABC.executeQuery();
            if (rs.next() && rs.getInt(1) > 0) {
                dagoPrestatzen = true;
            }
        }

        if (dagoPrestatzen) {

            try (PreparedStatement pstUpd = kon
                .prepareStatement("UPDATE eskaerak SET eskaera_egoera = ?
```
```

```

 WHERE id_eskaera = ?)) {
 pstUpd.setString(1, "Prestatzen");
 pstUpd.setInt(2, idEskaera);
 pstUpd.executeUpdate();
 }
 } else {

 try (PreparedStatement pstUpd = kon
 .prepareStatement("UPDATE eskaerak SET eskaera_egoera = ?
WHERE id_eskaera = ?)) {
 pstUpd.setString(1, "Osatua/Bidalita");
 pstUpd.setInt(2, idEskaera);
 pstUpd.executeUpdate();
 }
 }
}
}

```

## Eramangarria.java

```

package model;

import java.math.BigDecimal;
import java.sql.Timestamp;

/**
 * Eramangarria klasea.
 * Produktua klasaren azpiklasea da, eta ordenagailu eramangarrien ezaugarri
 * espezifikoak gordetzen ditu.
 * Prozesadorea, RAMa, diskoa, pantaila, bateria eta sistema eragilea bezalako
 * datuak kudeatzen ditu.
 */
public class Eramangarria extends Produktua {
 private String prozesadorea;
 private int ramGb;
 private int diskoaGb;
 private BigDecimal pantailaTamaina;
 private int bateriaWh;
 private String sistemaEragilea;
 private BigDecimal pisuaKg;

 /**
 * Eramangarria eraikitzalea.
 * Produktuaren oinarritzko datuez gain, eramangarriaren ezaugarriak
 * inizializatzen ditu.
 *
 * @param idProduktua Produktuaren IDa.
 * @param hornitztaileId Hornitztailearen IDa.
 * @param kategoriaId Kategoriaren IDa.
 * @param izena Izena.
 */
}
```

```

* @param marka Marka.
* @param mota Mota (Eramangarria).
* @param deskribapena Deskribapena.
* @param irudiaUrl Irudiaren URLa.
* @param biltegiId Biltegiaren IDa.
* @param produktuEgoera Produktuaren egoera.
* @param produktuEgoeraOharra Egoeraren oharra.
* @param salgai Salgai dagoen edo ez.
* @param salmentaPrezioa Salmenta prezioa.
* @param stock Stock kopurua.
* @param eskaintza Eskaintza prezioa (baldin badago).
* @param zergakEhunekoa Zergen ehunekoa.
* @param sortzeData Sortze data.
* @param eguneratzeData Eguneratze data.
* @param prozesadorearea Prozesadorearea.
* @param ramGb RAM memoria GBtan.
* @param diskoaGb Diskoaren kapazitatea GBtan.
* @param pantailaTamaina Pantailaren tamaina hazbetetan.
* @param bateriaWh Bateriaren kapazitatea Wh-tan.
* @param sistemaEragilea Sistema eragilea.
* @param pisuaKg Pisua KGtan.
*/
public Eramangarria(int idProduktua, int hornitzaireId, int kategoriaId, String
izen, String marka, String mota,
String deskribapena, String irudiaUrl, Integer biltegiId, String
produktuEgoera,
String produktuEgoeraOharra, boolean salgai, BigDecimal
salmentaPrezioa, int stock,
BigDecimal eskaintza, BigDecimal zergakEhunekoa, Timestamp sortzeData,
Timestamp eguneratzeData,
String prozesadorearea, int ramGb, int diskoaGb, BigDecimal
pantailaTamaina,
int bateriaWh, String sistemaEragilea, BigDecimal pisuaKg) {
super(idProduktua, hornitzaireId, kategoriaId, izena, marka, mota,
deskribapena, irudiaUrl, biltegiId,
produktuEgoera, produktuEgoeraOharra, salgai, salmentaPrezioa,
stock, eskaintza, zergakEhunekoa,
sortzeData, eguneratzeData);
this.prozesadorearea = prozesadorearea;
this.ramGb = ramGb;
this.diskoaGb = diskoaGb;
this.pantailaTamaina = pantailaTamaina;
this.bateriaWh = bateriaWh;
this.sistemaEragilea = sistemaEragilea;
this.pisuaKg = pisuaKg;
}

/**
* Prozesadorearea lortzen du.
*
* @return Prozesadorearea.
*/
public String getProzesadorearea() {
 return prozesadorearea;
}

/**

```

```
* Prozesadorea ezartzen du.
*
* @param prozesadorea Prozesadore berria.
*/
public void setProzesadorea(String prozesadorea) {
 this.prozesadorea = prozesadorea;
}

/**
 * RAM memoria lortzen du.
 *
* @return RAM memoria GBtan.
*/
public int getRamGb() {
 return ramGb;
}

/**
 * RAM memoria ezartzen du.
 *
* @param ramGb RAM memoria berria GBtan.
*/
public void setRamGb(int ramGb) {
 this.ramGb = ramGb;
}

/**
 * Diskoaren kapazitatea lortzen du.
 *
* @return Diskoa GBtan.
*/
public int getDiskoaGb() {
 return diskoaGb;
}

/**
 * Diskoaren kapazitatea ezartzen du.
 *
* @param diskoaGb Disko kapazitate berria GBtan.
*/
public void setDiskoaGb(int diskoaGb) {
 this.diskoaGb = diskoaGb;
}

/**
 * Pantailaren tamaina lortzen du.
 *
* @return Pantailaren tamaina hazbetetan.
*/
public BigDecimal getPantailaTamaina() {
 return pantailaTamaina;
}

/**
 * Pantailaren tamaina ezartzen du.
 *
* @param pantailaTamaina Pantailaren tamaina berria.
*/
```

```
/*
public void setPantailaTamaina(BigDecimal pantailaTamaina) {
 this.pantailaTamaina = pantailaTamaina;
}

/**
 * Bateriaren kapazitatea lortzen du.
 *
 * @return Bateria Wh-tan.
 */
public int getBateriaWh() {
 return bateriaWh;
}

/**
 * Bateriaren kapazitatea ezartzen du.
 *
 * @param bateriaWh Bateria kapazitate berria.
 */
public void setBateriaWh(int bateriaWh) {
 this.bateriaWh = bateriaWh;
}

/**
 * Sistema eragilea lortzen du.
 *
 * @return Sistema eragilea.
 */
public String getSistemaEragilea() {
 return sistemaEragilea;
}

/**
 * Sistema eragilea ezartzen du.
 *
 * @param sistemaEragilea Sistema eragile berria.
 */
public void setSistemaEragilea(String sistemaEragilea) {
 this.sistemaEragilea = sistemaEragilea;
}

/**
 * Pisua lortzen du.
 *
 * @return Pisua KGtan.
 */
public BigDecimal getPisuaKg() {
 return pisuaKg;
}

/**
 * Pisua ezartzen du.
 *
 * @param pisuaKg Pisu berria.
 */
public void setPisuaKg(BigDecimal pisuaKg) {
 this.pisuaKg = pisuaKg;
}
```

```
 }
}
```

# Eskaera.java

```
package model;

import java.math.BigDecimal;
import java.sql.Timestamp;

/**
 * Eskaera klasea.
 * Bezeroen eskaeren informazioa kudeatzen du.
 * Eskaeraren data, prezioa, egoera eta lotutako bezero/langilea gordetzen ditu.
 */
public class Eskaera {
 private int idEskaera;
 private int bezeroaId;
 private Integer langileaId;
 private Timestamp data;
 private Timestamp eguneratzeData;
 private BigDecimal guztiraPrezioa;
 private String fakturaZenbakia;
 private String fakturaUrl;
 private String eskaeraEgoera;

 /**
 * Eskaera eraikitzalea.
 *
 * @param idEskaera Eskaeraren IDa.
 * @param bezeroaId Bezeroaren IDa.
 * @param langileaId Langilearen IDa (aukerakoa).
 * @param data Eskaeraren data.
 * @param eguneratzeData Azken eguneratze data.
 * @param guztiraPrezioa Eskaeraren prezio totala.
 * @param fakturaZenbakia Fakturaren zenbakia (bakarra).
 * @param fakturaUrl Fakturaren URL edo bidea.
 * @param eskaeraEgoera Eskaeraren egoera.
 */
 public Eskaera(int idEskaera, int bezeroaId, Integer langileaId, Timestamp data,
 Timestamp eguneratzeData,
 BigDecimal guztiraPrezioa, String fakturaZenbakia, String fakturaUrl,
 String eskaeraEgoera) {
 this.idEskaera = idEskaera;
 this.bezeroaId = bezeroaId;
 this.langileaId = langileaId;
 this.data = data;
 this.eguneratzeData = eguneratzeData;
 this.guztiraPrezioa = guztiraPrezioa;
 this.fakturaZenbakia = fakturaZenbakia;
 this.fakturaUrl = fakturaUrl;
```

```
 this.eskaeraEgoera = eskaeraEgoera;
 }

 public int getIdEskaera() {
 return idEskaera;
 }

 public void setIdEskaera(int idEskaera) {
 this.idEskaera = idEskaera;
 }

 public int getBezeroaId() {
 return bezeroaId;
 }

 public void setBezeroaId(int bezeroaId) {
 this.bezeroaId = bezeroaId;
 }

 public Integer getLangileaId() {
 return langileaId;
 }

 public void setLangileaId(Integer langileaId) {
 this.langileaId = langileaId;
 }

 public Timestamp getData() {
 return data;
 }

 public void setData(Timestamp data) {
 this.data = data;
 }

 public Timestamp getEguneratzeData() {
 return eguneratzeData;
 }

 public void setEguneratzeData(Timestamp eguneratzeData) {
 this.eguneratzeData = eguneratzeData;
 }

 public BigDecimal getGuztiraPrezioa() {
 return guztiraPrezioa;
 }

 public void setGuztiraPrezioa(BigDecimal guztiraPrezioa) {
 this.guztiraPrezioa = guztiraPrezioa;
 }

 public String getFakturaZenbakia() {
 return fakturaZenbakia;
 }

 public void setFakturaZenbakia(String fakturaZenbakia) {
 this.fakturaZenbakia = fakturaZenbakia;
 }
```

```

}

public String getFakturaUrl() {
 return fakturaUrl;
}

public void setFakturaUrl(String fakturaUrl) {
 this.fakturaUrl = fakturaUrl;
}

public String getEskaeraEgoera() {
 return eskaeraEgoera;
}

public void setEskaeraEgoera(String eskaeraEgoera) {
 this.eskaeraEgoera = eskaeraEgoera;
}

}

```

## EskaeraLerroa.java

```

package model;

import java.math.BigDecimal;

/**
 * EskaeraLerroa klasea.
 * Eskaera baten barruko lerro bakoitza kudeatzen du.
 * Produktuaren IDa, kantitatea, prezioa eta egoera gordetzen ditu.
 */
public class EskaeraLerroa {
 private int idEskaeraLerroa;
 private int eskaeraId;
 private int produktuaId;
 private int kantitatea;
 private BigDecimal unitatePrezioa;
 private String eskaeraLerroEgoera;

 /**
 * EskaeraLerroa eraikitzailea.
 *
 * @param idEskaeraLerroa Lerroaren IDa.
 * @param eskaeraId Eskaeraren IDa.
 * @param produktuaId Produktuaren IDa.
 * @param kantitatea Produktu kantitatea.
 * @param unitatePrezioa Unitateko prezioa.
 * @param eskaeraLerroEgoera Lerroaren egoera.
 */
 public EskaeraLerroa(int idEskaeraLerroa, int eskaeraId, int produktuaId, int
kantitatea, BigDecimal unitatePrezioa,

```

```
 String eskaeraLerroEgoera) {
 this.idEskaeraLerroa = idEskaeraLerroa;
 this.eskaeraId = eskaeraId;
 this.produktuaId = produktuaId;
 this.kantitatea = kantitatea;
 this.unitatePrezioa = unitatePrezioa;
 this.eskaeraLerroEgoera = eskaeraLerroEgoera;
}

/**
 * Lerroaren IDa lortzen du.
 *
 * @return IDa.
 */
public int getIdEskaeraLerroa() {
 return idEskaeraLerroa;
}

/**
 * Lerroaren IDa ezartzen du.
 *
 * @param idEskaeraLerroa ID berria.
 */
public void setIdEskaeraLerroa(int idEskaeraLerroa) {
 this.idEskaeraLerroa = idEskaeraLerroa;
}

/**
 * Eskaeraren IDa lortzen du.
 *
 * @return Eskaeraren IDa.
 */
public int getEskaeraId() {
 return eskaeraId;
}

/**
 * Eskaeraren IDa ezartzen du.
 *
 * @param eskaeraId Eskaeraren ID berria.
 */
public void setEskaeraId(int eskaeraId) {
 this.eskaeraId = eskaeraId;
}

/**
 * Produktuaren IDa lortzen du.
 *
 * @return Produktuaren IDa.
 */
public int getProduktuaId() {
 return produktuaId;
}

/**
 * Produktuaren IDa ezartzen du.
 *
```

```
* @param produktuaId Produktuaren ID berria.
*/
public void setProduktuaId(int produktuaId) {
 this.produktuaId = produktuaId;
}

/**
 * Kantitatea lortzen du.
*
* @return Kantitatea.
*/
public int getKantitatea() {
 return kantitatea;
}

/**
 * Kantitatea ezartzen du.
*
* @param kantitatea Kantitate berria.
*/
public void setKantitatea(int kantitatea) {
 this.kantitatea = kantitatea;
}

/**
 * Unitateko prezioa lortzen du.
*
* @return Prezioa.
*/
public BigDecimal getUnitatePrezioa() {
 return unitatePrezioa;
}

/**
 * Unitateko prezioa ezartzen du.
*
* @param unitatePrezioa Prezio berria.
*/
public void setUnitatePrezioa(BigDecimal unitatePrezioa) {
 this.unitatePrezioa = unitatePrezioa;
}

/**
 * Lerroaren egoera lortzen du.
*
* @return Egoera.
*/
public String getEskaeraLerroEgoera() {
 return eskaeraLerroEgoera;
}

/**
 * Lerroaren egoera ezartzen du.
*
* @param eskaeraLerroEgoera Egoera berria.
*/
public void setEskaeraLerroEgoera(String eskaeraLerroEgoera) {
```

```

 this.eskaeraLerroEgoera = eskaeraLerroEgoera;
 }

 // -----
 // KUDEAKETA METODOAK (STATIC)
 // -----

 /**
 * Eskaera lerroa sortu.
 *
 * @param el EskaeraLerroa objektua
 */
 /**
 * Eskaera lerroa sortu datu-basean.
 *
 * @param el EskaeraLerroa objektua.
 * @throws java.sql.SQLException Datu-basean errorea gertatzen bada.
 */
 public static void eskaeraLerroaSortu(EskaeraLerroa el) throws
java.sql.SQLException {
 String sql = "INSERT INTO eskaera_lerroak (eskaera_id, produktua_id,
kantitatea, unitate_preszia, eskaera_lerro_egoera) "
 +
 "VALUES (?, ?, ?, ?, ?)";
 try (java.sql.Connection konexioa = db.DB_Konexioa.konektatu();
 java.sql.PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, el.getEskaeraId());
 pst.setInt(2, el.getProduktuaId());
 pst.setInt(3, el.getKantitatea());
 pst.setBigDecimal(4, el.getUnitatePreszia());
 pst.setString(5, el.getEskaeraLerroEgoera());
 pst.executeUpdate();
 }
}

/**
 * Eskaera lerroa editatu.
 *
 * @param el EskaeraLerroa objektua (id-a barne)
 */
/**
 * Eskaera lerroa editatu datu-basean.
 *
 * @param el EskaeraLerroa objektua (id-a barne).
 * @throws java.sql.SQLException Datu-basean errorea gertatzen bada.
 */
 public static void eskaeralerroaEditatu(EskaeraLerroa el) throws
java.sql.SQLException {
 String sql = "UPDATE eskaera_lerroak SET eskaera_id=?, produktua_id=?,
kantitatea=?," +
 "unitate_preszia=?, eskaera_lerro_egoera=? WHERE
id_eskaera_lerroa=?";
 try (java.sql.Connection konexioa = db.DB_Konexioa.konektatu();
 java.sql.PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, el.getEskaeraId());
 pst.setInt(2, el.getProduktuaId());
 pst.setInt(3, el.getKantitatea());

```

```

 pst.setBigDecimal(4, el.getUnitatePrezioa());
 pst.setString(5, el.getEskaeraLerroEgoera());
 pst.setInt(6, el.getIdEskaeraLerroa());
 pst.executeUpdate();
 }
}

/**
 * Eskaera lerroa ezabatu.
 *
 * @param idEskaeraLerroa Eskaera lerroaren IDa
 */
/**
 * Eskaera lerroa ezabatu datu-basetik.
 *
 * @param idEskaeraLerroa Eskaera lerroaren IDa.
 * @throws java.sql.SQLException Datu-basean errorea gertatzen bada.
 */
public static void eskaeraLerroaEzabatu(int idEskaeraLerroa) throws
java.sql.SQLException {
 String sql = "DELETE FROM eskaera_lerroak WHERE id_eskaera_lerroa=?";
 try (java.sql.Connection konexioa = db.DB_Konexioa.konektatu()) {
 java.sql.PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idEskaeraLerroa);
 pst.executeUpdate();
 }
 }

 /**
 * Eskaera lerroen informazioa ikusi (Eskaera batenak).
 *
 * @param idEskaera Eskaeraren IDa
 * @return Lerro zerrenda
 */
 /**
 * Eskaera lerroen informazioa ikusi (Eskaera batenak).
 *
 * @param idEskaera Eskaeraren IDa.
 * @return Eskaera lerro zerrenda.
 * @throws java.sql.SQLException Datu-basean errorea gertatzen bada.
 */
 public static java.util.List<EskaeraLerroa> eskaeraLerroaIkusi(int idEskaera)
throws java.sql.SQLException {
 java.util.List<EskaeraLerroa> lerroak = new java.util.ArrayList<>();
 String sql = "SELECT * FROM eskaera_lerroak WHERE eskaera_id=?";
 try (java.sql.Connection konexioa = db.DB_Konexioa.konektatu()) {
 java.sql.PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idEskaera);
 try (java.sql.ResultSet rs = pst.executeQuery()) {
 while (rs.next()) {
 lerroak.add(new EskaeraLerroa(
 rs.getInt("id_eskaera_lerroa"),
 rs.getInt("eskaera_id"),
 rs.getInt("produktua_id"),
 rs.getInt("kantitatea"),
 rs.getBigDecimal("unitate_prezioa"),
 rs.getString("eskaera_lerro_egoera")));
 }
 }
 }
 }
}

```

```

 }
 }
 return lerroak;
}
}

```

# FakturaPDF.java

```

package model;

import com.itextpdf.text.BaseColor;
import com.itextpdf.text.Chunk;
import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.Element;
import com.itextpdf.text.Font;
import com.itextpdf.text.FontFactory;
import com.itextpdf.text.Image;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.Rectangle;
import com.itextpdf.text.pdf.PdfPCell;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.File;
import java.math.BigDecimal;
import java.util.List;
import java.io.FileInputStream;

import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import db.DB_Konexioa;

/**
 * FakturaPDF klasea.
 * Eskaera eta bezeroen datuetatik abiatuta PDF formatuko fakturak sortzen
 * dituen klasea.
 * iText liburutegia erabiltzen du PDFak sortzeko.
 */
public class FakturaPDF {

 /**
 * Faktura PDF bat sortzen du emandako datuekin (objektuen bidez).
 *

```

```

* @param fitxategiPath PDF fitxategia gordeko den bidea.
* @param eskaera Eskaera objektua.
* @param bezeroa Bezeroa objektua.
* @param lerroak Eskaera lerroen zerrenda.
* @throws DocumentException PDFa sortzean errorea gertatzen bada.
* @throws IOException Fitxategia idaztean errorea gertatzen bada.
*/
public static void sortu(String fitxategiPath, Eskaera eskaera, Bezeroa
bezeroa, List<EskaeraLerroa> lerroak)
 throws DocumentException, IOException {

 // Bezero datuak prestatu

 File fitxategia = new File(fitxategiPath);
 if (fitxategia.getParentFile() != null) {
 fitxategia.getParentFile().mkdirs();
 }

 Document document = new Document();
 PdfWriter.getInstance(document, new FileOutputStream(fitxategiPath));

 document.open();

 // Letra tipoak
 Font izenburuaFont = FontFactory.getFont(FontFactory.HELVETICA_BOLD, 18,
BaseColor.BLACK);
 Font arruntaFont = FontFactory.getFont(FontFactory.HELVETICA, 12,
BaseColor.BLACK);
 Font letraTxikia = FontFactory.getFont(FontFactory.HELVETICA, 10,
BaseColor.GRAY);
 Font taulaGoiburua = FontFactory.getFont(FontFactory.HELVETICA_BOLD, 12,
BaseColor.WHITE);

 // GOIBURUA: LOGOA + ENPRESA DATUAK (Taula batekin hobeto lerrokatzeke)
 PdfPTable goiburukoTaula = new PdfPTable(2);
 goiburukoTaula.setWidthPercentage(100);
 goiburukoTaula.setWidths(new float[] { 1, 1 });
 goiburukoTaula.getDefaultCell().setBorder(Rectangle.NO_BORDER);

 // LOGOA (Ezkerrean)
 try {
 String logoPath = "irudiak/birtek_logo_zuri_borobila.png";
 File logoFile = new File(logoPath);
 if (logoFile.exists()) {
 Image img = Image.getInstance(logoPath);
 img.scaleToFit(80, 80);
 PdfPCell logoCell = new PdfPCell(img);
 logoCell.setBorder(Rectangle.NO_BORDER);
 logoCell.setHorizontalAlignment(Element.ALIGN_LEFT);
 goiburukoTaula.addCell(logoCell);
 } else {
 // beste bide batetik saiatzen gara
 logoPath = "C:\\\\Ander\\\\Workspace\\\\Java\\\\eclipse-
workspace\\\\ERRONKA_2_BIRTEK_APP\\\\irudiak\\\\birtek_logo_zuri_borobila.png";
 if (new File(logoPath).exists()) {
 Image img = Image.getInstance(logoPath);
 img.scaleToFit(80, 80);
 }
 }
 }
}

```

```

 PdfPCell logoCell = new PdfPCell(img);
 logoCell.setBorder(Rectangle.NO_BORDER);
 logoCell.setHorizontalAlignment(Element.ALIGN_LEFT);
 goiburukoTaula.addCell(logoCell);
 } else {
 goiburukoTaula.addCell(""); // Hutsik
 }
}
} catch (Exception e) {
 goiburukoTaula.addCell(""); // Errorea egonez gero, hutsik
}

// ENPRESA DATUAK (Eskuinean)
Paragraph empresaInfo = new Paragraph(
 "Birtek S.L.\nKale Nagusia 123, Ordizia\nCampus Goierri
Campusa\nIFZ: B12345678\nTel: 944 123 456",
 letraTxikia);
PdfPCell infoCell = new PdfPCell(empresaInfo);
infoCell.setBorder(Rectangle.NO_BORDER);
infoCell.setHorizontalAlignment(Element.ALIGN_RIGHT);
goiburukoTaula.addCell(infoCell);

document.add(goiburukoTaula);

document.add(Chunk.NEWLINE);

// IZENBURUA
Paragraph izenburua = new Paragraph("FAKTURA: " + eskaera.getIdEskaera(),
izenburuaFont);
izenburua.setAlignment(Element.ALIGN_CENTER);
document.add(izenburua);

Paragraph dataPar = new Paragraph("Data: " + eskaera.getData().toString(),
arruntaFont);
dataPar.setAlignment(Element.ALIGN_CENTER);
document.add(dataPar);

document.add(Chunk.NEWLINE);

// BEZEROA
document.add(new Paragraph("Bezeroa:",
FontFactory.getFont(FontFactory.HELVETICA_BOLD, 12)));
document.add(new Paragraph(bezeroa.getIzenaEdoSoziala(), arruntaFont));
document.add(new Paragraph(bezeroa.getIfzNan(), arruntaFont));
document.add(new Paragraph(bezeroa.getHelbidea(), arruntaFont));
document.add(new Paragraph(bezeroa.getEmaila(), arruntaFont));

document.add(Chunk.NEWLINE);

// TAULA
PdfPTable table = new PdfPTable(4);
table.setWidthPercentage(100);
table.setWidths(new float[] { 4, 1, 2, 2 }); // Zutabeen zabalerak

// Taula Goiburuak
addHeaderCell(table, "Produktua", taulaGoiburua);
addHeaderCell(table, "Kop.", taulaGoiburua);

```

```

addHeaderCell(table, "Prezioa", taulaGoiburua);
addHeaderCell(table, "Guztira", taulaGoiburua);

// Datuak
BigDecimal calculatedTotal = BigDecimal.ZERO;
for (EskaeraLerroa lerroa : lerroak) {
 String izena = getProduktuaIzena(lerroa.getProduktuaId());
 BigDecimal guztiraLerroa = lerroa.getUnitatePrezioa().multiply(new
BigDecimal(lerroa.getKantitatea()));
 calculatedTotal = calculatedTotal.add(guztiraLerroa);

 table.addCell(new Phrase(izena, arruntaFont));
 table.addCell(new Phrase(String.valueOf(lerroa.getKantitatea()), arruntaFont));
 table.addCell(new Phrase(lerroa.getUnitatePrezioa() + " \u20AC", arruntaFont));
 table.addCell(new Phrase(guztiraLerroa + " \u20AC", arruntaFont));
}

document.add(table);

document.add(Chunk.NEWLINE);

// TOTALAK
PdfPTable totalTable = new PdfPTable(2);
totalTable.setWidthPercentage(40);
totalTable.setHorizontalAlignment(Element.ALIGN_RIGHT);

addTotalRow(totalTable, "GUZTIRA (BEZ %21 barne):",
eskaera.getGuztiraPrezioa() + " \u20AC",
FontFactory.getFont(FontFactory.HELVETICA_BOLD, 14));

document.add(totalTable);

document.close();
}

private static String getProduktuaIzena(int produktuaId) {
String izena = "Produktua " + produktuaId; // Ordezko izena
String sql = "SELECT izena FROM produktuak WHERE id_produktua = ?";
try (Connection kon = DB_Konexioa.konektatu();
PreparedStatement pst = kon.prepareStatement(sql)) {
pst.setInt(1, produktuaId);
try (ResultSet rs = pst.executeQuery()) {
if (rs.next()) {
izena = rs.getString("izena");
}
}
} catch (SQLException e) {
System.err.println("Errorea produktu izena lortzean: " +
e.getMessage());
}
return izena;
}

private static void addHeaderCell(PdfPTable table, String text, Font font) {
PdfPCell cell = new PdfPCell(new Phrase(text, font));

```

```

 cell.setBackgroundColor(BaseColor.GRAY);
 cell.setHorizontalAlignment(Element.ALIGN_CENTER);
 cell.setPadding(5);
 table.addCell(cell);
 }

 private static void addTotalRow(PdfPTable table, String label, String value,
Font font) {
 PdfPCell cellLabel = new PdfPCell(new Phrase(label, font));
 cellLabel.setBorder(Rectangle.NO_BORDER);
 cellLabel.setHorizontalAlignment(Element.ALIGN_RIGHT);
 table.addCell(cellLabel);

 PdfPCell cellValue = new PdfPCell(new Phrase(value, font));
 cellValue.setBorder(Rectangle.NO_BORDER);
 cellValue.setHorizontalAlignment(Element.ALIGN_RIGHT);
 table.addCell(cellValue);
}

/**
 * Faktura FTP bidez zerbitzarira igotzeko metodoa.
 *
 * @param fitxategiPath Igo nahi den fitxategiaren bide osoa.
 * @param fitxategiIzena Fitxategiak zerbitzarian izango duen izena.
 */
public static void fakturaIgoZerbitzarira(String fitxategiPath, String
fitxategiIzena) {
 String server = "localhost";
 int port = 21;
 String user = "root"; // FTP erabiltzailea
 String pass = "1MG32025"; // FTP pasahitza

 FTPClient ftpClient = new FTPClient();
 try {
 ftpClient.connect(server, port);
 ftpClient.login(user, pass);
 ftpClient.enterLocalPassiveMode();

 ftpClient.setFileType(FTP.BINARY_FILE_TYPE);

 // "htdocs/fakturak" karpetara joan
 // Aldatu direktorioa beharrezkoa bada.
 // XAMPP-en FileZilla-k askotan erroa zuzenean ezartzen du
erabiltzailearen
 // home gisa.
 // Hemen suposatzen dugu "htdocs/fakturak" existitzen dela edo sortu
behar dela.
 // Bainaz erabiltzaileak esan du "htdocs/fakturak" karpetan gorde nahi
duela.

 // Saiatu direktorioa aldatzen, bestela sortu
 if (!ftpClient.changeWorkingDirectory("htdocs/fakturak")) {
 if (ftpClient.makeDirectory("htdocs")) {
 ftpClient.changeWorkingDirectory("htdocs");
 ftpClient.makeDirectory("fakturak");
 ftpClient.changeWorkingDirectory("fakturak");
 } else {

```

```
// Agian zuzenean fakturak karpeta erroan dago edo beste
egitura bat du
 ftpClient.makeDirectory("fakturak");
 ftpClient.changeWorkingDirectory("fakturak");
}
}

File firstLocalFile = new File(fitxategiPath);

String firstRemoteFile = fitxategiIzena;
FileInputStream inputStream = new FileInputStream(firstLocalFile);

System.out.println("Fitxategia igotzen hasten...");
boolean done = ftpClient.storeFile(firstRemoteFile, inputStream);
inputStream.close();
if (done) {
 System.out.println("Fitxategia ondo igo da.");
} else {
 System.out.println("Errorea fitxategia igotzean.");
}

} catch (IOException ex) {
 System.out.println("Errorea: " + ex.getMessage());
 ex.printStackTrace();
} finally {
 try {
 if (ftpClient.isConnected()) {
 ftpClient.logout();
 ftpClient.disconnect();
 }
 } catch (IOException ex) {
 ex.printStackTrace();
 }
}
}
}
```

# Fitxaketa.java

```
package model;

import java.sql.Date;
import java.sql.Time;
import java.sql.Timestamp;

/**
 * Fitxaketa klasea.
 * Langileen sarrera eta irteera fitxaketak kudeatzen ditu.
 * Fitxaketaren data, ordua, mota eta langilea gordetzen ditu.
 */
public class Fitxaketa {
```

```
private int idFitxaketa;
private int langileaId;
private Date data;
private Time ordua;
private String mota;
private Timestamp eguneratzeData;

/**
 * Fitxaketa eraikitzalea.
 *
 * @param idFitxaketa Fitxaketaren IDa.
 * @param langileaId Langilearen IDa.
 * @param data Data.
 * @param ordua Ordua.
 * @param mota Mota (Sarrera/Irteera).
 * @param eguneratzeData Eguneratze data.
 */
public Fitxaketa(int idFitxaketa, int langileaId, Date data, Time ordua, String mota, Timestamp eguneratzeData) {
 this.idFitxaketa = idFitxaketa;
 this.langileaId = langileaId;
 this.data = data;
 this.ordua = ordua;
 this.mota = mota;
 this.eguneratzeData = eguneratzeData;
}

/**
 * Fitxaketaren IDa lortzen du.
 *
 * @return IDa.
 */
public int getIdFitxaketa() {
 return idFitxaketa;
}

public void setIdFitxaketa(int idFitxaketa) {
 this.idFitxaketa = idFitxaketa;
}

/**
 * Langilearen IDa lortzen du.
 *
 * @return Langilearen IDa.
 */
public int getLangileaId() {
 return langileaId;
}

public void setLangileaId(int langileaId) {
 this.langileaId = langileaId;
}

/**
 * Data lortzen du.
 *
 * @return Data.
*/
```

```

/*
public Date getData() {
 return data;
}

public void setData(Date data) {
 this.data = data;
}

/**
 * Ordua lortzen du.
 *
 * @return Ordua.
 */
public Time getOrdua() {
 return ordua;
}

public void setOrdua(Time ordua) {
 this.ordua = ordua;
}

/**
 * Fitxaketa mota lortzen du.
 *
 * @return Mota.
 */
public String getMota() {
 return mota;
}

public void setMota(String mota) {
 this.mota = mota;
}

/**
 * Eguneratze data lortzen du.
 *
 * @return Eguneratze data.
 */
public Timestamp getEguneratzeData() {
 return eguneratzeData;
}

public void setEguneratzeData(Timestamp eguneratzeData) {
 this.eguneratzeData = eguneratzeData;
}
}

```

# Herria.java

```
package model;

/**
 * Herria klasea.
 * Herrien informazioa kudeatzeko klasea.
 * Herriaren izena, lurrealdea eta nazioa gordetzen ditu.
 */
public class Herria {
 private int idHerria;
 private String izena;
 private String lurrealdea;
 private String nazioa;

 /**
 * Herria eraikitzalea.
 *
 * @param idHerria Herriaren IDa.
 * @param izena Herriaren izena.
 * @param lurrealdea Probintzia edo lurrealdea.
 * @param nazioa Nazioa edo herrialdea.
 */
 public Herria(int idHerria, String izena, String lurrealdea, String nazioa) {
 this.idHerria = idHerria;
 this.izena = izena;
 this.lurrealdea = lurrealdea;
 this.nazioa = nazioa;
 }

 /**
 * Herriaren IDa lortzen du.
 *
 * @return IDa.
 */
 public int getIdHerria() {
 return idHerria;
 }

 /**
 * Herriaren IDa ezartzen du.
 *
 * @param idHerria ID berria.
 */
 public void setIdHerria(int idHerria) {
 this.idHerria = idHerria;
 }

 /**
 * Herriaren izena lortzen du.
 *
 * @return Izena.
 */
 public String getIzena() {
 return izena;
 }

 /**

```

```
* Herriaren izena ezartzen du.
*
* @param izena Izen berria.
*/
public void setIzena(String izena) {
 this.izena = izena;
}

/**
 * Lurraldea lortzen du.
 *
 * @return Lurraldea.
*/
public String getLurraldea() {
 return lurraldea;
}

/**
 * Lurraldea ezartzen du.
 *
 * @param lurraldea Lurralde berria.
*/
public void setLurraldea(String lurraldea) {
 this.lurraldea = lurraldea;
}

/**
 * Nazioa lortzen du.
 *
 * @return Nazioa.
*/
public String getNazioa() {
 return nazioa;
}

/**
 * Nazioa ezartzen du.
 *
 * @param nazioa Nazio berria.
*/
public void setNazioa(String nazioa) {
 this.nazioa = nazioa;
}

/**
 * Herriaren izena itzultzen du String bezala.
 *
 * @return Herriaren izena.
*/
@Override
public String toString() {
 return izena;
}
}
```

# Hornitzalea.java

```
package model;

import java.sql.Timestamp;

/**
 * Hornitzalea klasea.
 * Pertsona klasaren azpiklasea da, eta hornitzaleen informazioa gordetzen du.
 * Kontakturako pertsonaren izena kudeatzen du.
 */
public class Hornitzalea extends Pertsona {
 private String kontaktuPertsona;

 /**
 * Hornitzalea eraikitzailea.
 *
 * @param idHornitzalea Hornitzalearen IDa.
 * @param izenaSoziala Izen soziala.
 * @param ifzNan IFZ edo NAN.
 * @param kontaktuPertsona Kontaktu pertsona.
 * @param helbidea Helbidea.
 * @param herriaId Herriaren IDa.
 * @param postaKodea Posta kodea.
 * @param telefonoa Telefonoa.
 * @param emaila Emaila.
 * @param hizkuntza Hizkuntza.
 * @param pasahitza Pasahitza.
 * @param aktibo Aktibo dagoen.
 * @param eguneraszteData Eguneraszte data.
 */
 public Hornitzalea(int idHornitzalea, String izenaSoziala, String ifzNan,
String kontaktuPertsona,
 String helbidea, int herriaId, String postaKodea, String telefonoa,
String emaila,
 String hizkuntza, String pasahitza, boolean aktibo, Timestamp
eguneraszteData) {
 super(idHornitzalea, izenaSoziala, null, ifzNan, null, helbidea, herriaId,
postaKodea, telefonoa, emaila,
 hizkuntza, pasahitza, aktibo, null, eguneraszteData);
 this.kontaktuPertsona = kontaktuPertsona;
 }

 /**
 * Hornitzalearen IDa lortzen du.
 *
 * @return IDa.
 */
 public int getIdHornitzalea() {
 return this.id;
 }

 /**
 * Hornitzalearen IDa ezartzen du.
 */
```

```
*
* @param idHornitzalea ID berria.
*/
public void setIdHornitzalea(int idHornitzalea) {
 this.id = idHornitzalea;
}

/**
 * Izen soziala lortzen du.
 *
 * @return Izen soziala.
*/
public String getIzenaSoziala() {
 return this.izena;
}

/**
 * Izen soziala ezartzen du.
 *
 * @param izenaSoziala Izen sozial berria.
*/
public void setIzenaSoziala(String izenaSoziala) {
 this.izena = izenaSoziala;
}

/**
 * IFZ edo NAN lortzen du.
 *
 * @return IFZ edo NAN.
*/
public String getIfzNan() {
 return this.nanIfz;
}

/**
 * IFZ edo NAN ezartzen du.
 *
 * @param ifzNan IFZ edo NAN berria.
*/
public void setIfzNan(String ifzNan) {
 this.nanIfz = ifzNan;
}

/**
 * Kontaktu pertsona lortzen du.
 *
 * @return Kontaktu pertsona.
*/
public String getKontaktuPertsona() {
 return kontaktuPertsona;
}

/**
 * Kontaktu pertsona ezartzen du.
 *
 * @param kontaktuPertsona Kontaktu pertsona berria.
*/
```

```
public void setKontaktuPertsona(String kontaktuPertsona) {
 this.kontaktuPertsona = kontaktuPertsona;
}
}
```

# Konponketa.java

```
package model;

import java.sql.Timestamp;

/**
 * Konponketa klasea.
 * Produktuen konponketen informazioa kudeatzen du.
 * Produktua, arduraduna (langilea), datak eta egoera gordetzen ditu.
 */
public class Konponketa {
 private int idKonponketa;
 private int produktuaId;
 private int langileaId;
 private Timestamp hasieraData;
 private Timestamp amaieraData;
 private String konponketaEgoera;
 private int akatsaId;
 private String oharrak;
 private Timestamp egunearatzeData;

 /**
 * Konponketa eraikitzalea.
 *
 * @param idKonponketa Konponketaren IDa.
 * @param produktuaId Produktuaren IDa.
 * @param langileaId Arduradunaren (langilea) IDa.
 * @param hasieraData Hasiera data.
 * @param amaieraData Amaiera data.
 * @param konponketaEgoera Konponketaren egoera.
 * @param akatsaId Akatsaren IDa.
 * @param oharrak Oharrak.
 * @param egunearatzeData Egunearatze data.
 */
 public Konponketa(int idKonponketa, int produktuaId, int langileaId, Timestamp
hasieraData, Timestamp amaieraData,
 String konponketaEgoera, int akatsaId, String oharrak, Timestamp
egunearatzeData) {
 this.idKonponketa = idKonponketa;
 this.produktuaId = produktuaId;
 this.langileaId = langileaId;
 this.hasieraData = hasieraData;
 this.amaieraData = amaieraData;
 this.konponketaEgoera = konponketaEgoera;
```

```
 this.akatsaId = akatsaId;
 this.oharrak = oharrak;
 this.eguneratzeData = eguneratzeData;
 }

 /**
 * Konponketaren IDa lortzen du.
 *
 * @return IDa.
 */
 public int getIdKonponketa() {
 return idKonponketa;
 }

 public void setIdKonponketa(int idKonponketa) {
 this.idKonponketa = idKonponketa;
 }

 /**
 * Produktuaren IDa lortzen du.
 *
 * @return Produktuaren IDa.
 */
 public int getProduktuaId() {
 return produktuaId;
 }

 public void setProduktuaId(int produktuaId) {
 this.produktuaId = produktuaId;
 }

 /**
 * Langilearen IDa lortzen du.
 *
 * @return Langilearen IDa.
 */
 public int getLangileaId() {
 return langileaId;
 }

 public void setLangileaId(int langileaId) {
 this.langileaId = langileaId;
 }

 /**
 * Hasiera data lortzen du.
 *
 * @return Hasiera data.
 */
 public Timestamp getHasieraData() {
 return hasieraData;
 }

 public void setHasieraData(Timestamp hasieraData) {
 this.hasieraData = hasieraData;
 }
```

```
/**
 * Amaiera data lortzen du.
 *
 * @return Amaiera data.
 */
public Timestamp getAmaieraData() {
 return amaieraData;
}

public void setAmaieraData(Timestamp amaieraData) {
 this.amaieraData = amaieraData;
}

/**
 * Konponketaren egoera lortzen du.
 *
 * @return Egoera.
 */
public String getKonponketaEgoera() {
 return konponketaEgoera;
}

public void setKonponketaEgoera(String konponketaEgoera) {
 this.konponketaEgoera = konponketaEgoera;
}

/**
 * Akatsaren IDa lortzen du.
 *
 * @return Akatsaren IDa.
 */
public int getAkatsaId() {
 return akatsaId;
}

public void setAkatsaId(int akatsaId) {
 this.akatsaId = akatsaId;
}

/**
 * Oharrak lortzen ditu.
 *
 * @return Oharrak.
 */
public String getOharrak() {
 return oharrak;
}

public void setOharrak(String oharrak) {
 this.oharrak = oharrak;
}

/**
 * Eguneratze data lortzen du.
 *
 * @return Eguneratze data.
 */
```

```

public Timestamp getEguneratzeData() {
 return eguneratzeData;
}

public void setEguneratzeData(Timestamp eguneratzeData) {
 this.eguneratzeData = eguneratzeData;
}
}

```

## Langilea.java

```

package model;

import db.DB_Konexioa;

import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;

/**
 * Langilea klasea.
 * Pertsona klasaren azpiklasea da, eta langileen informazio espezifikoak
 * kudeatzen ditu.
 * Fitxaketak egiteko, bere datuak ikusteko eta editatzeko metodoak ditu.
 */
public class Langilea extends Pertsona {
 private String iban;
 private byte[] kurrikuluma; // PDF formatuan
 private int sailaId;
 private String saltoTxartelaUid;

 /**
 * Langilea eraikitzailea.
 *
 * @param idLangilea Langilearen IDa.
 * @param izena Izena.
 * @param abizena Abizena.
 * @param nan NAN zenbakia.
 * @param jaiotzaData Jaiotza data.
 * @param herriaId Herriaren IDa.
 * @param helbidea Helbidea.
 * @param postaKodea Posta kodea.
 * @param telefonoa Telefonoa.
 * @param emaila Emaila.
 * @param hizkuntza Hizkuntza.
 * @param pasahitza Pasahitza.
 */
}

```

```

* @param saltoTxartelaUid Salto txartelaren UIDa.
* @param altaData Alta data.
* @param eguneratzeData Eguneratze data.
* @param aktibo Aktibo dagoen.
* @param sailaId Sailaren IDa.
* @param iban IBAN kontu korrontea.
* @param kurrikulum Kurrikulum PDF formatuan (byte array).
*/
public Langilea(int idLangilea, String izena, String abizena, String nan, Date
jaiotzaData, int herriaId,
String helbidea, String postaKodea, String telefonoa, String emaila,
String hizkuntza,
String pasahitza, String saltoTxartelaUid, Timestamp altaData,
Timestamp eguneratzeData,
boolean aktibo, int sailaId, String iban, byte[] kurrikulum) {
super(idLangilea, izena, abizena, nan, jaiotzaData, helbidea, herriaId,
postaKodea, telefonoa, emaila,
hizkuntza, pasahitza, aktibo, altaData, eguneratzeData);
this.sailaId = sailaId;
this.iban = iban;
this.kurrikulum = kurrikulum;
}

/**
* Langilearen IDa lortzen du.
*
* @return IDa.
*/
public int getIdLangilea() {
 return this.id;
}

/**
* Langilearen IDa ezartzen du.
*
* @param idLangilea ID berria.
*/
public void setIdLangilea(int idLangilea) {
 this.id = idLangilea;
}

/**
* Langilearen NANa lortzen du.
*
* @return NAN zenbakia.
*/
public String getNan() {
 return this.nanIfz;
}

/**
* Langilearen NANa ezartzen du.
*
* @param nan NAN berria.
*/
public void setNan(String nan) {
 this.nanIfz = nan;
}

```

```
}

/**
 * Salto txartelaren UIDa lortzen du.
 *
 * @return UIDa.
 */
public String getSaltoTxartelaUid() {
 return saltoTxartelaUid;
}

/**
 * Salto txartelaren UIDa ezartzen du.
 *
 * @param saltoTxartelaUid UID berria.
 */
public void setSaltoTxartelaUid(String saltoTxartelaUid) {
 this.saltoTxartelaUid = saltoTxartelaUid;
}

/**
 * Sailaren IDa lortzen du.
 *
 * @return Sailaren IDa.
 */
public int getSailaId() {
 return sailaId;
}

/**
 * Sailaren IDa ezartzen du.
 *
 * @param sailaId Sailaren ID berria.
 */
public void setSailaId(int sailaId) {
 this.sailaId = sailaId;
}

/**
 * IBAN kontu korrontea lortzen du.
 *
 * @return IBANA.
 */
public String getIban() {
 return iban;
}

/**
 * IBAN kontu korrontea ezartzen du.
 *
 * @param iban IBAN berria.
 */
public void setIban(String iban) {
 this.iban = iban;
}

/**
```

```

* Kurrikuluma lortzen du.
*
* @return Kurrikuluma byte array bezala.
*/
public byte[] getKurrikuluma() {
 return kurrikuluma;
}

/**
* Kurrikuluma ezartzen du.
*
* @param kurrikuluma Byte array berria.
*/
public void setKurrikuluma(byte[] kurrikuluma) {
 this.kurrikuluma = kurrikuluma;
}

// --- FITXAKETA LOGIKA (OOP) ---

// Langile batentzat fitxaketa egiteko metodoa
/**
* Fitxaketa (Sarrera edo Irteera) bat erregistratzen du.
* Lehenik egiaztatzen du ea langilea barruan edo kanpoan dagoen, egoera
* koherentea izateko.
*
* @param fitxaketa_mota "Sarrera" edo "Irteera".
* @throws java.sql.SQLException Datu-basean errorea gertatzen bada edo egoera
* ez bada zuzena (adib. bi aldiz sartzen
* saiatzea).
*/
public void fitxatu(String fitxaketa_mota) throws java.sql.SQLException {
 String galdera = "SELECT mota FROM fitxaketak WHERE langilea_id = ? ORDER
BY id_fitzaketa DESC LIMIT 1";
 try (java.sql.Connection konexioa = DB_Konexioa.konektatu()) {
 boolean barruan = false;
 try (java.sql.PreparedStatement sententzia =
konexioa.prepareStatement(galdera)) {
 sententzia.setInt(1, this.getIdLangilea());
 try (java.sql.ResultSet rs = sententzia.executeQuery()) {
 String azkenMota = null;
 if (rs.next()) {
 azkenMota = rs.getString("mota");
 }

 // Ezarri barruan aldagai
 if ("Sarrera".equals(azkenMota)) {
 barruan = true;
 } else {
 barruan = false;
 }

 // Balidazioak barruan aldagai erabiliz
 if ("Sarrera".equals(fitxaketa_mota) && barruan) {
 throw new java.sql.SQLException("Jada barruan zaude.");
 }
 if ("Irteera".equals(fitxaketa_mota) && !barruan) {
 throw new java.sql.SQLException("Jada kanpoan zaude. Ezin

```

```
duzu irten sartu gabe.");
 }
 }
}

String sartuGaldera = "INSERT INTO fitxaketak (langilea_id, mota, data,
ordua) VALUES (?, ?, CURRENT_DATE, CURRENT_TIME)";
try (java.sql.PreparedStatement pstInsert =
konexioa.prepareStatement(sartuGaldera)) {
 pstInsert.setInt(1, this.getIdLangilea());
 pstInsert.setString(2, fitxaketa_mota);
 pstInsert.executeUpdate();
}
}

/**
 * Sarrera fitxaketa bat egiten du.
 *
 * @throws java.sql.SQLException Errorea gertatzen bada.
 */
public void sarreraFitzaketaEgin() throws java.sql.SQLException {
 fitxatu("Sarrera");
}

/**
 * Irteera fitxaketa bat egiten du.
 *
 * @throws java.sql.SQLException Errorea gertatzen bada.
 */
public void irteeraFitzaketaEgin() throws java.sql.SQLException {
 fitxatu("Irteera");
}

// langile batek bere fitxaketa historiala ikusteko metodoa (datuak itzultzen
// ditu)
/**
 * Langilearen fitxaketa historiala lortzen du.
 *
 * @return Ftxaketa zerrenda bat.
 */
public java.util.List<Fitzaketa> nireFitzaketakIkusi() {
 java.util.List<Fitzaketa> zerrenda = new java.util.ArrayList<>();
 String galdera = "SELECT id_fitzaketa, langilea_id, data, ordua, mota,
eguneratze_data FROM fitxaketak WHERE langilea_id = ? ORDER BY id_fitzaketa DESC";
 try (java.sql.Connection konexioa = DB_Konexioa.konektatu();
 java.sql.PreparedStatement sententzia =
konexioa.prepareStatement(galdera)) {
 sententzia.setInt(1, this.getIdLangilea());
 java.sql.ResultSet rs = sententzia.executeQuery();
 while (rs.next()) {
 zerrenda.add(new Fitzaketa(
 rs.getInt("id_fitzaketa"),
 rs.getInt("langilea_id"),
 rs.getDate("data"),
 rs.getTime("ordua"),
 rs.getString("mota"),
 rs.getTimestamp("eguneratze_data")));
 }
 }
}
```

```

 }
 } catch (java.sql.SQLException e) {
 e.printStackTrace();
 }
 return zerrenda;
}

/**
 * Herrien zerrenda lortzen du.
 *
 * @return Herria objektuen zerrenda.
 */
public List<Herria> herriakLortu() {
 List<Herria> zerrenda = new ArrayList<>();
 String sql = "SELECT * FROM herriak ORDER BY izena";
 try (java.sql.Connection konexioa = DB_Konexioa.konektatu();
 Statement st = konexioa.createStatement();
 ResultSet rs = st.executeQuery(sql)) {
 while (rs.next()) {
 zerrenda.add(new Herria(
 rs.getInt("id_herria"),
 rs.getString("izena"),
 rs.getString("lurraldea"),
 rs.getString("nazioa")));
 }
 } catch (Exception e) {
 e.printStackTrace();
 }
 return zerrenda;
}

/**
 * Herri berri bat sortzen du datu-basean.
 *
 * @param h Herria objektua.
 * @throws SQLException Datu-basean errorea gertatzen bada.
 */
public void herriaSortu(Herria h) throws SQLException {
 String sql = "INSERT INTO herriak (izena, lurraldea, nazioa) VALUES (?, ?, ?)";
 try (java.sql.Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setString(1, h.getIzena());
 pst.setString(2, h.getLurraldea());
 pst.setString(3, h.getNazioa());
 pst.executeUpdate();
 }
}

/**
 * Langilearen oinarrizko datuak editatzen ditu (perfil aldaketa).
 *
 * @param pasahitza Pasahitz berria.
 * @param hizkuntza Hizkuntza berria.
 * @param herriaId Herriaren ID berria.
 * @param telefonoa Telefono berria.
 * @param helbidea Helbide berria.
 */

```

```

* @throws java.sql.SQLException Datu-basean errorea gertatzen bada.
*/
public void nireLangileDatuakEditatu(String pasahitza, String hizkuntza, int
herriaId, String telefonoa,
String helbidea)
throws java.sql.SQLException {
String sql = "UPDATE langileak SET pasahitza = ?, hizkuntza = ?, herria_id
= ?, telefonoa = ?, helbidea = ?, eguneratze_data = NOW() WHERE id_langilea = ?";
try (java.sql.Connection konexioa = DB_Konexioa.konektatu();
java.sql.PreparedStatement sententzia =
konexioa.prepareStatement(sql)) {
sententzia.setString(1, pasahitza);
sententzia.setString(2, hizkuntza);
sententzia.setInt(3, herriaId);
sententzia.setString(4, telefonoa);
sententzia.setString(5, helbidea);
sententzia.setInt(6, this.getIdLangilea());
sententzia.executeUpdate();

// Objektuaren datuak eguneratu
this.setPasahitza(pasahitza);
this.setHizkuntza(hizkuntza);
this.setHerriaId(herriaId);
this.setTelefonoa(telefonoa);
this.setHelbidea(helbidea);
}
}

// langile batek bere fitxaketak historiala ikusteko metodoa:
// langile batek bere fitxaketak historiala ikusteko metodoa:
/***
 * Langilearen azken fitxaketa egoera lortzen du.
*
* @return Testu deskribatzalea ("BARRUAN" edo "KANPOAN" eta ordua).
*/
public String getFitxaketaEgoera() {
String galdera = "SELECT mota, data, ordua FROM fitxaketak WHERE
langilea_id = ? ORDER BY id_fitxaketa DESC LIMIT 1";
try (java.sql.Connection konexioa = DB_Konexioa.konektatu();
java.sql.PreparedStatement sententzia =
konexioa.prepareStatement(galdera)) {
sententzia.setInt(1, this.getIdLangilea());
java.sql.ResultSet rs = sententzia.executeQuery();
if (rs.next()) {
String mota = rs.getString("mota");
java.sql.Date data = rs.getDate("data");
java.sql.Time ordua = rs.getTime("ordua");
if ("Sarrera".equals(mota)) {
return " BARRUAN (" + ordua + ") Data: (" + data + ")";
} else {
return " KANPOAN (" + ordua + ") Data: (" + data + ")";
}
} else {
return "Ez dago erregistrorik.";
}
} catch (java.sql.SQLException e) {
e.printStackTrace();
}
}

```

```

 return "Errorea egoera lortzean";
 }
}

}

```

# LangileSaila.java

```

package model;

/**
 * LangileSaila klasea.
 * Langileen sailen informazioa kudeatzeko klasea.
 * Sailaren IDa, izena, kokapena eta deskribapena gordetzen ditu.
 */
public class LangileSaila {
 private int idSaila;
 private String izena;
 private String kokapena;
 private String deskribapena;

 /**
 * LangileSaila eraikitzalea.
 *
 * @param idSaila Sailaren IDa.
 * @param izena Sailaren izena.
 * @param kokapena Sailaren kokapena.
 * @param deskribapena Sailaren deskribapena.
 */
 public LangileSaila(int idSaila, String izena, String kokapena, String
deskribapena) {
 this.idSaila = idSaila;
 this.izena = izena;
 this.kokapena = kokapena;
 this.deskribapena = deskribapena;
 }

 /**
 * Sailaren IDa lortzen du.
 *
 * @return IDa.
 */
 public int getIdSaila() {
 return idSaila;
 }

 public void setIdSaila(int idSaila) {
 this.idSaila = idSaila;
 }

 /**

```

```

* Sailaren izena lortzen du.
*
* @return Izena.
*/
public String getIzena() {
 return izena;
}

public void setIzena(String izena) {
 this.izena = izena;
}

/***
 * Sailaren kokapena lortzen du.
*
* @return Kokapena.
*/
public String getKokapena() {
 return kokapena;
}

public void setKokapena(String kokapena) {
 this.kokapena = kokapena;
}

/***
 * Sailaren deskribapena lortzen du.
*
* @return Deskribapena.
*/
public String getDeskribapena() {
 return deskribapena;
}

public void setDeskribapena(String deskribapena) {
 this.deskribapena = deskribapena;
}
}

```

## MahaiGainekoa.java

```

package model;

import java.math.BigDecimal;
import java.sql.Timestamp;

/***
 * MahaiGainekoa klasea.
 * Produktua klasaren azpiklasea da, eta mahai gaineko ordenagailuen
 * informazioa kudeatzen du.
 * Prozesadorea, RAM, diskoa, txartel grafikoa, elikatze iturria eta kaxa

```

```

* formatua bezalako ezaugarriak gordetzen ditu.
*/
public class MahaiGainekoa extends Produktua {
 private String prozesadorea;
 private String plakaBasea;
 private int ramGb;
 private int diskoaGb;
 private String txartelGrafikoa;
 private int elikatzeIturriaW;
 private String kaxaFormatua;

 /**
 * MahaiGainekoa eraikitzailea.
 *
 * @param idProduktua Produktuaren IDa.
 * @param hornitzaireId Hornitzairearen IDa.
 * @param kategoriaId Kategoriaren IDa.
 * @param izena Izena.
 * @param marka Marka.
 * @param mota Mota (MahaiGainekoa).
 * @param deskribapena Deskribapena.
 * @param irudiaUrl Irudiaren URLa.
 * @param biltegiId Biltegiaren IDa.
 * @param produktuEgoera Produktuaren egoera.
 * @param produktuEgoeraOharra Egoeraren oharra.
 * @param salgai Salgai dagoen.
 * @param salmentaPrezioa Salmenta prezioa.
 * @param stock Stock kopurua.
 * @param eskaintza Eskaintza prezioa.
 * @param zergakEhunekoa Zergen ehunekoa.
 * @param sortzeData Sortze data.
 * @param eguneratzeData Eguneratze data.
 * @param prozesadorearea Prozesadorearea.
 * @param plakaBasea Plaka basea.
 * @param ramGb RAM memoria GBtan.
 * @param diskoaGb Diskoaren kapazitatea GBtan.
 * @param txartelGrafikoa Txartel grafikoa.
 * @param elikatzeIturriaW Elikatze iturria W-tan.
 * @param kaxaFormatua Kaxaren formatua.
 */
 public MahaiGainekoa(int idProduktua, int hornitzaireId, int kategoriaId,
String izena, String marka, String mota,
 String deskribapena, String irudiaUrl, Integer biltegiId, String
produktuEgoera,
 String produktuEgoeraOharra, boolean salgai, BigDecimal
salmentaPrezioa, int stock,
 BigDecimal eskaintza, BigDecimal zergakEhunekoa, Timestamp sortzeData,
Timestamp eguneratzeData,
 String prozesadorearea, String plakaBasea, int ramGb, int diskoaGb,
 String txartelGrafikoa, int elikatzeIturriaW, String kaxaFormatua) {
 super(idProduktua, hornitzaireId, kategoriaId, izena, marka, mota,
 deskribapena, irudiaUrl, biltegiId,
 produktuEgoera, produktuEgoeraOharra, salgai, salmentaPrezioa,
 stock, eskaintza, zergakEhunekoa,
 sortzeData, eguneratzeData);
 this.prozesadorearea = prozesadorearea;
 this.plakaBasea = plakaBasea;
 }
}

```

```
 this.ramGb = ramGb;
 this.diskoGb = diskGb;
 this.txartelGrafikoa = txartelGrafikoa;
 this.elikatzeIturriaW = elikatzeIturriaW;
 this.kaxaFormatua = kaxaFormatua;
}

/**
 * Prozesadorea lortzen du.
 *
 * @return Prozesadorea.
 */
public String getProzesadorea() {
 return prozesadorea;
}

/**
 * Prozesadorea ezartzen du.
 *
 * @param prozesadorea Prozesadore berria.
 */
public void setProzesadorea(String prozesadorea) {
 this.prozesadorea = prozesadorea;
}

/**
 * Plaka basea lortzen du.
 *
 * @return Plaka basea.
 */
public String getPlakaBasea() {
 return plakaBasea;
}

/**
 * Plaka basea ezartzen du.
 *
 * @param plakaBasea Plaka base berria.
 */
public void setPlakaBasea(String plakaBasea) {
 this.plakaBasea = plakaBasea;
}

/**
 * RAM memoria lortzen du.
 *
 * @return RAM memoria GBtan.
 */
public int getRamGb() {
 return ramGb;
}

/**
 * RAM memoria ezartzen du.
 *
 * @param ramGb RAM memoria berria.
 */

```

```
public void setRamGb(int ramGb) {
 this.ramGb = ramGb;
}

/**
 * Diskoaren kapazitatea lortzen du.
 *
 * @return Diskoa GBtan.
 */
public int getDiskoaGb() {
 return diskoaGb;
}

/**
 * Diskoaren kapazitatea ezartzen du.
 *
 * @param diskoaGb Disko kapazitate berria.
 */
public void setDiskoaGb(int diskoaGb) {
 this.diskoaGb = diskoaGb;
}

/**
 * Txartel grafikoa lortzen du.
 *
 * @return Txartel grafikoa.
 */
public String getTxartelGrafikoa() {
 return txartelGrafikoa;
}

/**
 * Txartel grafikoa ezartzen du.
 *
 * @param txartelGrafikoa Txartel grafiko berria.
 */
public void setTxartelGrafikoa(String txartelGrafikoa) {
 this.txartelGrafikoa = txartelGrafikoa;
}

/**
 * Elikatze iturria lortzen du.
 *
 * @return Elikatze iturria W-tan.
 */
public int getElikatzeIturriaW() {
 return elikatzeIturriaW;
}

/**
 * Elikatze iturria ezartzen du.
 *
 * @param elikatzeIturriaW Elikatze iturri berria.
 */
public void setElikatzeIturriaW(int elikatzeIturriaW) {
 this.elikatzeIturriaW = elikatzeIturriaW;
}
```

```

 /**
 * Kaxa formatua lortzen du.
 *
 * @return Kaxa formatua.
 */
 public String getKaxaFormatua() {
 return kaxaFormatua;
 }

 /**
 * Kaxa formatua ezartzen du.
 *
 * @param kaxaFormatua Kaxa formatu berria.
 */
 public void setKaxaFormatua(String kaxaFormatua) {
 this.kaxaFormatua = kaxaFormatua;
 }
}

```

## Mugikorra.java

```

package model;

import java.math.BigDecimal;
import java.sql.Timestamp;

/**
 * Mugikorra klasea.
 * Produktua klasearen azpiklasea da, eta telefono mugikorren informazioa
 * kudeatzen du.
 * Pantaila, biltegiratzea, RAM, kamera, bateria eta sistema eragilea bezalako
 * ezaugarriak gordetzen ditu.
 */
public class Mugikorra extends Produktua {
 private String pantailaTeknologia;
 private BigDecimal pantailaHazbeteak;
 private int biltegiratzeaGb;
 private int ramGb;
 private int kameraNagusaMp;
 private int bateriaMah;
 private String sistemaEragilea;
 private String sareak;

 /**
 * Mugikorra eraikitzailea.
 *
 * @param idProduktua Produktuaren IDa.
 * @param hornitztaileId Hornitztailearen IDa.
 * @param kategoriaId Kategoriaren IDa.
 * @param izena Izena.
 */
}

```

```

* @param marka Marka.
* @param mota Mota (Mugikorra).
* @param deskribapena Deskribapena.
* @param irudiaUrl Irudiaren URLa.
* @param biltegiId Biltegiaren IDa.
* @param produktuEgoera Produktuaren egoera.
* @param produktuEgoeraOharra Egoeraren oharra.
* @param salgai Salgai dagoen.
* @param salmentaPrezioa Salmenta prezioa.
* @param stock Stock kopurua.
* @param eskaintza Eskaintza prezioa.
* @param zergakEhunekoa Zergen ehunekoa.
* @param sortzeData Sortze data.
* @param egeneratzeData Egeneratze data.
* @param pantailaTeknologia Pantailaren teknologia.
* @param pantailaHazbeteak Pantailaren tamaina hazbetetan.
* @param biltegiratzeGb Biltegiratzea GBtan.
* @param ramGb RAM memoria GBtan.
* @param kameraNagusaMp Kamera nagusia MPtan.
* @param bateriaMah Bateriaren kapazitatea mAh-tan.
* @param sistemaEragilea Sistema eragilea.
* @param sareak Sarea (4G, 5G...).

*/
public Mugikorra(int idProduktua, int hornitzaireId, int kategoriaId, String
izen, String marka, String mota,
 String deskribapena, String irudiaUrl, Integer biltegiId, String
produktuEgoera,
 String produktuEgoeraOharra, boolean salgai, BigDecimal
salmentaPrezioa, int stock,
 BigDecimal eskaintza, BigDecimal zergakEhunekoa, Timestamp sortzeData,
Timestamp egeneratzeData,
 String pantailaTeknologia, BigDecimal pantailaHazbeteak, int
biltegiratzeGb,
 int ramGb, int kameraNagusaMp, int bateriaMah, String sistemaEragilea,
String sareak) {
 super(idProduktua, hornitzaireId, kategoriaId, izena, marka, mota,
deskribapena, irudiaUrl, biltegiId,
produktuEgoera, produktuEgoeraOharra, salgai, salmentaPrezioa,
stock, eskaintza, zergakEhunekoa,
sortzeData, egeneratzeData);
 this.pantailaTeknologia = pantailaTeknologia;
 this.pantailaHazbeteak = pantailaHazbeteak;
 this.biltegiratzeGb = biltegiratzeGb;
 this.ramGb = ramGb;
 this.kameraNagusaMp = kameraNagusaMp;
 this.bateriaMah = bateriaMah;
 this.sistemaEragilea = sistemaEragilea;
 this.sareak = sareak;
}

/**
 * Pantaila teknologia lortzen du.
 *
 * @return Pantaila teknologia.
 */
public String getPantailaTeknologia() {
 return pantailaTeknologia;
}

```

```
}

/**
 * Pantaila teknologia ezartzen du.
 *
 * @param pantailaTeknologia Teknologia berria.
 */
public void setPantailaTeknologia(String pantailaTeknologia) {
 this.pantailaTeknologia = pantailaTeknologia;
}

/**
 * Pantaila tamaina lortzen du.
 *
 * @return Tamaina hazbetetan.
 */
public BigDecimal getPantailaHazbeteak() {
 return pantailaHazbeteak;
}

/**
 * Pantaila tamaina ezartzen du.
 *
 * @param pantailaHazbeteak Tamaina berria.
 */
public void setPantailaHazbeteak(BigDecimal pantailaHazbeteak) {
 this.pantailaHazbeteak = pantailaHazbeteak;
}

/**
 * Biltegiratze kapazitatea lortzen du.
 *
 * @return Biltegiratzea GBtan.
 */
public int getBiltegiratzeaGb() {
 return biltegiratzeaGb;
}

/**
 * Biltegiratze kapazitatea ezartzen du.
 *
 * @param biltegiratzeaGb Kapazitate berria.
 */
public void setBiltegiratzeaGb(int biltegiratzeaGb) {
 this.biltegiratzeaGb = biltegiratzeaGb;
}

/**
 * RAM memoria lortzen du.
 *
 * @return RAM memoria GBtan.
 */
public int getRamGb() {
 return ramGb;
}

/**
```

```
* RAM memoria ezartzen du.
*
* @param ramGb RAM memoria berria.
*/
public void setRamGb(int ramGb) {
 this.ramGb = ramGb;
}

/**
 * Kamera nagusiko megapixelak lortzen du.
 *
* @return Kamera MP.
*/
public int getKameraNagusaMp() {
 return kameraNagusaMp;
}

/**
 * Kamera nagusiko megapixelak ezartzen du.
 *
* @param kameraNagusaMp Kamera MP berria.
*/
public void setKameraNagusaMp(int kameraNagusaMp) {
 this.kameraNagusaMp = kameraNagusaMp;
}

/**
 * Bateria kapazitatea lortzen du.
 *
* @return Bateria mAh-tan.
*/
public int getBateriaMah() {
 return bateriaMah;
}

/**
 * Bateria kapazitatea ezartzen du.
 *
* @param bateriaMah Bateria kapazitate berria.
*/
public void setBateriaMah(int bateriaMah) {
 this.bateriaMah = bateriaMah;
}

/**
 * Sistema eragilea lortzen du.
 *
* @return Sistema eragilea.
*/
public String getSistemaEragilea() {
 return sistemaEragilea;
}

/**
 * Sistema eragilea ezartzen du.
 *
* @param sistemaEragilea Sistema eragile berria.
```

```

 */
 public void setSistemaEragilea(String sistemaEragilea) {
 this.sistemaEragilea = sistemaEragilea;
 }

 /**
 * Sare motak lortzen ditu.
 *
 * @return Sareak.
 */
 public String getSareak() {
 return sareak;
 }

 /**
 * Sare motak ezartzen ditu.
 *
 * @param sareak Sarea berria.
 */
 public void setSareak(String sareak) {
 this.sareak = sareak;
 }
}

```

## Pantaila.java

```

package model;

import java.math.BigDecimal;
import java.sql.Timestamp;

/**
 * Pantaila klasea.
 * Produktua klasaren azpiklasea da, eta monitor edo pantailen informazioa
 * kudeatzen du.
 * Bereizmena, panel mota, freskatze tasa, konexioak eta kurbatura bezalako
 * ezaugarriak gordetzen ditu.
 */
public class Pantaila extends Produktua {
 private BigDecimal hazbeteak;
 private String bereizmena;
 private String panelMota;
 private int freskatzeTasaHz;
 private String konexioak;
 private String kurbatura;

 /**
 * Pantaila eraikitzailea.
 *
 * @param idProduktua Produktuaren IDa.
 * @param hornitzaireId Hornitzairearen IDa.
 */
}

```

```

* @param kategoriaId Kategoriaren IDa.
* @param izena Izena.
* @param marka Marka.
* @param mota Mota (Pantaila).
* @param deskribapena Deskribapena.
* @param irudiaUrl Irudiaren URLa.
* @param biltegiId Biltegiaren IDa.
* @param produktuEgoera Produktuaren egoera.
* @param produktuEgoeraOharra Egoeraren oharra.
* @param salgai Salgai dagoen.
* @param salmentaPrezioa Salmenta prezioa.
* @param stock Stock kopurua.
* @param eskaintza Eskaintza prezioa.
* @param zergakEhunekoa Zergen ehunekoa.
* @param sortzeData Sortze data.
* @param eguneratzeData Eguneratze data.
* @param hazbeteak Pantailaren tamaina hazbetetan.
* @param bereizmena Bereizmena (adib. 1920x1080).
* @param panelMota Panel mota (IPS, VA, TN...).
* @param freskatzeTasaHz Freskatze tasa Hz-tan.
* @param konexioak Konexioak (HDMI, DP...).
* @param kurbatura Kurbatura (baldin badauka).

*/
public Pantaila(int idProduktua, int hornitzaireId, int kategoriaId, String
izena, String marka, String mota,
String deskribapena, String irudiaUrl, Integer biltegiId, String
produktuEgoera,
String produktuEgoeraOharra, boolean salgai, BigDecimal
salmentaPrezioa, int stock,
BigDecimal eskaintza, BigDecimal zergakEhunekoa, Timestamp sortzeData,
Timestamp eguneratzeData,
BigDecimal hazbeteak, String bereizmena, String panelMota, int
freskatzeTasaHz,
String konexioak, String kurbatura) {
super(idProduktua, hornitzaireId, kategoriaId, izena, marka, mota,
deskribapena, irudiaUrl, biltegiId,
produktuEgoera, produktuEgoeraOharra, salgai, salmentaPrezioa,
stock, eskaintza, zergakEhunekoa,
sortzeData, eguneratzeData);
this.hazbeteak = hazbeteak;
this.bereizmena = bereizmena;
this.panelMota = panelMota;
this.freskatzeTasaHz = freskatzeTasaHz;
this.konexioak = konexioak;
this.kurbatura = kurbatura;
}

/**
* Pantaila tamaina lortzen du.
*
* @return Tamaina hazbetetan.
*/
public BigDecimal getHazbeteak() {
return hazbeteak;
}

/**

```

```
* Pantaila tamaina ezartzen du.
*
* @param hazbeteak Tamaina berria.
*/
public void setHazbeteak(BigDecimal hazbeteak) {
 this.hazbeteak = hazbeteak;
}

/**
 * Bereizmena lortzen du.
 *
* @return Bereizmena.
*/
public String getBereizmena() {
 return bereizmena;
}

/**
 * Bereizmena ezartzen du.
 *
* @param bereizmena Bereizmen berria.
*/
public void setBereizmena(String bereizmena) {
 this.bereizmena = bereizmena;
}

/**
 * Panel mota lortzen du.
 *
* @return Panel mota.
*/
public String getPanelMota() {
 return panelMota;
}

/**
 * Panel mota ezartzen du.
 *
* @param panelMota Panel mota berria.
*/
public void setPanelMota(String panelMota) {
 this.panelMota = panelMota;
}

/**
 * Freskatze tasa lortzen du.
 *
* @return Freskatze tasa Hz-tan.
*/
public int getFreskatzeTasaHz() {
 return freskatzeTasaHz;
}

/**
 * Freskatze tasa ezartzen du.
 *
* @param freskatzeTasaHz Freskatze tasa berria.
*/
```

```

 */
 public void setFreskatzeTasaHz(int freskatzeTasaHz) {
 this.freskatzeTasaHz = freskatzeTasaHz;
 }

 /**
 * Konexioak lortzen ditu.
 *
 * @return Konexioak.
 */
 public String getKonexioak() {
 return konexioak;
 }

 /**
 * Konexioak ezartzen ditu.
 *
 * @param konexioak Konexio berriak.
 */
 public void setKonexioak(String konexioak) {
 this.konexioak = konexioak;
 }

 /**
 * Kurbatura lortzen du.
 *
 * @return Kurbatura.
 */
 public String getKurbatura() {
 return kurbatura;
 }

 /**
 * Kurbatura ezartzen du.
 *
 * @param kurbatura Kurbatura berria.
 */
 public void setKurbatura(String kurbatura) {
 this.kurbatura = kurbatura;
 }
}

```

## Pertsona.java

```

package model;

import java.sql.Date;
import java.sql.Timestamp;

/*
 * Superklasea langile, bezero eta hornitzaireen atributu komunak gordetzeko.

```

```

*/
public abstract class Pertsona {
 protected int id;
 protected String izena;
 protected String abizena;
 protected String nanIfz;
 protected Date jaiotzaData;
 protected String helbidea;
 protected int herriaId;
 protected String postaKodea;
 protected String telefonoa;
 protected String emaila;
 protected String hizkuntza;
 protected String pasahitza;
 protected boolean aktibo;
 protected Timestamp altaData;
 protected Timestamp eguneratzeData;

 /**
 * Pertsona eraikitzalea.
 *
 * @param id IDa.
 * @param izena Izena.
 * @param abizena Abizena.
 * @param nanIfz IFZ edo NAN.
 * @param jaiotzaData Jaiotza data.
 * @param helbidea Helbidea.
 * @param herriaId Herriaren IDa.
 * @param postaKodea Posta kodea.
 * @param telefonoa Telefonoa.
 * @param emaila Emaila.
 * @param hizkuntza Hizkuntza.
 * @param pasahitza Pasahitza.
 * @param aktibo Aktibo dagoen.
 * @param altaData Alta data.
 * @param eguneratzeData Eguneratze data.
 */

 public Pertsona(int id, String izena, String abizena, String nanIfz, Date
jaiotzaData, String helbidea,
 int herriaId, String postaKodea, String telefonoa, String emaila,
String hizkuntza,
 String pasahitza, boolean aktibo, Timestamp altaData, Timestamp
eguneratzeData) {
 this.id = id;
 this.izena = izena;
 this.abizena = abizena;
 this.nanIfz = nanIfz;
 this.jaiotzaData = jaiotzaData;
 this.helbidea = helbidea;
 this.herriaId = herriaId;
 this.postaKodea = postaKodea;
 this.telefonoa = telefonoa;
 this.emaila = emaila;
 this.hizkuntza = hizkuntza;
 this.pasahitza = pasahitza;
 this.aktibo = aktibo;
 this.altaData = altaData;
 }
}

```

```
 this.eguneratzeData = eguneratzeData;
 }

// Getters eta Setters
/***
 * IDa lortzen du.
 *
 * @return IDa.
 */
public int getId() {
 return id;
}

/***
 * IDa ezartzen du.
 *
 * @param id ID berria.
 */
public void setId(int id) {
 this.id = id;
}

/***
 * Izena lortzen du.
 *
 * @return Izena.
 */
public String getIzena() {
 return izena;
}

/***
 * Izena ezartzen du.
 *
 * @param izena Izen berria.
 */
public void setIzena(String izena) {
 this.izena = izena;
}

/***
 * Abizena lortzen du.
 *
 * @return Abizena.
 */
public String getAbizena() {
 return abizena;
}

/***
 * Abizena ezartzen du.
 *
 * @param abizena Abizen berria.
 */
public void setAbizena(String abizena) {
 this.abizena = abizena;
}
```

```
/**
 * NAN edo IFZ lortzen du.
 *
 * @return NAN edo IFZ.
 */
public String getNanIfz() {
 return nanIfz;
}

/**
 * NAN edo IFZ ezartzen du.
 *
 * @param nanIfz NAN edo IFZ berria.
 */
public void setNanIfz(String nanIfz) {
 this.nanIfz = nanIfz;
}

/**
 * Jaiotza data lortzen du.
 *
 * @return Jaiotza data.
 */
public Date getJaiotzaData() {
 return jaiotzaData;
}

/**
 * Jaiotza data ezartzen du.
 *
 * @param jaiotzaData Jaiotza data berria.
 */
public void setJaiotzaData(Date jaiotzaData) {
 this.jaiotzaData = jaiotzaData;
}

/**
 * Helbidea lortzen du.
 *
 * @return Helbidea.
 */
public String getHelbidea() {
 return helbidea;
}

/**
 * Helbidea ezartzen du.
 *
 * @param helbidea Helbide berria.
 */
public void setHelbidea(String helbidea) {
 this.helbidea = helbidea;
}

/**
 * Herriaren IDa lortzen du.
 */
```

```
*
* @return Herriaren IDa.
*/
public int getHerriaId() {
 return herriaId;
}

/**
 * Herriaren IDa ezartzen du.
 *
 * @param herriaId Herriaren ID berria.
*/
public void setHerriaId(int herriaId) {
 this.herriaId = herriaId;
}

/**
 * Posta kodea lortzen du.
 *
 * @return Posta kodea.
*/
public String getPostaKodea() {
 return postaKodea;
}

/**
 * Posta kodea ezartzen du.
 *
 * @param postaKodea Posta kode berria.
*/
public void setPostaKodea(String postaKodea) {
 this.postaKodea = postaKodea;
}

/**
 * Telefona lortzen du.
 *
 * @return Telefona.
*/
public String getTelefona() {
 return telefonoa;
}

/**
 * Telefona ezartzen du.
 *
 * @param telefonoa Telefono berria.
*/
public void setTelefona(String telefonoa) {
 this.telefona = telefonoa;
}

/**
 * Emaila lortzen du.
 *
 * @return Emaila.
*/
```

```
public String getEmaila() {
 return emaila;
}

/**
 * Emaila ezartzen du.
 *
 * @param emaila Email berria.
 */
public void setEmaila(String emaila) {
 this.emaila = emaila;
}

/**
 * Hizkuntza lortzen du.
 *
 * @return Hizkuntza.
 */
public String getHizkuntza() {
 return hizkuntza;
}

/**
 * Hizkuntza ezartzen du.
 *
 * @param hizkuntza Hizkuntza berria.
 */
public void setHizkuntza(String hizkuntza) {
 this.hizkuntza = hizkuntza;
}

/**
 * Pasahitza lortzen du.
 *
 * @return Pasahitza.
 */
public String getPasahitza() {
 return pasahitza;
}

/**
 * Pasahitza ezartzen du.
 *
 * @param pasahitza Pasahitz berria.
 */
public void setPasahitza(String pasahitza) {
 this.pasahitza = pasahitza;
}

/**
 * Aktibo dagoen lortzen du.
 *
 * @return True aktibo badago, false bestela.
 */
public boolean isAktibo() {
 return aktibo;
}
```

```

/**
 * Aktibo egoera ezartzen du.
 *
 * @param aktibo Aktibo egoera berria.
 */
public void setAktibo(boolean aktibo) {
 this.aktibo = aktibo;
}

/**
 * Alta data lortzen du.
 *
 * @return Alta data.
 */
public Timestamp getAltaData() {
 return altaData;
}

/**
 * Alta data ezartzen du.
 *
 * @param altaData Alta data berria.
 */
public void setAltaData(Timestamp altaData) {
 this.altaData = altaData;
}

/**
 * Egeneratze data lortzen du.
 *
 * @return Egeneratze data.
 */
public Timestamp getEgeneratzeData() {
 return egeneratzeData;
}

/**
 * Egeneratze data ezartzen du.
 *
 * @param egeneratzeData Egeneratze data berria.
 */
public void setEgeneratzeData(Timestamp egeneratzeData) {
 this.egeneratzeData = egeneratzeData;
}
}

```

# Produktua.java

```
package model;
```

```

import java.math.BigDecimal;
import java.sql.Timestamp;

/**
 * Produktua klase abstraktua.
 * Produkta mota guztien oinarrizko atributuak eta metodoak definitzen ditu.
 * Hornitzaila, kategoria, prezioa, stock-a eta egoera bezalako datuak
 * kudeatzen ditu.
 */
public abstract class Produktua {
 private int idProduktua;
 private int hornitzailleId;
 private int kategoriaId;
 private String izena;
 private String marka;
 private String mota;
 private String deskribapena;
 private String irudiaUrl;
 private Integer biltegiId;
 private String produktuEgoera;
 private String produktuEgoeraOharra;
 private boolean salgai;
 private BigDecimal salmentaPrezioa;
 private int stock;
 private BigDecimal eskaintza;
 private BigDecimal zergakEhunekoa;
 private Timestamp sortzeData;
 private Timestamp eguneratzeData;

 /**
 * Produktua eraikitzalea.
 *
 * @param idProduktua Produktuaren IDa.
 * @param hornitzailleId Hornitzailaren IDa.
 * @param kategoriaId Kategoriaren IDa.
 * @param izena Izena.
 * @param marka Marka.
 * @param mota Mota.
 * @param deskribapena Deskribapena.
 * @param irudiaUrl Irudiaren URLa.
 * @param biltegiId Biltegiaren IDa.
 * @param produktuEgoera Produktuaren egoera.
 * @param produktuEgoeraOharra Egoeraren oharra.
 * @param salgai Salgai dagoen.
 * @param salmentaPrezioa Salmenta prezioa.
 * @param stock Stock kopurua.
 * @param eskaintza Eskaintza prezioa.
 * @param zergakEhunekoa Zergen ehunekoa.
 * @param sortzeData Sortze data.
 * @param eguneratzeData Eguneratze data.
 */
 public Produktua(int idProduktua, int hornitzailleId, int kategoriaId, String
izena, String marka, String mota,
 String deskribapena, String irudiaUrl, Integer biltegiId, String
produktuEgoera,
 String produktuEgoeraOharra, boolean salgai, BigDecimal
salmentaPrezioa, int stock,

```

```
 BigDecimal eskaintza, BigDecimal zergakEhunekoa, Timestamp sortzeData,
Timestamp eguneratzeData) {
 this.idProduktua = idProduktua;
 this.hornitzaireId = hornitzaireId;
 this.kategoriaId = kategoriaId;
 this.izena = izena;
 this.marka = marka;
 this.mota = mota;
 this.deskribapena = deskribapena;
 this.irudiaUrl = irudiaUrl;
 this.biltegiId = biltegiId;
 this.produktuEgoera = produktuEgoera;
 this.produktuEgoera0harra = produktuEgoera0harra;
 this.salgai = salgai;
 this.salmentaPrezioa = salmentaPrezioa;
 this.stock = stock;
 this.eskaintza = eskaintza;
 this.zergakEhunekoa = zergakEhunekoa;
 this.sortzeData = sortzeData;
 this.eguneratzeData = eguneratzeData;
}

/**
 * Produktuaren IDa lortzen du.
 *
 * @return IDa.
 */
public int getIdProduktua() {
 return idProduktua;
}

/**
 * Produktuaren IDa ezartzen du.
 *
 * @param idProduktua ID berria.
 */
public void setIdProduktua(int idProduktua) {
 this.idProduktua = idProduktua;
}

/**
 * Hornitzairearen IDa lortzen du.
 *
 * @return Hornitzairearen IDa.
 */
public int getHornitzaireId() {
 return hornitzaireId;
}

/**
 * Hornitzairearen IDa ezartzen du.
 *
 * @param hornitzaireId Hornitzaire ID berria.
 */
public void setHornitzaireId(int hornitzaireId) {
 this.hornitzaireId = hornitzaireId;
}
```

```
/**
 * Kategoriaren IDa lortzen du.
 *
 * @return Kategoriaren IDa.
 */
public int getKategoriaId() {
 return kategoriaId;
}

/**
 * Kategoriaren IDa ezartzen du.
 *
 * @param kategoriaId Kategoria ID berria.
 */
public void setKategoriaId(int kategoriaId) {
 this.kategoriaId = kategoriaId;
}

/**
 * Izena lortzen du.
 *
 * @return Izena.
 */
public String getIzena() {
 return izena;
}

/**
 * Izena ezartzen du.
 *
 * @param izena Izen berria.
 */
public void setIzena(String izena) {
 this.izena = izena;
}

/**
 * Marka lortzen du.
 *
 * @return Marka.
 */
public String getMarka() {
 return marka;
}

/**
 * Marka ezartzen du.
 *
 * @param marka Marka berria.
 */
public void setMarka(String marka) {
 this.marka = marka;
}

/**
 * Mota lortzen du.
 */
```

```
*
* @return Mota.
*/
public String getMota() {
 return mota;
}

/**
 * Mota ezartzen du.
 *
 * @param mota Mota berria.
 */
public void setMota(String mota) {
 this.mota = mota;
}

/**
 * Deskribapena lortzen du.
 *
 * @return Deskribapena.
 */
public String getDeskribapena() {
 return deskribapena;
}

/**
 * Deskribapena ezartzen du.
 *
 * @param deskribapena Deskribapen berria.
 */
public void setDeskribapena(String deskribapena) {
 this.deskribapena = deskribapena;
}

/**
 * Irudiaren URLa lortzen du.
 *
 * @return Irudiaren URLa.
 */
public String getIrudiaUrl() {
 return irudiaUrl;
}

/**
 * Irudiaren URLa ezartzen du.
 *
 * @param irudiaUrl URL berria.
 */
public void setIrudiaUrl(String irudiaUrl) {
 this.irudiaUrl = irudiaUrl;
}

/**
 * Biltegiaren IDa lortzen du.
 *
 * @return Biltegiaren IDa.
 */
```

```
public Integer getBiltegiId() {
 return biltegiId;
}

/**
 * Biltegiaren IDa ezartzen du.
 *
 * @param biltegiId Biltegi ID berria.
 */
public void setBiltegiId(Integer biltegiId) {
 this.biltegiId = biltegiId;
}

/**
 * Produktuaren egoera lortzen du.
 *
 * @return Egoera.
 */
public String getProduktuEgoera() {
 return produktuEgoera;
}

/**
 * Produktuaren egoera ezartzen du.
 *
 * @param produktuEgoera Egoera berria.
 */
public void setProduktuEgoera(String produktuEgoera) {
 this.produktuEgoera = produktuEgoera;
}

/**
 * Produktuaren egoeraren oharra lortzen du.
 *
 * @return Oharra.
 */
public String getProduktuEgoeraOharra() {
 return produktuEgoeraOharra;
}

/**
 * Produktuaren egoeraren oharra ezartzen du.
 *
 * @param produktuEgoeraOharra Ohar berria.
 */
public void setProduktuEgoeraOharra(String produktuEgoeraOharra) {
 this.produktuEgoeraOharra = produktuEgoeraOharra;
}

/**
 * Salgai dagoen lortzen du.
 *
 * @return True salgai badago, false bestela.
 */
public boolean isSalgai() {
 return salgai;
}
```

```
/**
 * Salgai egoera ezartzen du.
 *
 * @param salgai Salgai egoera berria.
 */
public void setSalgai(boolean salgai) {
 this.salgai = salgai;
}

/**
 * Salmenta prezioa lortzen du.
 *
 * @return Prezioa.
 */
public BigDecimal getSalmentaPrezioa() {
 return salmentaPrezioa;
}

/**
 * Salmenta prezioa ezartzen du.
 *
 * @param salmentaPrezioa Prezio berria.
 */
public void setSalmentaPrezioa(BigDecimal salmentaPrezioa) {
 this.salmentaPrezioa = salmentaPrezioa;
}

/**
 * Stock kopurua lortzen du.
 *
 * @return Stock.
 */
public int getStock() {
 return stock;
}

/**
 * Stock kopurua ezartzen du.
 *
 * @param stock Stock berria.
 */
public void setStock(int stock) {
 this.stock = stock;
}

/**
 * Eskaintza prezioa lortzen du.
 *
 * @return Eskaintza.
 */
public BigDecimal getEskaintza() {
 return eskaintza;
}

/**
 * Eskaintza prezioa ezartzen du.
 */
```

```
*
* @param eskaintza Eskaintza berria.
*/
public void setEskaintza(BigDecimal eskaintza) {
 this.eskaintza = eskaintza;
}

/**
 * Zergen ehunekoak lortzen du.
 *
 * @return Zergak.
*/
public BigDecimal getZergakEhunekoak() {
 return zergakEhunekoak;
}

/**
 * Zergen ehunekoak ezartzen du.
 *
 * @param zergakEhunekoak Zerga ehuneko berria.
*/
public void setZergakEhunekoak(BigDecimal zergakEhunekoak) {
 this.zergakEhunekoak = zergakEhunekoak;
}

/**
 * Sortze data lortzen du.
 *
 * @return Sortze data.
*/
public Timestamp getSortzeData() {
 return sortzeData;
}

/**
 * Sortze data ezartzen du.
 *
 * @param sortzeData Data berria.
*/
public void setSortzeData(Timestamp sortzeData) {
 this.sortzeData = sortzeData;
}

/**
 * Eguneratze data lortzen du.
 *
 * @return Eguneratze data.
*/
public Timestamp getEguneratzeData() {
 return eguneratzeData;
}

/**
 * Eguneratze data ezartzen du.
 *
 * @param eguneratzeData Data berria.
*/
```

```
 public void setEguneratzeData(Timestamp eguneratzeData) {
 this.eguneratzeData = eguneratzeData;
 }
}
```

# ProduktuKategoria.java

```
package model;

/**
 * ProduktuKategoria klasea.
 * Produktu kategorien informazioa kudeatzeko klasea.
 * Kategoriaren IDa eta izena gordetzen ditu.
 */
public class ProduktuKategoria {
 private int idKategoria;
 private String izena;

 /**
 * ProduktuKategoria eraikitzalea.
 *
 * @param idKategoria Kategoriaren IDa.
 * @param izena Kategoriaren izena.
 */
 public ProduktuKategoria(int idKategoria, String izena) {
 this.idKategoria = idKategoria;
 this.izena = izena;
 }

 /**
 * Kategoriaren IDa lortzen du.
 *
 * @return IDa.
 */
 public int getIdKategoria() {
 return idKategoria;
 }

 public void setIdKategoria(int idKategoria) {
 this.idKategoria = idKategoria;
 }

 /**
 * Kategoriaren izena lortzen du.
 *
 * @return Izena.
 */
 public String getIzena() {
 return izena;
 }
}
```

```
 public void setIzena(String izena) {
 this.izena = izena;
 }
}
```

# SalmentaLangilea.java

```
package model;

import db.DB_Konexioa;
import java.io.File;
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;

/**
 * SalmentaLangilea klasea.
 * Langilea klasaren azpiklasea da, eta salmenta arloko langileen
 * funtzionalitateak kudeatzen ditu.
 * Bezeroak, eskaerak, fakturak eta produktuak kudeatzeko metodoak ditu.
 */
public class SalmentaLangilea extends Langilea {

 /**
 * SalmentaLangilea eraikitzalea.
 * Langilea objektu batetik abiatuta sortzen da.
 *
 * @param l Langilea objektua.
 */
 public SalmentaLangilea(Langilea l) {
 super(l.getIdLangilea(), l.getIzena(), l.getAbizena(), l.getNan(),
l.getJaiotzaData(), l.getHerriaId(),
 l.getHelbidea(), l.getPostaKodea(), l.getTelefonoa(),
l.getEmaila(), l.getHizkuntza(),
 l.getPasahitza(), l.getSaltoTxartelaUid(), l.getAltaData(),
l.getEguneratzeData(),
 l.isAktibo(), l.getSailaId(), l.getIban(), l.getKurrikuluma());
 }

 /**
 * Faktura bat sortu eskaera batetik abiatuta.
 *
 * @param idEskaera Eskaeraren IDa
 * @return Sortutako PDF fitxategia
 * @throws Exception
 */
}
```

```

/*
 */
/**
 * Faktura bat sortzen du eskaera batetik abiatuta.
 * PDF fitxategia sortzen du eta datu-basean erregistratzen du.
 *
 * @param idEskaera Eskaeraren IDa.
 * @return Sortutako PDF fitxategia, edo null errorea gertatu bada.
 * @throws Exception Errorea prozesuan.
 */
private static final String FAKTURA_BIDEA = "C:\\\\Xampp\\\\htdocs\\\\fakturak";

public File fakturaSortu(int idEskaera) throws Exception {
 // Faktura karpeta ziurtatu
 File karpeta = new File(FAKTURA_BIDEA);
 if (!karpeta.exists()) {
 karpeta.mkdirs();
 }

 File fakturaFitxategia = new File(karpeta, "faktura_" + idEskaera +
".pdf");

 // Datuak lortu
 Eskaera eskaera = eskaeraIkusi(idEskaera);
 if (eskaera == null) {
 throw new Exception("Ez da eskaera aurkitu: " + idEskaera);
 }

 Bezeroa bezeroa = bezeroaIkusi(eskaera.getBezeroaId());
 if (bezeroa == null) {
 throw new Exception("Ez da bezeroa aurkitu eskaerarentzat: " +
idEskaera);
 }

 List<EskaeraLerroa> lerroak = eskaeraLerroakIkusi(idEskaera);

 // PDF Sortu
 FakturaPDF.sortu(fakturaFitxategia.getAbsolutePath(), eskaera, bezeroa,
 lerroak);

 // DBan gorde (Eskaera taulan)
 String fakturaZenbakia = "FAK-" + idEskaera + "-" +
System.currentTimeMillis();
 String sqlUpdate = "UPDATE eskaerak SET faktura_zenbakia = ?, faktura_url = ?
 WHERE id_eskaera = ?";

 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pstUpdate = konexioa.prepareStatement(sqlUpdate)) {
 pstUpdate.setString(1, fakturaZenbakia);
 pstUpdate.setString(2, fakturaFitxategia.getAbsolutePath());
 pstUpdate.setInt(3, idEskaera);
 pstUpdate.executeUpdate();
 }

 return fakturaFitxategia;
}

```

```

/**
 * Faktura ezabatu eskaera IDa erabiliz.
 *
 * @param idEskaera Eskaeraren IDa
 */
/**
 * Faktura bat ezabatzen du eskaera IDa erabiliz.
 * Fitxategi fisikoa eta datu-baseko erregistroa ezabatzen ditu.
 *
 * @param idEskaera Eskaeraren IDa.
 * @throws Exception Errorea ezabatzean.
 */
public void fakturaEzabatu(int idEskaera) throws Exception {
 String sqlSelect = "SELECT faktura_url FROM eskaerak WHERE id_eskaera = ?";
 String sqlUpdate = "UPDATE eskaerak SET faktura_zenbakia = NULL,
faktura_url = NULL WHERE id_eskaera = ?";

 try (Connection konexioa = DB_Konexioa.konektatu()) {
 // 1. Fitxategia lortu eta ezabatu
 try (PreparedStatement pstSelect =
konexioa.prepareStatement(sqlSelect)) {
 pstSelect.setInt(1, idEskaera);
 try (ResultSet rs = pstSelect.executeQuery()) {
 if (rs.next()) {
 String bidea = rs.getString("faktura_url");
 if (bidea != null) {
 File fitxategia = new File(bidea);
 if (fitxategia.exists()) {
 fitxategia.delete();
 }
 }
 } else {
 throw new Exception("Eskaera honek ez du fakturarik.");
 }
 }
 }
 // 2. DBtik egunerau (ezabatu beharrean)
 try (PreparedStatement pstUpdate =
konexioa.prepareStatement(sqlUpdate)) {
 pstUpdate.setInt(1, idEskaera);
 pstUpdate.executeUpdate();
 }
 }
}

// -----
// BEZEROEN KUDEAKETA
// -----

/***
 * Bezero berria sortu.
 *
 * @param b Bezero objektua
 */
/***
 * Bezero berri bat sortzen du datu-basean.
*/

```

```

/*
 * @param b Bezeroa objektua.
 * @throws ExceptionErrorea sortzean.
 */
public void bezeroBerriaSortu(Bezeroa b) throws Exception {
 String sql = "INSERT INTO bezeroak (izena_edo_soziala, abizena, ifz_nan,
jaiotza_data, sexua, " +
 "bezero_ordinaketa_txartela, helbidea, herria_id, posta_kodea,
telefonoa, emaila, " +
 "hizkuntza, pasahitza, aktibo) " +
 "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);"

 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {

 pst.setString(1, b.getIzenaEdoSoziala());
 pst.setString(2, b.getAbizena());
 pst.setString(3, b.getIfzNan());
 pst.setDate(4, b.getJaiotzaData());
 pst.setString(5, b.getSexua());
 pst.setString(6, b.getBezeroOrdinaketaTxartela());
 pst.setString(7, b.getHelbidea());
 pst.setInt(8, b.getHerriaId());
 pst.setString(9, b.getPostaKodea());
 pst.setString(10, b.getTelefonoa());
 pst.setString(11, b.getEmaila());
 pst.setString(12, b.getHizkuntza());
 pst.setString(13, b.getPasahitza());
 pst.setBoolean(14, b.isAktibo());

 pst.executeUpdate();
 }
}

/**
 * Bezeroaren informazioa editatu.
 *
 * @param b Bezero objektua (id-a barne)
 */
/**
 * Bezero baten informazioa editatzen du.
 *
 * @param b Bezero objektua (id-a barne).
 * @throws ExceptionErrorea editatzean.
 */
public void bezeroaEditatu(Bezeroa b) throws Exception {
 String sql = "UPDATE bezeroak SET izena_edo_soziala=? , abizena=? ,
ifz_nan=? , jaiotza_data=? , " +
 "sexua=? , bezero_ordinaketa_txartela=? , helbidea=? , herria_id=? ,
posta_kodea=? , " +
 "telefonoa=? , emaila=? , hizkuntza=? , pasahitza=? , aktibo=? WHERE
id_bezeroa=?";

 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {

 pst.setString(1, b.getIzenaEdoSoziala());

```

```

 pst.setString(2, b.getAbizena());
 pst.setString(3, b.getIfzNan());
 pst.setDate(4, b.getJaiotzaData());
 pst.setString(5, b.getSexua());
 pst.setString(6, b.getBezeroOrdainketaTxartela());
 pst.setString(7, b.getHelbidea());
 pst.setInt(8, b.getHerriaId());
 pst.setString(9, b.getPostaKodea());
 pst.setString(10, b.getTelefonoa());
 pst.setString(11, b.getEmaila());
 pst.setString(12, b.getHizkuntza());
 pst.setString(13, b.getPasahitza());
 pst.setBoolean(14, b.isAktibo());
 pst.setInt(15, b.getIdBezeroa());

 pst.executeUpdate();
 }
}

/**
 * Bezero bat ezabatu (Order check included)
 *
 * @param idBezeroa Bezeroaren IDa
 */
/**
 * Bezero bat ezabatzen du.
 * Lehenik egiaztatzen du ea eskaerarik duen; badu, ezin da ezabatu.
 *
 * @param idBezeroa Bezeroaren IDa.
 * @throws Exception Errorea ezabatzean edo eskaerak baditu.
 */
public void bezeroaKendu(int idBezeroa) throws Exception {
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 // Check for existing orders
 try (PreparedStatement pstCheck = konexioa
 .prepareStatement("SELECT COUNT(*) FROM eskaerak WHERE
bezeroa_id = ?")) {
 pstCheck.setInt(1, idBezeroa);
 ResultSet rs = pstCheck.executeQuery();
 if (rs.next() && rs.getInt(1) > 0) {
 throw new Exception("Ezin da bezeroa ezabatu: Eskaerak ditu.");
 }
 }

 String sql = "DELETE FROM bezeroak WHERE id_bezeroa=?";
 try (PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idBezeroa);
 pst.executeUpdate();
 }
 }
}

/**
 * Bezeroaren informazioa ikusi.
 *
 * @param idBezeroa Bezeroaren IDa
 * @return Bezero objektua

```

```

/*
/**
 * Bezero baten informazioa lortzen du.
 *
 * @param idBezeroa Bezeroaren IDa.
 * @return Bezero objektua edo null aurkitzen ez bada.
 * @throws ExceptionErrorea irakurtzean.
 */
public Bezeroa bezeroaIkusi(int idBezeroa) throws Exception {
 String sql = "SELECT * FROM bezeroak WHERE id_bezeroa=?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idBezeroa);
 try (ResultSet rs = pst.executeQuery()) {
 if (rs.next()) {
 return new Bezeroa(
 rs.getInt("id_bezeroa"),
 rs.getString("izena_edo_soziala"),
 rs.getString("abizena"),
 rs.getString("ifz_nan"),
 rs.getDate("jaiotza_data"),
 rs.getString("sexua"),
 rs.getString("bezero_ordainketa_txartela"),
 rs.getString("helbidea"),
 rs.getInt("herria_id"),
 rs.getString("posta_kodea"),
 rs.getString("telefonoa"),
 rs.getString("emaila"),
 rs.getString("hizkuntza"),
 rs.getString("pasahitzta"),
 rs.getTimestamp("alta_data"),
 rs.getTimestamp("egunearatze_data"),
 rs.getBoolean("aktibo"));
 }
 }
 }
 return null; // Ez da aurkitu
}

// -----
// FAKTUREN KUDEAKETA
// -----

/**
 * Bezeroaren faktura ezabatu.
 *
 * @param idFaktura Fakturaren IDa
 */
/**
 * Bezero baten faktura ezabatzen du IDaren arabera.
 *
 * @param idFaktura Fakturaren IDa.
 * @throws ExceptionErrorea ezabatzean.
 */
public void bezeroFakturaEzabatu(int idFaktura) throws Exception {
 // In current schema, idFaktura is equivalent to idEskaera
 fakturaEzabatu(idFaktura);
}

```

```

}

// -----
// PRODUKTUEN KUDEAKETA
// -----

/***
 * Produktuak ikusi motaren arabera iragaziz.
 *
 * @param mota Produktu mota (adib. 'Eramangarria', 'Mugikorra'...). Null edo
 * hutsik bada, denak itzuli.
 * @return Produktu zerrenda
 */
/***
 * Produktuak ikusten ditu, aukeran motaren arabera iragaziz.
 *
 * @param mota Produktu mota (adib. "Eramangarria", "Guztiak"...). Null edo
 * hutsik bada, denak itzultzen dira.
 * @return Produktu zerrenda.
 * @throws Exception Errorea datuak lortzean.
 */
public List<Produktua> produktuakIkusi(String mota) throws Exception {
 List<Produktua> produktuak = new ArrayList<>();
 String sql = "SELECT * FROM produktuak";

 // Iragazkia aplikatu
 if (mota != null && !mota.trim().isEmpty() &&
!mota.equalsIgnoreCase("Guztiak")) {
 sql += " WHERE mota = ?";
 }

 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {

 if (mota != null && !mota.trim().isEmpty() &&
!mota.equalsIgnoreCase("Guztiak")) {
 pst.setString(1, mota);
 }

 try (ResultSet rs = pst.executeQuery()) {
 while (rs.next()) {
 produktuak.add(instanziatuProduktua(rs));
 }
 }
 }
 return produktuak;
}

/***
 * Produktua ikusi.
 *
 * @param idProduktua Produktuaren IDa
 * @return Produktua objektua
 */
/***
 * Produktu baten informazio zehatza lortzen du.
 *

```

```

* @param idProduktua Produktuaren IDa.
* @return Produktu objektua edo null.
* @throws ExceptionErrorea lortzean.
*/
public Produktua produktuaIkusi(int idProduktua) throws Exception {
 String sql = "SELECT * FROM produktuak WHERE id_produktua=?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idProduktua);
 try (ResultSet rs = pst.executeQuery()) {
 if (rs.next()) {
 return instanziatuProduktua(rs);
 }
 }
 }
 return null;
}

/**
 * ResultSet-ek 'Produktua' azpiklase egokia sortzen duen metodo pribatua.
 * Oharra: Soilik 'produktuak' taulako datuak erabiltzen dira.
 * Azpiklaseen eremu espezifikoak null/0 balioekin hasieratzen dira.
 *
 * @param rs ResultSet kurtsorea (dagoeneko .next() eginda egon behar du)
 * @return Produktuaren azpiklasearen instantzia (Eramangarria, Mugikorra...)
 * @throws Exception
*/
/**
 * ResultSet-ek Produktua azpiklase egokia instantziatzeko duen metodo
 * laguntzailea.
 *
 * @param rs ResultSet kurtsorea.
 * @return Produktuaren azpiklasearen instantzia (Eramangarria, Mugikorra...).
 * @throws Exception Errorea sortzean.
*/
private Produktua instanziatuProduktua	ResultSet rs) throws Exception {
 String mota = rs.getString("mota");

 // Komunak diren datuak (Base Constructor arguments)
 int id = rs.getInt("id_produktua");
 int hornitzaleId = rs.getInt("hornitzale_id");
 int kategoriaId = rs.getInt("kategoria_id");
 String izena = rs.getString("izena");
 String marka = rs.getString("marka");
 // mota already retrieved
 String deskribapena = rs.getString("deskribapena");
 String irudiaUrl = rs.getString("irudia_url");
 Object bIdObj = rs.getObject("biltegi_id");
 Integer biltegiId = (bIdObj == null) ? null : ((Number) bIdObj).intValue();
 String egoera = rs.getString("produktu_egoera");
 String egoeraOharra = rs.getString("produktu_egoera_oharra");
 boolean salgai = rs.getBoolean("salgai");
 BigDecimal prezioa = rs.getBigDecimal("salmenta_prezioa");
 int stock = rs.getInt("stock");
 BigDecimal eskaintza = rs.getBigDecimal("eskaintza");
 BigDecimal zergak = rs.getBigDecimal("zergak_ehunekoa");
 Timestamp sortzeData = rs.getTimestamp("sortze_data");
}

```

```

Timestamp eguneratzeData = rs.getTimestamp("eguneratze_data");

switch (mota) {
 case "Eramangarria":
 return new Eramangarria(id, hornitzaireId, kategoriaId, izena,
marka, mota, deskribapena, irudiaUrl,
 biltegiId, egoera, egoeraOharra, salgai, prezioa, stock,
eskaintza, zergak, sortzeData,
 eguneratzeData,
 null, 0, 0, null, 0, null, null);
 case "Mugikorra":
 return new Mugikorra(id, hornitzaireId, kategoriaId, izena, marka,
mota, deskribapena, irudiaUrl,
 biltegiId, egoera, egoeraOharra, salgai, prezioa, stock,
eskaintza, zergak, sortzeData,
 eguneratzeData,
 null, null, 0, 0, 0, 0, null, null);
 case "Mahai-gainekoa":
 return new MahaiGainecko(id, hornitzaireId, kategoriaId, izena,
marka, mota, deskribapena, irudiaUrl,
 biltegiId, egoera, egoeraOharra, salgai, prezioa, stock,
eskaintza, zergak, sortzeData,
 eguneratzeData,
 null, null, 0, 0, null, 0, null);
 case "Tableta":
 return new Tableta(id, hornitzaireId, kategoriaId, izena, marka,
mota, deskribapena, irudiaUrl,
 biltegiId, egoera, egoeraOharra, salgai, prezioa, stock,
eskaintza, zergak, sortzeData,
 eguneratzeData,
 null, 0, null, null, 0, false);
 case "Pantaila":
 return new Pantaila(id, hornitzaireId, kategoriaId, izena, marka,
mota, deskribapena, irudiaUrl,
 biltegiId, egoera, egoeraOharra, salgai, prezioa, stock,
eskaintza, zergak, sortzeData,
 eguneratzeData,
 null, null, null, 0, null, null);
 case "Softwarea":
 return new Softwarea(id, hornitzaireId, kategoriaId, izena, marka,
mota, deskribapena, irudiaUrl,
 biltegiId, egoera, egoeraOharra, salgai, prezioa, stock,
eskaintza, zergak, sortzeData,
 eguneratzeData,
 null, null, null, null);
 case "Zerbitzaria":
 return new Zerbitzaria(id, hornitzaireId, kategoriaId, izena,
marka, mota, deskribapena, irudiaUrl,
 biltegiId, egoera, egoeraOharra, salgai, prezioa, stock,
eskaintza, zergak, sortzeData,
 eguneratzeData,
 0, null, 0, 0, false, null);
 default:
 // Fallback: aurkitu ez den mota bada, edozein kasuan Produktua
abstraktua denez
 // klase anonimo batekin itzuliko dugu, behintzat oinarrizko datuak
edukitzeko.

```

```

 return new Produktua(id, hornitzaireId, kategoriaId, izena, marka,
mota, deskribapena, irudiaUrl,
 biltegiId, egoera, egoeraOharra, salgai, prezioa, stock,
eskaintza, zergak, sortzeData,
 eguneratzeData) {
 };
}
}

/**
 * Produktuari eskaintza aldatu.
 *
 * @param idProduktua Produktuaren IDa
 * @param eskaintza Eskaintza berria
 */
/**
 * Produktu bati eskaintza prezioa aldatzen dio.
 *
 * @param idProduktua Produktuaren IDa.
 * @param eskaintza Eskaintza berria.
 * @throws ExceptionErrorea eguneratzean.
 */
public void produktuariEskaintzaAldatzeko(int idProduktua, BigDecimal
eskaintza) throws Exception {
 String sql = "UPDATE produktuak SET eskaintza=? WHERE id_produktua=?";
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setDecimal(1, eskaintza);
 pst.setInt(2, idProduktua);
 pst.executeUpdate();
 }
 }

 /**
 * Produktua salgai jarri (soilik prezioa definituta badu).
 *
 * @param idProduktua Produktuaren IDa
 */
 /**
 * Produktua salgai jartzen du (soilik prezioa definituta badu).
 *
 * @param idProduktua Produktuaren IDa.
 * @throws ExceptionErrorea eguneratzean.
 */
 public void produktuaSalgaijarri(int idProduktua) throws Exception {
 // Lehenengo prezioa daukala egiaztatu
 String checkSql = "SELECT salmenta_precioa FROM produktuak WHERE
id_produktua=?";
 boolean prezioaDu = false;

 try (Connection konexioa = DB_Konexioa.konektatu());
 PreparedStatement pst = konexioa.prepareStatement(checkSql)) {
 pst.setInt(1, idProduktua);
 try (ResultSet rs = pst.executeQuery()) {
 if (rs.next()) {
 BigDecimal prezioa = rs.getDecimal("salmenta_precioa");
 if (prezioa != null && prezioa.compareTo(BigDecimal.ZERO) > 0)

```

```

{
 prezioaDu = true;
}
}

if (prezioaDu) {
 String updateSql = "UPDATE produktuak SET salgai=1 WHERE
id_produktua=?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(updateSql)) {
 pst.setInt(1, idProduktua);
 pst.executeUpdate();
 }
} else {
 // Aukerakoa: Salbuespena bota edo mezua erakutsi
 // throw new Exception("Produktuak ez du preziorik eta ezin da salgai
jarri.");
}
}

/**
 * Produktuari prezioa jarri.
 *
 * @param idProduktua Produktuaren IDa
 * @param prezioa Prezio berria
 */
/**
 * Produktuari salmenta prezioa ezartzen dio.
 *
 * @param idProduktua Produktuaren IDa.
 * @param prezioa Prezio berria.
 * @throws Exception Errorea eguneraatzean.
 */
public void produktuariPrezioaJarri(int idProduktua, BigDecimal prezioa) throws
Exception {
 String sql = "UPDATE produktuak SET salmenta_precioa=? WHERE
id_produktua=?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setBigDecimal(1, prezioa);
 pst.setInt(2, idProduktua);
 pst.executeUpdate();
 }
}

// -----
// ESKAERA LERROEN KUDEAKETA
// -----

/**
 * Eskaera baten lerroak (produktuak) ikusi.
 *
 * @param idEskaera Eskaeraren IDa
 * @return Eskaera lerroen zerrenda
 */

```

```
/*
 * Eskaera baten lerroak (produktuak) lortzen ditu.
 *
 * @param idEskaera Eskaeraren IDa.
 * @return Eskaera lerroen zerrenda.
 * @throws SQLException Errorea datu-basean.
 */
public List<EskaeraLerroa> eskaeraLerroakIkusi(int idEskaera) throws
SQLException {
 return EskaeraLerroa.eskaeraLerroaIkusi(idEskaera);
}

/**
 * Eskaera bati lerro (produktu) berri bat gehitu.
 *
 * @param idEskaera Eskaeraren IDa
 * @param idProduktua Produktuaren IDa
 * @param kantitatea Kantitatea
 * @param prezioa Unitateko prezioa
 */
/**
 * Eskaera bati lerro (produktu) berri bat gehitzen dio.
 *
 * @param idEskaera Eskaeraren IDa.
 * @param idProduktua Produktuaren IDa.
 * @param kantitatea Kantitatea.
 * @param prezioa Unitateko prezioa.
 * @throws SQLException Errorea gehitzean.
 */
public void eskaeraLerroaGehitu(int idEskaera, int idProduktua, int kantitatea,
BigDecimal prezioa)
 throws SQLException {
 // IDa 0 jartzen dugu, DBak autoincrement bidez sortuko baitu
 EskaeraLerroa el = new EskaeraLerroa(0, idEskaera, idProduktua, kantitatea,
prezioa, "Prestatzen");
 EskaeraLerroa.eskaeraLerroaSortu(el);
}

/**
 * Eskaera lerro bat editatu.
 *
 * @param idEskaeraLerroa Lerroaren IDa
 * @param idEskaera Eskaeraren IDa
 * @param idProduktua Produktuaren IDa
 * @param kantitatea Kantitatea
 * @param prezioa Unitateko prezioa
 */
/**
 * Eskaera lerro bat editatzen du.
 *
 * @param idEskaeraLerroa Lerroaren IDa.
 * @param idEskaera Eskaeraren IDa.
 * @param idProduktua Produktuaren IDa.
 * @param kantitatea Kantitatea.
 * @param prezioa Unitateko prezioa.
 * @throws SQLException Errorea editatzean.
 */
```

```
 public void eskaeraLerroakEditatu(int idEskaeraLerroa, int idEskaera, int
idProduktua, int kantitatea,
 BigDecimal prezioa) throws SQLException {
 EskaeraLerroa el = new EskaeraLerroa(idEskaeraLerroa, idEskaera,
idProduktua, kantitatea, prezioa,
 "Prestatzen");
 EskaeraLerroa.eskaeralerroaEditatu(el);
 }

 /**
 * Eskaera lerro bat ezabatu.
 *
 * @param idEskaeraLerroa Lerroaren IDa
 */
 /**
 * Eskaera lerro bat ezabatzen du.
 *
 * @param idEskaeraLerroa Lerroaren IDa.
 * @throws SQLException Errorea ezabatzean.
 */
 public void eskaeraLerroaEzabatu(int idEskaeraLerroa) throws SQLException {
 EskaeraLerroa.eskaeraLerroaEzabatu(idEskaeraLerroa);
 }

 /**
 * Produktuari prezioa aldatu (produktuariPrezioaJarri-ren berdina).
 *
 * @param idProduktua Produktuaren IDa
 * @param prezioa Prezio berria
 */
 /**
 * Produktuari prezioa aldatzen dio (produktuariPrezioaJarri-ren berdina).
 *
 * @param idProduktua Produktuaren IDa.
 * @param prezioa Prezio berria.
 * @throws Exception Errorea eguneratzean.
 */
 public void produktuariPrezioaAldatu(int idProduktua, BigDecimal prezioa)
throws Exception {
 produktuariPrezioaJarri(idProduktua, prezioa);
 }

 // -----
 // ESKAEREN KUDEAKETA
 // -----
```

  

```
 /**
 * Eskaeren informazioa ikusi (bezero zehatz batentzako).
 *
 * @param idBezeroa Bezeroaren IDa
 * @return Eskaera zerrenda
 */
 /**
 * Bezero baten eskaerak ikusten ditu.
 *
 * @param idBezeroa Bezeroaren IDa.
 * @return Eskaera zerrenda.
```

```

* @throws Exception Errorea lortzean.
*/
public List<Eskaera> eskaerakIkusi(int idBezeroa) throws Exception {
 List<Eskaera> eskaerak = new ArrayList<>();
 String sql = "SELECT * FROM eskaerak WHERE bezeroa_id=?";

 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idBezeroa);
 try (ResultSet rs = pst.executeQuery()) {
 while (rs.next()) {
 Object langileaIdObj = rs.getObject("langilea_id");
 Integer langileaId = (langileaIdObj != null) ? ((Number) langileaIdObj).intValue() : null;

 eskaerak.add(new Eskaera(
 rs.getInt("id_eskaera"),
 rs.getInt("bezeroa_id"),
 langileaId,
 rs.getTimestamp("data"),
 rs.getTimestamp("eguneratze_data"),
 rs.getBigDecimal("guztira_prezioa"),
 rs.getString("faktura_zenbakia"),
 rs.getString("faktura_url"),
 rs.getString("eskaera_egoera")));
 }
 }
 }
 return eskaerak;
}

/**
 * Eskaera bat ikusi.
 *
 * @param idEskaera Eskaeraren IDa
 * @return Eskaera objektua
 * @throws Exception
 */
public Eskaera eskaeraIkusi(int idEskaera) throws Exception {
 Eskaera eskaera = null;
 String sql = "SELECT * FROM eskaerak WHERE id_eskaera=?";

 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idEskaera);
 try (ResultSet rs = pst.executeQuery()) {
 if (rs.next()) {
 Object langileaIdObj = rs.getObject("langilea_id");
 Integer langileaId = (langileaIdObj != null) ? ((Number) langileaIdObj).intValue() : null;

 eskaera = new Eskaera(
 rs.getInt("id_eskaera"),
 rs.getInt("bezeroa_id"),
 langileaId,
 rs.getTimestamp("data"),
 rs.getTimestamp("eguneratze_data"),
 rs.getString("faktura_zenbakia"),
 rs.getString("faktura_url"),
 rs.getString("eskaera_egoera"));
 }
 }
 }
 return eskaera;
}

```

```

 rs.getBigDecimal("guztira_presioa"),
 rs.getString("faktura_zenbakia"),
 rs.getString("faktura_url"),
 rs.getString("eskaera_egoera")));
 }
}
return eskaera;
}

/**
 * Eskaera osoa (eskaera + lerroak) sortzen du transakzio bakarrean.
 *
 * @param e Eskaera objektua.
 * @param lerroak Eskaera lerroen zerrenda.
 * @return Sortutako eskaeraren IDa.
 * @throws SQLException Errorea datu-basean.
 */
public int eskaeraOsoaSortu(Eskaera e, List<EskaeraLerroa> lerroak) throws
SQLException {
 Connection konexioa = null;
 int idEskaera = -1;
 try {
 konexioa = DB_Konexioa.konektatu();
 konexioa.setAutoCommit(false);

 String sqlEskaera = "INSERT INTO eskaerak (bezeroa_id, langilea_id,
data, guztira_presioa, eskaera_egoera) VALUES (?, ?, ?, NOW(), ?, ?)";
 try (PreparedStatement pst = konexioa.prepareStatement(sqlEskaera,
Statement.RETURN_GENERATED_KEYS)) {
 pst.setInt(1, e.getBezeroaId());
 if (e.getLangileaId() != null) {
 pst.setInt(2, e.getLangileaId());
 } else {
 pst.setNull(2, java.sql.Types.INTEGER);
 }
 pst.setBigDecimal(3, e.getGuztiraPresioa());
 pst.setString(4, e.getEskaeraEgoera());
 pst.executeUpdate();

 ResultSet rs = pst.getGeneratedKeys();
 if (rs.next()) {
 idEskaera = rs.getInt(1);
 } else {
 throw new SQLException("Ez da eskaera IDrik sortu.");
 }
 }

 String sqlLerroa = "INSERT INTO eskaera_lerroak (eskaera_id,
produktua_id, kantitatea, unitate_presioa, eskaera_lerro_egoera) VALUES (?, ?, ?, ?, ?)";
 try (PreparedStatement pstLerroa =
konexioa.prepareStatement(sqlLerroa)) {
 for (EskaeraLerroa l : lerroak) {
 pstLerroa.setInt(1, idEskaera);
 pstLerroa.setInt(2, l.getProduktuaId());
 pstLerroa.setInt(3, l.getKantitatea());
 }
 }
 }
}
```

```

 pstLerroa.setBigDecimal(4, l.getUnitatePrezioa());
 pstLerroa.setString(5, e.getEskaeraEgoera());
 pstLerroa.addBatch();
 }
 pstLerroa.executeBatch();
}

konexioa.commit();
return idEskaera;
} catch (SQLException ex) {
 if (konexioa != null)
 konexioa.rollback();
 throw ex;
} finally {
 if (konexioa != null) {
 konexioa.setAutoCommit(true);
 konexioa.close();
 }
}
}

/**
 * Eskaera osoa (eskaera + lerroak) editatzen du transakzio bakarrean.
 * Lerro zaharrak ezabatu eta berriak sartzen ditu.
 *
 * @param e Eskaera objektua (id-a barne).
 * @param lerroak Eskaera lerroen zerrenda berria.
 * @throws SQLExceptionErrorea datu-basean.
 */
public void eskaeraOsoaEditatu(Eskaera e, List<EskaeraLerroa> lerroak) throws
SQLException {
 Connection konexioa = null;
 try {
 konexioa = DB_Konexioa.konektatu();
 konexioa.setAutoCommit(false);

 // 1. Eskaera eguneratu
 String sqlUpdate = "UPDATE eskaerak SET bezeroa_id = ?, guztira_prezioa
= ?, eskaera_egoera = ?, eguneratze_data = NOW() WHERE id_eskaera = ?";
 try (PreparedStatement pst = konexioa.prepareStatement(sqlUpdate)) {
 pst.setInt(1, e.getBezeroaId());
 pst.setBigDecimal(2, e.getGuztiraPrezioa());
 pst.setString(3, e.getEskaeraEgoera());
 pst.setInt(4, e.getIdEskaera());
 pst.executeUpdate();
 }

 // 2. Lerro zaharrak ezabatu
 String sqlDeleteLerroak = "DELETE FROM eskaera_lerroak WHERE eskaera_id
= ?";
 try (PreparedStatement pstDelete =
konexioa.prepareStatement(sqlDeleteLerroak)) {
 pstDelete.setInt(1, e.getIdEskaera());
 pstDelete.executeUpdate();
 }

 // 3. Lerro berriak sartu
 }
}

```

```

 String sqlInsertLerroa = "INSERT INTO eskaera_lerroak (eskaera_id, produktua_id, kantitatea, unitate_preszia, eskaera_lerro_egoera) VALUES (?, ?, ?, ?, ?)";
 try (PreparedStatement pstLerroa =
konexioa.prepareStatement(sqlInsertLerroa)) {
 for (EskaeraLerroa l : lerroak) {
 pstLerroa.setInt(1, e.getIdEskaera());
 pstLerroa.setInt(2, l.getProduktuaId());
 pstLerroa.setInt(3, l.getKantitatea());
 pstLerroa.setBigDecimal(4, l.getUnitatePreszia());
 pstLerroa.setString(5, e.getEskaeraEgoera());
 pstLerroa.addBatch();
 }
 pstLerroa.executeBatch();
 }

 konexioa.commit();
 } catch (SQLException ex) {
 if (konexioa != null)
 konexioa.rollback();
 throw ex;
 } finally {
 if (konexioa != null) {
 konexioa.setAutoCommit(true);
 konexioa.close();
 }
 }
}

/**
 * Eskaera ezabatu.
 *
 * @param idEskaera Eskaeraren IDa
 */
/**
 * Eskaera bat ezabatzen du.
 *
 * @param idEskaera Eskaeraren IDa.
 * @throws Exception Errorea ezabatzean.
 */
public void eskaeraEzabatu(int idEskaera) throws Exception {
 Connection konexioa = null;
 try {
 konexioa = DB_Konexioa.konektatu();
 konexioa.setAutoCommit(false);

 // 1. Eskaera lerroak ezabatu (Foreign Key constraint dela eta)
 String sqlLerroak = "DELETE FROM eskaera_lerroak WHERE eskaera_id = ?";
 try (PreparedStatement pstLerroak =
konexioa.prepareStatement(sqlLerroak)) {
 pstLerroak.setInt(1, idEskaera);
 pstLerroak.executeUpdate();
 }

 // 2. Eskaera bera ezabatu
 String sqlEskaera = "DELETE FROM eskaerak WHERE id_eskaera = ?";
 try (PreparedStatement pstEskaera =

```

```

konexioa.prepareStatement(sqlEskaera)) {
 pstEskaera.setInt(1, idEskaera);
 pstEskaera.executeUpdate();
}

 konexioa.commit();
} catch (SQLException ex) {
 if (konexioa != null) {
 konexioa.rollback();
 }
 throw ex;
} finally {
 if (konexioa != null) {
 konexioa.setAutoCommit(true);
 konexioa.close();
 }
}
}
}
}

```

## Sarrera.java

```

package model;

import java.sql.Timestamp;

/**
 * Sarrera klasea.
 * Produktuen sarrerak (hornitzaleengandik) kudeatzeko klasea.
 * Identifikatzaileak, datak eta egoera gordetzen ditu.
 */
public class Sarrera {
 private int idSarrera;
 private Timestamp data;
 private Timestamp eguneratzeData;
 private int hornitzaleaId;
 private int langileaId;
 private String sarreraEgoera;

 /**
 * Sarrera eraikitzailea.
 *
 * @param idSarrera Sarreraren IDa.
 * @param data Sarrera data.
 * @param eguneratzeData Eguneratze data.
 * @param hornitzaleaId Hornitzalearen IDa.
 * @param langileaId Langilearen IDa (sarrera kudeatu duena).
 * @param sarreraEgoera Sarreraren egoera.
 */
 public Sarrera(int idSarrera, Timestamp data, Timestamp eguneratzeData, int
hornitzaleaId, int langileaId,

```

```
 String sarreraEgoera) {
 this.idSarrera = idSarrera;
 this.data = data;
 this.eguneratzeData = eguneratzeData;
 this.hornitzaireId = hornitzaireId;
 this.langileaId = langileaId;
 this.sarreraEgoera = sarreraEgoera;
}

/**
 * Sarreraren IDa lortzen du.
 *
 * @return IDa.
 */
public int getIdSarrera() {
 return idSarrera;
}

public void setIdSarrera(int idSarrera) {
 this.idSarrera = idSarrera;
}

/**
 * Sarrera data lortzen du.
 *
 * @return Data.
 */
public Timestamp getData() {
 return data;
}

public void setData(Timestamp data) {
 this.data = data;
}

/**
 * Eguneratze data lortzen du.
 *
 * @return Eguneratze data.
 */
public Timestamp getEguneratzeData() {
 return eguneratzeData;
}

public void setEguneratzeData(Timestamp eguneratzeData) {
 this.eguneratzeData = eguneratzeData;
}

/**
 * Hornitzairearen IDa lortzen du.
 *
 * @return Hornitzairearen IDa.
 */
public int getHornitzaireId() {
 return hornitzaireId;
}
```

```

public void setHornitzaireaId(int hornitzaireaId) {
 this.hornitzaireaId = hornitzaireaId;
}

/**
 * Langilearen IDa lortzen du.
 *
 * @return Langilearen IDa.
 */
public int getLangileId() {
 return langileId;
}

public void setLangileId(int langileId) {
 this.langileId = langileId;
}

/**
 * Sarreraren egoera lortzen du.
 *
 * @return Egoera.
 */
public String getSarreraEgoera() {
 return sarreraEgoera;
}

public void setSarreraEgoera(String sarreraEgoera) {
 this.sarreraEgoera = sarreraEgoera;
}
}

```

## SarreraLerroa.java

```

package model;

/**
 * SarreraLerroa klasea.
 * Sarrera baten lerro bakoitza irudikatzen du (produktu bat eta kantitatea).
 */
public class SarreraLerroa {
 private int idSarreraLerroa;
 private int sarreraId;
 private int produktuaId;
 private int kantitatea;
 private String sarreraLerroEgoera;

 /**
 * SarreraLerroa eraikitzailea.
 *
 * @param idSarreraLerroa Lerroaren IDa.
 * @param sarreraId Sarreraren IDa.
 */
 public SarreraLerroa(int idSarreraLerroa, int sarreraId, int produktuaId, int kantitatea, String sarreraLerroEgoera) {
 this.idSarreraLerroa = idSarreraLerroa;
 this.sarreraId = sarreraId;
 this.produktuaId = produktuaId;
 this.kantitatea = kantitatea;
 this.sarreraLerroEgoera = sarreraLerroEgoera;
 }
}

```

```
* @param produktuaId Produktuaren IDa.
* @param kantitatea Kantitatea.
* @param sarreraLerroEgoera Lerroaren egoera.
*/
public SarreraLerroa(int idSarreraLerroa, int sarreraId, int produktuaId, int
kantitatea,
 String sarreraLerroEgoera) {
 this.idSarreraLerroa = idSarreraLerroa;
 this.sarreraId = sarreraId;
 this.produktuaId = produktuaId;
 this.kantitatea = kantitatea;
 this.sarreraLerroEgoera = sarreraLerroEgoera;
}

/**
 * Lerroaren IDa lortzen du.
 *
 * @return IDa.
 */
public int getIdSarreraLerroa() {
 return idSarreraLerroa;
}

public void setIdSarreraLerroa(int idSarreraLerroa) {
 this.idSarreraLerroa = idSarreraLerroa;
}

/**
 * Sarreraren IDa lortzen du.
 *
 * @return Sarreraren IDa.
 */
public int getSarreraId() {
 return sarreraId;
}

public void setSarreraId(int sarreraId) {
 this.sarreraId = sarreraId;
}

/**
 * Produktuaren IDa lortzen du.
 *
 * @return Produktuaren IDa.
 */
public int getProduktuaId() {
 return produktuaId;
}

public void setProduktuaId(int produktuaId) {
 this.produktuaId = produktuaId;
}

/**
 * Kantitatea lortzen du.
 *
 * @return Kantitatea.
*/
```

```

 */
 public int getKantitatea() {
 return kantitatea;
 }

 public void setKantitatea(int kantitatea) {
 this.kantitatea = kantitatea;
 }

 /**
 * Lerroaren egoera lortzen du.
 *
 * @return Egoera.
 */
 public String getSarreraLerroEgoera() {
 return sarreraLerroEgoera;
 }

 public void setSarreraLerroEgoera(String sarreraLerroEgoera) {
 this.sarreraLerroEgoera = sarreraLerroEgoera;
 }
}

```

## Softwarea.java

```

package model;

import java.math.BigDecimal;
import java.sql.Timestamp;

/**
 * Softwarea klasea.
 * Produktua klasaren azpiklasea da eta software motako produktuen ezaugarriak
 * gordetzen ditu.
 * Adibidez: lizentzia mota, bertsioa, garatzailea.
 */
public class Softwarea extends Produktua {
 private String softwareMota;
 private String lizentziaMota;
 private String bertsioa;
 private String garatzailea;

 /**
 * Softwarea eraikitzailea.
 *
 * @param idProduktua Produktuaren IDa.
 * @param hornitzaireId Hornitzairearen IDa.
 * @param kategoriaId Kategoriaren IDa.
 * @param izena Izena.
 * @param marka Marka.
 * @param mota Mota.
 */
}

```

```

* @param deskribapena Deskribapena.
* @param irudiaUrl Irudiaren URLa.
* @param biltegiId Biltegiaren IDa.
* @param produktuEgoera Produktuaren egoera.
* @param produktuEgoeraOharra Egoeraren oharra.
* @param salgai Salgai dagoen.
* @param salmentaPrezioa Salmenta prezioa.
* @param stock Stock kopurua.
* @param eskaintza Eskaintza prezioa.
* @param zergakEhunekoa Zergen ehunekoa.
* @param sortzeData Sortze data.
* @param eguneratzeData Eguneratze data.
* @param softwareMota Software mota (adib. Sistema eragilea,
* Bulegotika...).
* @param lizentziaMota Lizentzia mota.
* @param bertsioa Bertsioa.
* @param garatzailea Garatzailea.
*/
public Software(int idProduktua, int hornitzaireId, int kategoriaId, String
izen, String marka, String mota,
 String deskribapena, String irudiaUrl, Integer biltegiId, String
produktuEgoera,
 String produktuEgoeraOharra, boolean salgai, BigDecimal
salmentaPrezioa, int stock,
 BigDecimal eskaintza, BigDecimal zergakEhunekoa, Timestamp sortzeData,
Timestamp eguneratzeData,
 String softwareMota, String lizentziaMota, String bertsioa, String
garatzailea) {
 super(idProduktua, hornitzaireId, kategoriaId, izen, marka, mota,
deskribapena, irudiaUrl, biltegiId,
 produktuEgoera, produktuEgoeraOharra, salgai, salmentaPrezioa,
stock, eskaintza, zergakEhunekoa,
 sortzeData, eguneratzeData);
 this.softwareMota = softwareMota;
 this.lizentziaMota = lizentziaMota;
 this.bertsioa = bertsioa;
 this.garatzailea = garatzailea;
}

/**
 * Software mota lortzen du.
 *
 * @return Software mota.
 */
public String getSoftwareMota() {
 return softwareMota;
}

/**
 * Software mota ezartzen du.
 *
 * @param softwareMota Mota berria.
 */
public void setSoftwareMota(String softwareMota) {
 this.softwareMota = softwareMota;
}

```

```
/**
 * Lizentzia mota lortzen du.
 *
 * @return Lizentzia mota.
 */
public String getLizentziaMota() {
 return lizentziaMota;
}

/**
 * Lizentzia mota ezartzen du.
 *
 * @param lizentziaMota Lizentzia mota berria.
 */
public void setLizentziaMota(String lizentziaMota) {
 this.lizentziaMota = lizentziaMota;
}

/**
 * Bertsioa lortzen du.
 *
 * @return Bertsioa.
 */
public String getBertsioa() {
 return bertsioa;
}

/**
 * Bertsioa ezartzen du.
 *
 * @param bertsioa Bertsio berria.
 */
public void setBertsioa(String bertsioa) {
 this.bertsioa = bertsioa;
}

/**
 * Garatzailea lortzen du.
 *
 * @return Garatzailea.
 */
public String getGaratzailea() {
 return garatzailea;
}

/**
 * Garatzailea ezartzen du.
 *
 * @param garatzailea Garatzaile berria.
 */
public void setGaratzailea(String garatzailea) {
 this.garatzailea = garatzailea;
}
}
```

# Tableta.java

```
package model;

import java.math.BigDecimal;
import java.sql.Timestamp;

/**
 * Tableta klasea.
 * Produktua klasaren azpiklasea da eta tableta motako produktuen ezaugarriak
 * gordetzen ditu.
 * Adibidez: pantaila, biltegiratzea, konektibitatea...
 */
public class Tableta extends Produktua {
 private BigDecimal pantailaHazbeteak;
 private int biltegiratzeaGb;
 private String konektibitatea;
 private String sistemaEragilea;
 private int bateriaMah;
 private boolean arkatzarekinBateragarria;

 /**
 * Tableta eraikitzalea.
 *
 * @param idProduktua Produktuaren IDa.
 * @param hornitzaireId Hornitzairearen IDa.
 * @param kategoriaId Kategoriaren IDa.
 * @param izena Izena.
 * @param marka Marka.
 * @param mota Mota.
 * @param deskribapena Deskribapena.
 * @param irudiaUrl Irudiaren URLa.
 * @param biltegiId Biltegiaren IDa.
 * @param produktuEgoera Produktuaren egoera.
 * @param produktuEgoeraOharra Egoeraren oharra.
 * @param salgai Salgai dagoen.
 * @param salmentaPrezioa Salmenta prezioa.
 * @param stock Stock kopurua.
 * @param eskaintza Eskaintza prezioa.
 * @param zergakEhunekoa Zergen ehunekoa.
 * @param sortzeData Sortze data.
 * @param eguneratzeData Eguneratze data.
 * @param pantailaHazbeteak Pantailaren hazbeteak.
 * @param biltegiratzeaGb Biltegiratzea GBtan.
 * @param konektibitatea Konektibitatea (WiFi, 4G...).
 * @param sistemaEragilea Sistema eragilea.
 * @param bateriaMah Bateriaren edukiera (mAh).
 * @param arkatzarekinBateragarria Arkatzarekin bateragarria den.
 */
 public Tableta(int idProduktua, int hornitzaireId, int kategoriaId, String
izena, String marka, String mota,
 String deskribapena, String irudiaUrl, Integer biltegiId, String
produktuEgoera,
 String produktuEgoeraOharra, boolean salgai, BigDecimal
```

```
salmentaPrezioa, int stock,
 BigDecimal eskaintza, BigDecimal zergakEhuneko, Timestamp sortzeData,
Timestamp eguneratzeData,
 BigDecimal pantailaHazbeteak, int biltegiratzeGb, String
konektibitatea,
 String sistemaEragilea, int bateriaMah, boolean
arkatzarekinBateragarria) {
 super(idProduktua, hornitzaleId, kategoriaId, izena, marka, mota,
deskribapena, irudiaUrl, biltegiId,
 produktuEgoera, produktuEgoera0harra, salgai, salmentaPrezioa,
stock, eskaintza, zergakEhuneko,
 sortzeData, eguneratzeData);
 this.pantailaHazbeteak = pantailaHazbeteak;
 this.biltegiratzeGb = biltegiratzeGb;
 this.konektibitatea = konektibitatea;
 this.sistemaEragilea = sistemaEragilea;
 this.bateriaMah = bateriaMah;
 this.arkatzarekinBateragarria = arkatzarekinBateragarria;
}

/**
 * Pantailaren hazbeteak lortzen ditu.
 *
 * @return Hazbeteak.
 */
public BigDecimal getPantailaHazbeteak() {
 return pantailaHazbeteak;
}

/**
 * Pantailaren hazbeteak ezartzen ditu.
 *
 * @param pantailaHazbeteak Hazbete berriak.
 */
public void setPantailaHazbeteak(BigDecimal pantailaHazbeteak) {
 this.pantailaHazbeteak = pantailaHazbeteak;
}

/**
 * Biltegiratza lortzen du GBtan.
 *
 * @return Biltegiratza.
 */
public int getBiltegiratzeGb() {
 return biltegiratzeGb;
}

/**
 * Biltegiratza ezartzen du.
 *
 * @param biltegiratzeGb Biltegiratze berria.
 */
public void setBiltegiratzeGb(int biltegiratzeGb) {
 this.biltegiratzeGb = biltegiratzeGb;
}

/**
```

```
* Konektibitatea lortzen du.
*
* @return Konektibitatea.
*/
public String getKonektibitatea() {
 return konektibitatea;
}

/***
* Konektibitatea ezartzen du.
*
* @param konektibitatea Konektibitate berria.
*/
public void setKonektibitatea(String konektibitatea) {
 this.konektibitatea = konektibitatea;
}

/***
* Sistema eragilea lortzen du.
*
* @return Sistema eragilea.
*/
public String getSistemaEragilea() {
 return sistemaEragilea;
}

/***
* Sistema eragilea ezartzen du.
*
* @param sistemaEragilea Sistema eragile berria.
*/
public void setSistemaEragilea(String sistemaEragilea) {
 this.sistemaEragilea = sistemaEragilea;
}

/***
* Bateria edukiera lortzen du mAh-tan.
*
* @return Bateria.
*/
public int getBateriaMah() {
 return bateriaMah;
}

/***
* Bateria edukiera ezartzen du.
*
* @param bateriaMah Bateria berria.
*/
public void setBateriaMah(int bateriaMah) {
 this.bateriaMah = bateriaMah;
}

/***
* Arkatzarekin bateragarria den lortzen du.
*
* @return True bateragarria bada.
*/
```

```

 */
 public boolean isArkatzarekinBateragarria() {
 return arkatzarekinBateragarria;
 }

 /**
 * Arkatzarekin bateragarria den ezartzen du.
 *
 * @param arkatzarekinBateragarria Bateragarritasuna.
 */
 public void setArkatzarekinBateragarria(boolean arkatzarekinBateragarria) {
 this.arkatzarekinBateragarria = arkatzarekinBateragarria;
 }
}

```

## TeknikariLangilea.java

```

package model;

import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;
import java.util.ArrayList;
import java.util.List;

import db.DB_Konexioa;

/**
 * TeknikariLangilea klasea.
 * Langilea klasearen azpiklasea da, eta teknikari arloko langileen
 * funtzionalitateak kudeatzen ditu.
 * Produktuen mantenua, konponketak eta akatsen kudeaketa egiteko metodoak ditu.
 */
public class TeknikariLangilea extends Langilea {

 /**
 * TeknikariLangilea eraikitzailea.
 * Langilea objektu batetik abiatuta sortzen da.
 *
 * @param l Langilea objektua.
 */
 public TeknikariLangilea(Langilea l) {
 super(l.getIdLangilea(), l.getIzena(), l.getAbizena(), l.getNan(),
l.getJaiotzaData(), l.getHerriaId(),
 l.getHelbidea(), l.getPostaKodea(), l.getTelefonoa(),
l.getEmaila(), l.getHizkuntza(),
 l.getPasahitza(), l.getSaltoTxartelaUid(), l.getAltaData(),
l.getEguneratzeData(),

```

```

 l.isAktibo(), l.getSailaId(), l.getIban(), l.getKurrikuluma());
 }

 // -----
 // PRODUKTUEN KUDEAKETA
 // -----

 /**
 * Biltegira iritsi diren produktu guztiak bistaratu (salgai daudenak eta ez
 * daudenak).
 *
 * @return Produktu guztien zerrenda.
 */
 /**
 * Biltegira iritsi diren produktu guztiak bistaratu (salgai daudenak eta ez
 * daudenak).
 *
 * @return Produktu guztien zerrenda.
 * @throws SQLException Errorea irakurtzean.
 */
 public List<Produktua> produktuakIkusi() throws SQLException {
 List<Produktua> produktuak = new ArrayList<>();
 String sql = "SELECT * FROM produktuak";

 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql);
 ResultSet rs = pst.executeQuery()) {

 while (rs.next()) {
 // Produktua abstract denez, klase anonimo bat erabiltzen dugu
 instantziatzeko
 // edo base-datuak soilik kargatzeko.
 Produktua p = new Produktua(
 rs.getInt("id_produktua"),
 rs.getInt("hornitzale_id"),
 rs.getInt("kategoria_id"),
 rs.getString("izena"),
 rs.getString("marka"),
 rs.getString("mota"),
 rs.getString("deskribapena"),
 rs.getString("irudia_url"),
 rs.getInt("biltegi_id"), // Nullable izan daiteke
 rs.getString("produktu_egoera"),
 rs.getString("produktu_egoera_oharra"),
 rs.getBoolean("salgai"),
 rs.getBigDecimal("salmenta_prezioa"),
 rs.getInt("stock"),
 rs.getBigDecimal("eskaintza"),
 rs.getBigDecimal("zergak_ehunekoa"),
 rs.getTimestamp("sortze_data"),
 rs.getTimestamp("eguneratze_data")) {
 }; // Klase anonimoa
 produktuak.add(p);
 }
 }
 return produktuak;
 }
}

```

```

/**
 * Produktu bat editatu (egoera eta salgai).
 *
 * @param idProduktua Produktuaren IDa.
 * @param salgai Salgai dagoen ala ez.
 * @param egoera Produktuaren egoera ('Berria', 'Berritua A', 'Berritua
B',
 * 'Hondatua', 'Zehazteko').
 */
/***
 * Produktu bat editatu (egoera eta salgai).
 *
 * @param idProduktua Produktuaren IDa.
 * @param salgai Salgai dagoen ala ez.
 * @param egoera Produktuaren egoera ('Berria', 'Berritua A', 'Berritua
B',
 * 'Hondatua', 'Zehazteko').
 * @throws SQLException Errorea editatzean.
 */
public void produktuaEditatu(int idProduktua, boolean salgai, String egoera)
throws SQLException {
 String sql = "UPDATE produktuak SET salgai = ?, produktu_egoera = ? WHERE
id_produktua = ?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setBoolean(1, salgai);
 pst.setString(2, egoera);
 pst.setInt(3, idProduktua);
 pst.executeUpdate();
 }
}

/***
 * Produktuari irudia gehitu.
 *
 * @param idProduktua Produktuaren IDa.
 * @param irudiaUrl Irudiaren izena edo bidea (jpg).
 */
/***
 * Produktuari irudia gehitu.
 *
 * @param idProduktua Produktuaren IDa.
 * @param irudiaUrl Irudiaren izena edo bidea (jpg).
 * @throws SQLException Errorea egunerezan.
 */
public void produktuariIrudiaGehitu(int idProduktua, String irudiaUrl) throws
SQLException {
 String sql = "UPDATE produktuak SET irudia_url = ? WHERE id_produktua = ?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setString(1, irudiaUrl);
 pst.setInt(2, idProduktua);
 pst.executeUpdate();
 }
}

```

```

/**
 * Produktuari prezioa eta eskaintza ezerri.
 *
 * @param idProduktua Produktuaren IDa.
 * @param prezioa Salmenta prezioa.
 * @param eskaintza Eskaintza prezioa (baldin badago).
 */
/***
 * Produktuari prezioa eta eskaintza ezerri.
 *
 * @param idProduktua Produktuaren IDa.
 * @param prezioa Salmenta prezioa.
 * @param eskaintza Eskaintza prezioa (baldin badago).
 * @throws SQLException Errorea eguneraztzean.
 */
public void prezioaEzarri(int idProduktua, BigDecimal prezioa, BigDecimal
eskaintza) throws SQLException {
 String sql = "UPDATE produktuak SET salmenta_prezioa = ?, eskaintza = ?
WHERE id_produktua = ?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setBigDecimal(1, prezioa);
 pst.setBigDecimal(2, eskaintza);
 pst.setInt(3, idProduktua);
 pst.executeUpdate();
 }
}

/***
 * Produktu bat sistematik ezabatu.
 *
 * @param idProduktua Produktuaren IDa.
 */
/***
 * Produktu bat sistematik ezabatu.
 *
 * @param idProduktua Produktuaren IDa.
 * @throws SQLException Errorea ezabatzean.
 */
public void produktuaBorratu(int idProduktua) throws SQLException {
 String sql = "DELETE FROM produktuak WHERE id_produktua = ?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idProduktua);
 pst.executeUpdate();
 }
}

// -----
// KONPONKETEN KUDEAKETA
// -----

/***
 * Konponketa guztiak bistaratu.
 *
 * @return Konponketa zerrenda.
 */

```

```

/**
 * Konponketa guztiak bistaratu.
 *
 * @return Konponketa zerrenda.
 * @throws SQLException Errorea irakurtzean.
 */
public List<Konponketa> konponketakIkusi() throws SQLException {
 List<Konponketa> konponketak = new ArrayList<>();
 String sql = "SELECT * FROM konponketak";

 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql);
 ResultSet rs = pst.executeQuery()) {

 while (rs.next()) {
 Konponketa k = new Konponketa(
 rs.getInt("id_konponketa"),
 rs.getInt("produktua_id"),
 rs.getInt("langilea_id"),
 rs.getTimestamp("hasiera_data"),
 rs.getTimestamp("amaiera_data"),
 rs.getString("konponketa_egoera"),
 rs.getInt("akatsa_id"),
 rs.getString("oharrak"),
 rs.getTimestamp("egunearatze_data"));
 konponketak.add(k);
 }
 }
 return konponketak;
}

/**
 * Konponketa berri bat sortu.
 *
 * @param k Konponketa objektua.
 */
/**
 * Konponketa berri bat sortu.
 *
 * @param k Konponketa objektua.
 * @throws SQLException Errorea sortzean.
 */
public void konponketaEgin(Konponketa k) throws SQLException {
 String sql = "INSERT INTO konponketak (produktua_id, langilea_id,
hasiera_data, konponketa_egoera, akatsa_id, oharrak) VALUES (?, ?, ?, ?, ?, ?)";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, k.getProduktuaId());
 pst.setInt(2, k.getLangileaId());
 pst.setTimestamp(3, k.getHasieraData());
 pst.setString(4, k.getKonponketaEgoera());
 pst.setInt(5, k.getAkatsaId());
 pst.setString(6, k.getOharrak());
 pst.executeUpdate();
 }
}

```

```

/**
 * Sortuta dagoen konponketa baten datuak aldatu.
 *
 * @param k Konponketa objektua (id-a barne).
 */
/**
 * Sortuta dagoen konponketa baten datuak aldatu.
 *
 * @param k Konponketa objektua (id-a barne).
 * @throws SQLExceptionErrorea editatzean.
 */
public void konponketaEditatu(Konponketa k) throws SQLException {
 String sql = "UPDATE konponketak SET produktua_id = ?, langilea_id = ?, hasiera_data = ?, amaiera_data = ?, konponketa_egoera = ?, akatsa_id = ?, oharrak = ? WHERE id_konponketa = ?";
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, k.getProduktuaId());
 pst.setInt(2, k.getLangileaId());
 pst.setTimestamp(3, k.getHasieraData());
 pst.setTimestamp(4, k.getAmaieraData());
 pst.setString(5, k.getKonponketaEgoera());
 pst.setInt(6, k.getAkatsaId());
 pst.setString(7, k.getOharrak());
 pst.setInt(8, k.getIdKonponketa());
 pst.executeUpdate();
 }
 }
}

/**
 * Konponketa jakin bat ezabatu.
 *
 * @param idKonponketa Konponketaren IDa.
 */
/**
 * Konponketa jakin bat ezabatu.
 *
 * @param idKonponketa Konponketaren IDa.
 * @throws SQLExceptionErrorea ezabatzean.
 */
public void konponketaEzabatu(int idKonponketa) throws SQLException {
 String sql = "DELETE FROM konponketak WHERE id_konponketa = ?";
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idKonponketa);
 pst.executeUpdate();
 }
 }
}

/**
 * Produktu berri bat sortu (Bakarrik TeknikariLangilea eta BiltegiLangilea-k
 * egin dezakete).
 */
/**
 * Produktu berri bat sortu (Bakarrik TeknikariLangilea eta BiltegiLangilea-k
 * egin dezakete).
 */

```

```

* @param p Produktua
* @throws SQLExceptionErrorea sortzean.
*/
public void produktuBatSortu(Produktua p) throws SQLException {
 // Oharra: Metodo honek produktu bat sortzen du 'produktuak' taulan.
 // Ez du stock sarrerarik gestionatzen (hori BiltegiLangileak egiten du
 // Sarrera bidez).
 String sql = "INSERT INTO produktuak (izena, marka, kategoria_id, mota,
biltegi_id, hornitzaire_id, stock, produktu_egoera, deskribapena, irudia_url,
produktu_egoera_oharra, salgai, salmenta_prezioa, zergak_ehunekoa) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

 try (Connection kon = DB_Konexioa.konektatu();
 PreparedStatement pst = kon.prepareStatement(sql)) {

 pst.setString(1, p.getIzena());
 pst.setString(2, p.getMarka());
 pst.setInt(3, p.getKategoriaId());
 pst.setString(4, p.getMota());
 if (p.getBiltegiId() != null) {
 pst.setInt(5, p.getBiltegiId());
 } else {
 pst.setNull(5, Types.INTEGER);
 }
 pst.setInt(6, p.getHornitzaireId());
 pst.setInt(7, p.getStock());
 pst.setString(8, p.getProduktuEgoera());
 pst.setString(9, p.getDeskribapena());
 pst.setString(10, p.getIrudiaUrl());
 pst.setString(11, p.getProduktuEgoeraOharra());
 pst.setBoolean(12, p.isSalgai());
 pst.setBigDecimal(13, p.getSalmentaPrezioa());
 pst.setBigDecimal(14, p.getZergakEhunekoa());

 pst.executeUpdate();
 }
}

// -----
// AKATSAK KUDEAKETA
// -----

/**
 * Akatsen informazioa ikusi.
 *
 * @return Akatsen zerrenda.
 */
/**
 * Akatsen informazioa ikusi.
 *
 * @return Akatsen zerrenda.
 * @throws SQLException Errorea irakurtzean.
 */
public List<Akatsa> akatsaIkusi() throws SQLException {
 List<Akatsa> akatsak = new ArrayList<>();
 String sql = "SELECT * FROM akatsak";

 try (Connection konexioa = DB_Konexioa.konektatu();

```

```

 PreparedStatement pst = konexioa.prepareStatement(sql);
 ResultSet rs = pst.executeQuery()) {

 while (rs.next()) {
 Akatsa a = new Akatsa(
 rs.getInt("id_akatsa"),
 rs.getString("izena"),
 rs.getString("deskribapena"));
 akatsak.add(a);
 }
 }
 return akatsak;
 }

 /**
 * Akats berri bat sortu.
 *
 * @param a Akatsa objektua.
 */
 /**
 * Akats berri bat sortu.
 *
 * @param a Akatsa objektua.
 * @throws SQLExceptionErrorea sortzean.
 */
 public void akatsaSortu(Akatsa a) throws SQLException {
 String sql = "INSERT INTO akatsak (izena, deskribapena) VALUES (?, ?)";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setString(1, a.getIzena());
 pst.setString(2, a.getDeskribapena());
 pst.executeUpdate();
 }
 }

 /**
 * Akats bat editatu.
 *
 * @param a Akatsa objektua (id-a barne).
 */
 /**
 * Akats bat editatu.
 *
 * @param a Akatsa objektua (id-a barne).
 * @throws SQLExceptionErrorea editatzean.
 */
 public void akatsaEditatu(Akatsa a) throws SQLException {
 String sql = "UPDATE akatsak SET izena = ?, deskribapena = ? WHERE
id_akatsa = ?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setString(1, a.getIzena());
 pst.setString(2, a.getDeskribapena());
 pst.setInt(3, a.getIdAkatsa());
 pst.executeUpdate();
 }
 }
}

```

```

/**
 * Akats bat ezabatu.
 *
 * @param idAkatsa Akatsaren IDa.
 */
/**
 * Akats bat ezabatu.
 *
 * @param idAkatsa Akatsaren IDa.
 * @throws SQLException Errorea ezabatzean.
 */
public void akatsaEzabatu(int idAkatsa) throws SQLException {
 String sql = "DELETE FROM akatsak WHERE id_akatsa = ?";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 pst.setInt(1, idAkatsa);
 pst.executeUpdate();
 }
}

```

## Zerbitzaria.java

```

package model;

import java.math.BigDecimal;
import java.sql.Timestamp;

/**
 * Zerbitzaria klasea.
 * Produktua klasareen azpiklasea da eta zerbitzari motako produktuen
 * ezaugarriak gordetzen ditu.
 * Adibidez: prozesadore nukleoak, RAM mota, disko badiak...
 */
public class Zerbitzaria extends Produktua {
 private int prozesadoreNukleoak;
 private String ramMota;
 private int diskobadiak;
 private int rackUnitateak;
 private boolean elikatzeIturriErredundantea;
 private String raidKontroladora;

 /**
 * Zerbitzaria eraikitzailea.
 *
 * @param idProduktua Produktuaren IDa.
 * @param hornitzairenId Hornitzairen IDa.
 * @param kategoriaId Kategoriaren IDa.
 * @param izena Izena.
 * @param marka Marka.
 */

```

```

* @param mota Mota.
* @param deskribapena Deskribapena.
* @param irudiaUrl Irudiaren URLa.
* @param biltegiId Biltegiaren IDa.
* @param produktuEgoera Produktuaren egoera.
* @param produktuEgoeraOharra Egoeraren oharra.
* @param salgai Salgai dagoen.
* @param salmentaPrezioa Salmenta prezioa.
* @param stock Stock kopurua.
* @param eskaintza Eskaintza prezioa.
* @param zergakEhunekoa Zergen ehunekoa.
* @param sortzeData Sortze data.
* @param eguneratzeData Eguneratze data.
* @param prozesadoreNukleoak Prozesadorearen nukleo kopurua.
* @param ramMota RAM mota.
* @param diskоБadiak Disko badia kopurua.
* @param rackUnitateak Rack unitate kopurua (U).
* @param elikatzeIturriErredundantea Elikatze iturri erredundantea duen.
* @param raidKontroladora RAID kontroladora mota.
*/
public Zerbitzaria(int idProduktua, int hornitzairenId, int kategoriaId, String izena, String marka, String mota,
String deskribapena, String irudiaUrl, Integer biltegiId, String produktuEgoera,
String produktuEgoeraOharra, boolean salgai, BigDecimal salmentaPrezioa, int stock,
BigDecimal eskaintza, BigDecimal zergakEhunekoa, Timestamp sortzeData,
Timestamp eguneratzeData,
int prozesadoreNukleoak, String ramMota, int diskоБadiak, int rackUnitateak,
boolean elikatzeIturriErredundantea, String raidKontroladora) {
 super(idProduktua, hornitzairenId, kategoriaId, izena, marka, mota,
deskribapena, irudiaUrl, biltegiId,
produktuEgoera, produktuEgoeraOharra, salgai, salmentaPrezioa,
stock, eskaintza, zergakEhunekoa,
sortzeData, eguneratzeData);
 this.prozesadoreNukleoak = prozesadoreNukleoak;
 this.ramMota = ramMota;
 this.diskоБadiak = diskоБadiak;
 this.rackUnitateak = rackUnitateak;
 this.elikatzeIturriErredundantea = elikatzeIturriErredundantea;
 this.raidKontroladora = raidKontroladora;
}

/**
 * Prozesadore nukleo kopurua lortzen du.
*
* @return Nukleoak.
*/
public int getProzesadoreNukleoak() {
 return prozesadoreNukleoak;
}

/**
 * Prozesadore nukleo kopurua ezartzen du.
*
* @param prozesadoreNukleoak Nukleo berriak.

```

```
/*
public void setProzesadoreNukleoak(int prozesadoreNukleoak) {
 this.prozesadoreNukleoak = prozesadoreNukleoak;
}

/**
 * RAM mota lortzen du.
 *
 * @return RAM mota.
 */
public String getRamMota() {
 return ramMota;
}

/**
 * RAM mota ezartzen du.
 *
 * @param ramMota Mota berria.
 */
public void setRamMota(String ramMota) {
 this.ramMota = ramMota;
}

/**
 * Disko badiak lortzen ditu.
 *
 * @return Badiak.
 */
public int getDiskoBadiak() {
 return diskobadiak;
}

/**
 * Disko badiak ezartzen ditu.
 *
 * @param diskobadiak Badia berriak.
 */
public void setDiskoBadiak(int diskobadiak) {
 this.diskobadiak = diskobadiak;
}

/**
 * Rack unitateak lortzen ditu.
 *
 * @return Rack unitateak.
 */
public int getRackUnitateak() {
 return rackUnitateak;
}

/**
 * Rack unitateak ezartzen ditu.
 *
 * @param rackUnitateak Unitate berriak.
 */
public void setRackUnitateak(int rackUnitateak) {
 this.rackUnitateak = rackUnitateak;
}
```

```

}

/**
 * Elikatze iturri erredundantea duen lortzen du.
 *
 * @return True badu.
 */
public boolean isElikatzeIturriErredundantea() {
 return elikatzeIturriErredundantea;
}

/**
 * Elikatze iturri erredundantearen egoera ezartzen du.
 *
 * @param elikatzeIturriErredundantea Egoera berria.
 */
public void setElikatzeIturriErredundantea(boolean elikatzeIturriErredundantea)
{
 this.elikatzeIturriErredundantea = elikatzeIturriErredundantea;
}

/**
 * RAID kontroladora lortzen du.
 *
 * @return RAID kontroladora.
 */
public String getRaidKontroladora() {
 return raidKontroladora;
}

/**
 * RAID kontroladora ezartzen du.
 *
 * @param raidKontroladora Kontroladora berria.
 */
public void setRaidKontroladora(String raidKontroladora) {
 this.raidKontroladora = raidKontroladora;
}
}

```

## BezeroaDialog.java

```

package ui;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.sql.Date;
import java.util.List;

import javax.swing.JButton;

```

```
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;

import model.Bezeroa;
import model.Herria;
import model.SalmentaLangilea;

/**
 * BezeroaDialog klasea.
 * Bezeroak sortzeko, editatzeko eta ikusteko leihoa (JDialog).
 * Datuak sartzeko eremuak, balidazioak eta herrien kudeaketa barne hartzen
 * ditu.
 */
public class BezeroaDialog extends JDialog {

 private static final long serialVersionUID = 1L;
 private final JPanel contentPanel = new JPanel();

 private JTextField izenaField;
 private JTextField abizenaField;
 private JTextField nanField;
 private JTextField jaiotzaDataField;
 private JComboBox<String> sexuaBox;
 private JTextField txartelaField;
 private JTextField helbideaField;

 // Herria kudeaketa
 private JComboBox<Herria> herriaBox;
 private JButton herriaGehituBtn;

 private JTextField postaKodeaField;
 private JTextField telefonoaField;
 private JTextField emailField;
 private JComboBox<String> hizkuntzaBox;
 private JTextField pasahitzaField;
 private JButton okButton;

 private boolean onartua = false;
 private Bezeroa bezeroa;
 private SalmentaLangilea langilea;

 /**
 * Create the dialog.
 */
 /**
 * BezeroaDialog eraikitzailea.
 *
 * @param parent Guraso leihoa (JFrame).
 * @param title Leihoaaren izenburua.
 * @param bezeroaEdizioa Editatu beharreko bezeroa (null bada, sortu).
 * @param langilea Langilea objektua (herriak kudeatzeko).
 */
}
```

```
/*
public BezeroaDialog(JFrame parent, String title, Bezeroa bezeroaEdizioa,
SalmentaLangilea langilea) {
 super(parent, title, true);
 this.bezeroa = bezeroaEdizioa;
 this.langilea = langilea;

 setBounds(100, 100, 500, 550);
 getContentPane().setLayout(new BorderLayout());
 contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
 getContentPane().add(contentPanel, BorderLayout.CENTER);
 contentPanel.setLayout(new GridLayout(0, 2, 5, 5));

 contentPanel.add(new JLabel("Izena / Soziala:"));
 izenaField = new JTextField();
 contentPanel.add(izenaField);

 contentPanel.add(new JLabel("Abizena:"));
 abizenaField = new JTextField();
 contentPanel.add(abizenaField);

 contentPanel.add(new JLabel("NAN / IFZ:"));
 nanField = new JTextField();
 contentPanel.add(nanField);

 contentPanel.add(new JLabel("Jaiotza Data (YYYY-MM-DD):"));
 jaiotzaDataField = new JTextField();
 contentPanel.add(jaiotzaDataField);

 contentPanel.add(new JLabel("Sexua:"));
 String[] sexuak = { "gizona", "emakumea", "ez-binarioa" };
 sexuaBox = new JComboBox(<>(sexuak));
 contentPanel.add(sexuaBox);

 contentPanel.add(new JLabel("Ordainketa Txartela:"));
 txartelaField = new JTextField();
 contentPanel.add(txartelaField);

 contentPanel.add(new JLabel("Helbidea:"));
 helbideaField = new JTextField();
 contentPanel.add(helbideaField);

 // Herria ComboBox eta Botoia panel batean
 contentPanel.add(new JLabel("Herria:"));
 JPanel herriaPanela = new JPanel(new BorderLayout());
 herriaBox = new JComboBox(<>());
 herriaGehituBtn = new JButton("+");
 herriaGehituBtn.setToolTipText("Gehitu Herri Berria");
 herriaPanela.add(herriaBox, BorderLayout.CENTER);
 herriaPanela.add(herriaGehituBtn, BorderLayout.EAST);
 contentPanel.add(herriaPanela);

 contentPanel.add(new JLabel("Posta Kodea:"));
 postaKodeaField = new JTextField();
 contentPanel.add(postaKodeaField);

 contentPanel.add(new JLabel("Telefonoa:"));
```

```
telefonoaField = new JTextField();
contentPanel.add(telefonoaField);

contentPanel.add(new JLabel("Emaila:"));
emailField = new JTextField();
contentPanel.add(emailField);

contentPanel.add(new JLabel("Hizkuntza:"));
String[] hizkuntzak = { "Euskara", "Gaztelania", "Frantsesa", "Ingelesa" };
hizkuntzaBox = new JComboBox<>(hizkuntzak);
contentPanel.add(hizkuntzaBox);

contentPanel.add(new JLabel("Pasahitza:"));
pasahitzaField = new JTextField();
contentPanel.add(pasahitzaField);

// Herriak kargatu
herriakKargatu();

// Herria gehitu logika
herriaGehituBtn.addActionListener(e -> herriaBerriaGehitu());

if (bezeroa != null) {
 datuakKargatu();
}

JPanel buttonPane = new JPanel();
buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));
getContentPane().add(buttonPane, BorderLayout.SOUTH);

okButton = new JButton("Gorde");
okButton.addActionListener(e -> {
 if (balidatu()) {
 onartua = true;
 setVisible(false);
 }
});
buttonPane.add(okButton);
getRootPane().setDefaultButton(okButton);

cancelButton = new JButton("Ezeztatu");
cancelButton.addActionListener(e -> {
 onartua = false;
 setVisible(false);
});
buttonPane.add(cancelButton);
}

/**
 * Herrien zerrenda kargatzen du ComboBox-ean.
 */
private void herriakKargatu() {
 herriaBox.removeAllItems();
 if (langilea != null) {
 List<Herria> zerrenda = langilea.herriakLortu();
 for (Herria h : zerrenda) {
 herriaBox.addItem(h);
 }
 }
}
```

```

 }
 }

 /**
 * Herri berria gehitzeko leihoa erakusten du eta datu-basean gordetzen du.
 */
 private void herriaBerriaGehitu() {
 JTextField izenaF = new JTextField();
 JTextField lurrealdeaF = new JTextField();
 JTextField nazioaF = new JTextField();

 Object[] mezua = {
 "Herria:", izenaF,
 "Lurrealdea:", lurrealdeaF,
 "Nazioa:", nazioaF
 };

 int opt = JOptionPane.showConfirmDialog(this, mezua, "Gehitu Herria",
JOptionPane.OK_CANCEL_OPTION);
 if (opt == JOptionPane.OK_OPTION) {
 if (izenaF.getText().trim().isEmpty()) {
 JOptionPane.showMessageDialog(this, "Izena derrigorrezkoa da.");
 return;
 }
 try {
 Herria hBerria = new Herria(0, izenaF.getText(),
lurrealdeaF.getText(), nazioaF.getText());
 langilea.herriaSortu(hBerria);
 herriakKargatu();
 // Aukeratu berria (azkena izena arabera ordenatuta egon daiteke,
beraz bilatu)
 for (int i = 0; i < herriaBox.getItemCount(); i++) {
 Herria h = herriaBox.getItemAt(i);
 if (h.getIzena().equalsIgnoreCase(izenaF.getText())) {
 herriaBox.setSelectedIndex(i);
 break;
 }
 }
 } catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea herria sortzean: " +
e.getMessage());
 }
 }
 }

 /**
 * Editatu beharreko bezeroaren datuak kargatzen ditu eremuetan.
 */
 private void datuakKargatu() {
 izenaField.setText(bezeroa.getIzenaEdoSoziala());
 abizenaField.setText(bezeroa.getAbizena());
 nanField.setText(bezeroa.getIfzNan());
 if (bezeroa.getJaiotzaData() != null) {
 jaiotzaDataField.setText(bezeroa.getJaiotzaData().toString());
 }
 }
}

```

```

sexuaBox.setSelectedItem(bezeroa.getSexua());
txartelaField.setText(bezeroa.getBezeroaOrdainketaTxartela());
helbideaField.setText(bezeroa.getHelbidea());

// Herria hautatu ID bidez
int hId = bezeroa.getHerriaId();
for (int i = 0; i < herriaBox.getItemCount(); i++) {
 if (herriaBox.getItemAt(i).getIdHerria() == hId) {
 herriaBox.setSelectedIndex(i);
 break;
 }
}

postaKodeaField.setText(bezeroa.getPostaKodea());
telefonoaField.setText(bezeroa.getTelefonoa());
emailField.setText(bezeroa.getEmaila());
hizkuntzaBox.setSelectedItem(bezeroa.getHizkuntza());
pasahitzaField.setText(bezeroa.getPasahitza());
}

/**
 * Sartutako datuak balidatzen ditu.
 *
 * @return True datuak zuzenak badira, false bestela.
 */
private boolean validatu() {
 if (izenaField.getText().trim().isEmpty() ||
nanField.getText().trim().isEmpty()
 || emailField.getText().trim().isEmpty()) {
 JOptionPane.showMessageDialog(this, "Izena, NAN eta Emaila
derrigorrezkoak dira.", "Errorea",
 JOptionPane.ERROR_MESSAGE);
 return false;
 }
 if (herriaBox.getSelectedItem() == null) {
 JOptionPane.showMessageDialog(this, "Herria aukeratu behar da.",
 "Errorea", JOptionPane.ERROR_MESSAGE);
 return false;
 }
 return true;
}

/**
 * Erabiltzaileak OK botoia sakatu duen egiaztatzen du.
 *
 * @return True onartu bada.
 */
public boolean isOnartua() {
 return onartua;
}

/**
 * Sortu edo editatutako Bezero objektua itzultzen du.
 *
 * @return Bezeroa.
 */
public Bezeroa getBezeroa() {

```

```

 int id = (bezeroa != null) ? bezeroa.getIdBezeroa() : 0;
 Date jaiotza = null;
 try {
 if (!jaiotzaDataField.getText().trim().isEmpty()) {
 jaiotza = Date.valueOf(jaiotzaDataField.getText().trim());
 }
 } catch (Exception e) {
 } // Ignoratu formatu okerra

 Herria aukeratua = (Herria) herriaBox.getSelectedItem();
 int herriaId = (aukeratua != null) ? aukeratua.getIdHerria() : 0;

 return new Bezeroa(
 id,
 izenaField.getText().trim(),
 abizenaField.getText().trim(),
 nanField.getText().trim(),
 jaiotza,
 (String) sexuaBox.getSelectedItem(),
 txartelaField.getText().trim(),
 helbideaField.getText().trim(),
 herriaId,
 postaKodeaField.getText().trim(),
 telefonoaField.getText().trim(),
 emailField.getText().trim(),
 (String) hizkuntzaBox.getSelectedItem(),
 pasahitzaField.getText().trim(),
 null, null, true);
 }

 /**
 * Leihoa ikusteko moduan jarri (editagarria ez izateko).
 *
 * @param viewMode True ikusteko bada, false editatzeko.
 */
 public void setEditMode(boolean viewMode) {
 izenaField.setEditable(!viewMode);
 abizenaField.setEditable(!viewMode);
 nanField.setEditable(!viewMode);
 jaiotzaDataField.setEditable(!viewMode);
 sexuaBox.setEnabled(!viewMode);
 txartelaField.setEditable(!viewMode);
 helbideaField.setEditable(!viewMode);
 herriaBox.setEnabled(!viewMode);
 herriaGehituBtn.setVisible(!viewMode);
 postaKodeaField.setEditable(!viewMode);
 telefonoaField.setEditable(!viewMode);
 emailField.setEditable(!viewMode);
 hizkuntzaBox.setEnabled(!viewMode);
 pasahitzaField.setEditable(!viewMode);

 if (okButton != null) {
 okButton.setVisible(!viewMode);
 }
 }
}

```

# ComboItem.java

```
package ui;

/**
 * ComboBox-eten objektuak gorde eta bistaratzeko klase laguntzailea.
 * IDa eta testua (label) gordetzen ditu.
 */
public class ComboItem {
 private int id;
 private String label;

 /**
 * ComboItem eraikitzalea.
 *
 * @param id Identifikatzailea.
 * @param label Testua.
 */
 public ComboItem(int id, String label) {
 this.id = id;
 this.label = label;
 }

 /**
 * IDa lortzen du.
 *
 * @return IDa.
 */
 public int getId() {
 return id;
 }

 /**
 * Objektuaren testu adierazpena itzultzen du (ComboBox-ean agertuko dena).
 *
 * @return Labela.
 */
 @Override
 public String toString() {
 return label;
 }
}
```

# EskaeraDialog.java

```
package ui;

import db.DB_Konexioa;

import javax.swing.*;
import java.awt.*;
import java.math.BigDecimal;
import java.sql.*;
import javax.swing.table.DefaultTableModel;

/**
 * EskaeraDialog klasea.
 * Eskaerak sortzeko eta editatzeko leihoa.
 * Bezeroa aukeratu, produktuak gehitu/kendu eta prezio totala kalkulatzen du.
 */
public class EskaeraDialog extends JDialog {

 private static final long serialVersionUID = 1L;
 private JComboBox<BezeroItem> bezeroKomboa;
 private JComboBox<String> egoeraKomboa;
 private JLabel prezioTotalaLabel;
 private JTable lerroakTaula;
 private DefaultTableModel lerroakModel;
 private boolean onartua = false;

 // Produktuak gehitzeko elementuak
 private JComboBox<ProduktuaItem> produktuKomboa;
 private JSpinner kantitateSpinner;
 private JButton gehituLerroaBotoia;
 private JButton kenduLerroaBotoia;

 // Helper classes
 /**
 * Bezeroa ComboBox-ean erakusteko klase laguntzailea.
 */
 public static class BezeroItem {
 int id;
 String izena;

 public BezeroItem(int id, String izena) {
 this.id = id;
 this.izena = izena;
 }

 @Override
 public String toString() {
 return izena + " (ID: " + id + ")";
 }
 }

 /**
 * Produktua ComboBox-ean erakusteko klase laguntzailea.
 */
 public static class ProduktuaItem {
 int id;
 String izena;
```

```

 BigDecimal prezioa;
 int stock;
 BigDecimal eskaintza;

 public ProduktuaItem(int id, String izena, BigDecimal prezioa, int stock,
BigDecimal eskaintza) {
 this.id = id;
 this.izena = izena;
 this.prezioa = prezioa;
 this.stock = stock;
 this.eskaintza = eskaintza;
 }

 @Override
 public String toString() {
 return izena + " (" + prezioa + " \u00a0) - Stock: " + stock;
 }
}

// Constructor updated (price removed from arguments as it's calculated)
/**
 * EskaeraDialog eraikitzalea.
 *
 * @param jabea Guraso leihoa.
 * @param izenburua Leihoaaren izenburua.
 * @param bezeroaId Bezeroaren IDa (editatzen bada), bestela null.
 * @param egoera Eskaeraren egoera (editatzen bada), bestela null.
 */
public EskaeraDialog(Frame jabea, String izenburua, Integer bezeroaId, String
egoera) {
 super(jabea, izenburua, true);
 setLayout(new BorderLayout());
 setSize(800, 600); // Handiagoa egin dugu
 setLocationRelativeTo(jabea);

 // --- GOIKO PANELA (Bezeroa eta Egoera) ---
 JPanel goikoPanela = new JPanel(new GridLayout(2, 2, 10, 10));
 goikoPanela.setBorder(BorderFactory.createTitledBorder("Eskaera Datuak"));

 goikoPanela.add(new JLabel("Bezeroa:"));
 bezeroKomboa = new JComboBox<>();
 kargatuBezeroak();
 if (bezeroaId != null)
 hautatuBezeroa(bezeroaId);
 goikoPanela.add(bezeroKomboa);

 goikoPanela.add(new JLabel("Egoera:"));
 String[] egoerak = { "Prestatzen", "Osatua/Bidalita", "Ezabatua" };
 egoeraKomboa = new JComboBox<>(egoerak);
 if (egoera != null)
 egoeraKomboa.setSelectedItem(egoera);
 goikoPanela.add(egoeraKomboa);

 add(goikoPanela, BorderLayout.NORTH);

 // --- ERDIKO PANELA (Lerroak eta Produktuak) ---
 JPanel erdikoPanela = new JPanel(new BorderLayout());

```

```

 erdikoPanela.setBorder(BorderFactory.createTitledBorder("Eskaera
Lerroak"));

 // Produktua gehitzeko kontrolak
 JPanel produktuPanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
 produktuKomboa = new JComboBox();
 kargatuProduktuak();
 produktuPanela.add(new JLabel("Produktua:"));
 produktuPanela.add(produktuKomboa);

 kantitateSpinner = new JSpinner(new SpinnerNumberModel(1, 1, 1000, 1));
 produktuPanela.add(new JLabel("Kantitatea:"));
 produktuPanela.add(kantitateSpinner);

 gehituLerroaBotoia = new JButton("Gehitu");
 gehituLerroaBotoia.addActionListener(e -> lerroaGehitu());
 produktuPanela.add(gehituLerroaBotoia);

 kenduLerroaBotoia = new JButton("Kendu");
 kenduLerroaBotoia.addActionListener(e -> lerroaKendu());
 produktuPanela.add(kenduLerroaBotoia);

 erdikoPanela.add(produktuPanela, BorderLayout.NORTH);

 // Taula
 String[] zutabeak = { "Produktua ID", "Produktua", "Prezioa", "Kantitatea",
"Deskontua (%)", "Guztira" };
 lerroakModel = new DefaultTableModel(zutabeak, 0) {
 @Override
 public boolean isCellEditable(int row, int column) {
 return false;
 }
 };
 lerroakTaula = new JTable(lerroakModel);
 erdikoPanela.add(new JScrollPane(lerroakTaula), BorderLayout.CENTER);

 add(erdikoPanela, BorderLayout.CENTER);

 // --- BEHEKO PANELA (Totala eta Botoiak) ---
 JPanel behekoPanela = new JPanel(new BorderLayout());

 JPanel totalPanela = new JPanel(new FlowLayout(FlowLayout.RIGHT));
 prezioTotalaLabel = new JLabel("0.00 €");
 prezioTotalaLabel.setFont(new Font("Arial", Font.BOLD, 20));
 totalPanela.add(new JLabel("PREZIO TOTALA: "));
 totalPanela.add(prezioTotalaLabel);
 behekoPanela.add(totalPanela, BorderLayout.NORTH);

 JPanel botoiPanela = new JPanel();
 JButton onartuBotoia = new JButton("Gorde Eskaera");
 onartuBotoia.addActionListener(e -> {
 if (balidatu()) {
 onartua = true;
 dispose();
 }
 });
 JButton ezeztatuBotoia = new JButton("Ezeztatu");

```

```

 ezeztatuBotoia.addActionListener(e -> dispose());
 botoiPanela.add(onartuBotoia);
 botoiPanela.add(ezeztatuBotoia);
 behekoPanela.add(botoiPanela, BorderLayout.SOUTH);

 add(behekoPanela, BorderLayout.SOUTH);
 }

 /**
 * Bezero aktiboak kargatzen ditu ComboBox-ean.
 */
 private void kargatuBezeroak() {
 try (Connection konexioa = DB_Konexioa.konektatu();
 Statement stmt = konexioa.createStatement();
 ResultSet rs = stmt
 .executeQuery("SELECT id_bezeroa, izena_edo_soziala FROM
bezeroak WHERE aktibo = 1")) {
 while (rs.next()) {
 bezeroKomboa.addItem(new BezeroItem(rs.getInt("id_bezeroa"),
rs.getString("izena_edo_soziala")));
 }
 } catch (SQLException e) {
 e.printStackTrace();
 }
 }

 /**
 * Salgai dauden produktuak kargatzen ditu ComboBox-ean.
 */
 private void kargatuProduktuak() {
 try (Connection konexioa = DB_Konexioa.konektatu();
 Statement stmt = konexioa.createStatement();
 ResultSet rs = stmt.executeQuery(
 "SELECT id_produktua, izena, salmenta_preszia, stock,
eskaintza FROM produktuak WHERE salgai = 1")) {
 while (rs.next()) {
 produktuKomboa.addItem(new ProduktuaItem(rs.getInt("id_produktua"),
rs.getString("izena"),
rs.getBigDecimal("salmenta_preszia"), rs.getInt("stock"),
rs.getBigDecimal("eskaintza")));
 }
 } catch (SQLException e) {
 e.printStackTrace();
 }
 }

 /**
 * Aukeratutako produktua saskiari gehitzen dio.
 * Stock-a egiaztatzen du eta prezioak kalkulatzen ditu.
 */
 private void lerroaGehitu() {
 ProduktuaItem p = (ProduktuaItem) produktuKomboa.getSelectedItem();
 if (p == null)
 return;

 int kantitatea = (int) kantitateSpinner.getValue();

```

```

// Stock balidazioa: Zenbat dago jada taulan?
int dagoenKantitatea = 0;
for (int i = 0; i < lerroakModel.getRowCount(); i++) {
 if ((int) lerroakModel.getValueAt(i, 0) == p.id) {
 dagoenKantitatea += (int) lerroakModel.getValueAt(i, 3);
 }
}

if (dagoenKantitatea + kantitatea > p.stock) {
 JOptionPane.showMessageDialog(this,
 "Ezin da stock-a baino gehiago gehitu.\nStock erabilgarria: " +
p.stock
 + "\nZure hautaketa (lehendik dagoena barne): " +
(dagoenKantitatea + kantitatea),
 "Stock mugatua", JOptionPane.WARNING_MESSAGE);
 return;
}

BigDecimal deskontua = p.eskaintza != null ? p.eskaintza : BigDecimal.ZERO;

BigDecimal prezioOsoa = p.prezioa.multiply(new BigDecimal(kantitatea));
BigDecimal deskontuZenbatekoa = prezioOsoa.multiply(deskontua).divide(new
BigDecimal(100));
BigDecimal finala = prezioOsoa.subtract(deskontuZenbatekoa);

lerroakModel.addRow(new Object[] { p.id, p.izena, p.prezioa, kantitatea,
deskontua, finala });
totalaEguneratu();
}

/**
 * Aukeratutako lerroa saskitik kentzen du.
 */
private void lerroaKendu() {
 int selectedRow = lerroakTaula.getSelectedRow();
 if (selectedRow != -1) {
 lerroakModel.removeRow(selectedRow);
 totalaEguneratu();
 }
}

/**
 * Saskiko prezio totala birkalkulatzen eta egunерatzentzu du.
 */
private void totalaEguneratu() {
 BigDecimal totala = BigDecimal.ZERO;
 for (int i = 0; i < lerroakModel.getRowCount(); i++) {
 totala = totala.add((BigDecimal) lerroakModel.getValueAt(i, 5)); // 5 =
Guztira zutabea
 }
 prezioTotalaLabel.setText(String.format("%.2f \u20AC", totala));
}

/**
 * Bezero zehatz bat hautatzen du ComboBox-ean (editatzean).
 *
 * @param id Bezeroaren IDa.

```

```

/*
private void hautatuBezeroa(int id) {
 for (int i = 0; i < bezeroKomboa.getItemCount(); i++) {
 if (bezeroKomboa.getItemAt(i).id == id) {
 bezeroKomboa.setSelectedIndex(i);
 break;
 }
 }
}

/**
 * Eskaera balidatzen du (bezeroa aukeratuta eta lerroak egotea).
 *
 * @return True baliozkoa bada.
 */
private boolean balidatu() {
 if (bezeroKomboa.getSelectedItem() == null) {
 JOptionPane.showMessageDialog(this, "Aukeratu bezero bat.");
 return false;
 }
 if (lerroakModel.getRowCount() == 0) {
 JOptionPane.showMessageDialog(this, "Gehitu gutxienez produktu bat.");
 return false;
 }
 return true;
}

/**
 * Lerro bat zuzenean gehitzen du taulara (kargatzean erabiltzen da).
 *
 * @param id Produktu IDa.
 * @param izena Produktu izena.
 * @param prezioa Unitate prezioa.
 * @param kantitatea Kantitatea.
 * @param deskontua Deskontua (%).
 */
public void addZuzeneanLerroa(int id, String izena, BigDecimal prezioa, int
kantitatea, double deskontua) {
 BigDecimal prezioOsoa = prezioa.multiply(new BigDecimal(kantitatea));
 BigDecimal deskontuZenbateko = prezioOsoa.multiply(new
BigDecimal(deskontua)).divide(new BigDecimal(100));
 BigDecimal finala = prezioOsoa.subtract(deskontuZenbateko);

 lerroakModel.addRow(new Object[] { id, izena, prezioa, kantitatea,
deskontua, finala });
 totalaEguneratu();
}

/**
 * Erabiltzaileak onartu duen egiaztatzen du.
 *
 * @return True onartu bada.
 */
public boolean isOnartua() {
 return onartua;
}

```

```

/**
 * Aukeratutako bezeroaren IDa itzultzen du.
 *
 * @return Bezero IDa.
 */
public int getBezeroaId() {
 return ((BezeroItem) bezeroKomboa.getSelectedItem()).id;
}

/**
 * Aukeratutako egoera itzultzen du.
 *
 * @return Egoera.
 */
public String getEgoera() {
 return (String) egoeraKomboa.getSelectedItem();
}

/**
 * Prezio totala itzultzen du.
 *
 * @return Totala.
 */
public BigDecimal getPrezioTotala() {
 String t = prezioTotalaLabel.getText().replace(" \u20AC", "").replace(",",
".");
 return new BigDecimal(t);
}

/**
 * Saskiko lerroak itzultzen ditu.
 *
 * @return Objektu array zerrenda: [prodId, kantitatea, unitatePrezioa
 * (deskontuarekin)].
 */
public java.util.List<Object[]> getLerroak() {
 java.util.List<Object[]> lista = new java.util.ArrayList<>();
 for (int i = 0; i < lerroakModel.getRowCount(); i++) {
 int prodId = (int) lerroakModel.getValueAt(i, 0);
 int kant = (int) lerroakModel.getValueAt(i, 3);
 BigDecimal totalaRow = (BigDecimal) lerroakModel.getValueAt(i, 5);
 // Unitate prezioa (deskontuarekin) = Totala / Kantitatea
 BigDecimal unitatePrezioa = totalaRow.divide(new BigDecimal(kant), 2,
java.math.RoundingMode.HALF_UP);

 lista.add(new Object[] { prodId, kant, unitatePrezioa });
 }
 return lista;
}

```

# FitxaketakKudeatu.java

```
package ui;

import db.DB_Konexioa;

import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.*;

/**
 * FitxaketakKudeatu klasea.
 * Langileen fitxaketak (sarrerak eta irteerak) kudeatzeko interfaze grafikoa.
 * Fitxaketak ikusi, sortu, editatu eta ezabatzeko aukera ematen du.
 */
public class FitxaketakKudeatu extends JFrame {

 private static final long serialVersionUID = 1L;
 private JPanel edukiPanela;
 private JTable taula;
 private DefaultTableModel eredua;

 private class Langilea {
 int id;
 String izena;
 String abizena;

 public Langilea(int id, String izena, String abizena) {
 this.id = id;
 this.izena = izena;
 this.abizena = abizena;
 }

 @Override
 public String toString() {
 return id + " - " + izena + " " + abizena;
 }
 }

 /**
 * Applikazioa abiarazi.
 */
 public static void main(String[] args) {
 EventQueue.invokeLater(new Runnable() {
 public void run() {
 try {
 FitxaketakKudeatu frame = new
FitxaketakKudeatu();
 frame.setVisible(true);
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 });
 }
}
```

```
/**
 * Leihoa sortu eta inizializatu.
 */
public FitxaketakKudeatu() {
 setTitle("Birtek - Langileen Fitxaketak Kudeatu");
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 setBounds(100, 100, 800, 500);
 edukiPanela = new JPanel();
 edukiPanela.setBorder(new EmptyBorder(5, 5, 5, 5));
 setContentPane(edukiPanela);
 edukiPanela.setLayout(new BorderLayout(0, 0));

 // Goiko panela (Izenburua)
 JPanel goikoPanela = new JPanel();
 goikoPanela.setBackground(new Color(0, 102, 102));
 edukiPanela.add(goikoPanela, BorderLayout.NORTH);
 JLabel izenburuEtiketa = new JLabel("FITXAKETEN KUDEAKETA");
 izenburuEtiketa.setForeground(Color.WHITE);
 izenburuEtiketa.setFont(new Font("SansSerif", Font.BOLD, 18));
 goikoPanela.add(izenburuEtiketa);

 // Erdiko panela (Taula)
 JScrollPane korritzePanela = new JScrollPane();
 edukiPanela.add(korritzePanela, BorderLayout.CENTER);
 taula = new JTable();
 korritzePanela.setViewportView(taula);

 // Beheko panela (Botoiak)
 JPanel behekoPanela = new JPanel();
 edukiPanela.add(behekoPanela, BorderLayout.SOUTH);
 behekoPanela.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

 JButton kargatuBotoia = new JButton("Kargatu");
 kargatuBotoia.addActionListener(e -> datuakKargatu());
 behekoPanela.add(kargatuBotoia);

 JButton berriaBotoia = new JButton("Berria");
 berriaBotoia.addActionListener(e -> fitxaketaBerria());
 behekoPanela.add(berriaBotoia);

 JButton aldatuBotoia = new JButton("Modifikatu");
 aldatuBotoia.addActionListener(e -> fitxaketaAldatu());
 behekoPanela.add(aldatuBotoia);

 JButton ezabatuBotoia = new JButton("Ezabatu");
 ezabatuBotoia.addActionListener(e -> fitxaketaEzabatu());
 behekoPanela.add(ezabatuBotoia);

 JButton irtenBotoia = new JButton("Ateria");
 irtenBotoia.addActionListener(e -> dispose());
 behekoPanela.add(irtenBotoia);

 // Hasierako datuak kargatu
 datuakKargatu();
}
/**
```

```

 * Datu-basetik fitxaketa datuak kargatu eta taulan erakutsi.
 */
 private void datuakKargatu() {
 String galdera = "SELECT id_fitzaketa, langilea_id, data,
CAST(ordua AS CHAR) AS ordua, mota FROM fitxaketak ORDER BY id_fitzaketa DESC";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement sententziaPrestatua =
konexioa.prepareStatement(galdera);
 ResultSet emaitza =
sententziaPrestatua.executeQuery()) {

 eredua = TaulaModelatzaila.eredueraiki(emaitza);
 taula.setModel(eredua);

 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea datuak
kargatzean: " + e.getMessage(), "Errorea",
JOptionPane.ERROR_MESSAGE);
 }
 }

 /**
 * Langileen zerrenda lortu ComboBox-ean erakusteko.
 *
 * @return Langileen ComboBox eredua.
 */
 private DefaultComboBoxModel<Langilea> langileakLortu() {
 DefaultComboBoxModel<Langilea> eredua = new DefaultComboBoxModel<>
();
 String galdera = "SELECT id_langilea, izena, abizena FROM langileak
ORDER BY id_langilea";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement sententziaPrestatua =
konexioa.prepareStatement(galdera);
 ResultSet emaitza =
sententziaPrestatua.executeQuery()) {
 while (emaitza.next()) {
 eredua.addElement(new Langilea(
 emaitza.getInt("id_langilea"),
 emaitza.getString("izena"),
 emaitza.getString("abizena")));
 }
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea langileak
kargatzean: " + e.getMessage());
 }
 return eredua;
 }

 /**
 * Fitxaketa berri bat sortzeko elkarrizketa-leihoa ireki eta datuak gorde.
 */
 private void fitxaketaBerria() {
 JComboBox<Langilea> langileaHautatzailea = new JComboBox<>
(langileakLortu());
 String[] aukerak = { "Sarrera", "Irteera" };
 JComboBox<String> motaHautatzailea = new JComboBox<>(aukerak);

```

```

 java.time.LocalDateTime orain = java.time.LocalDateTime.now();
 JTextField dataTestua = new
JTextField(orain.toLocalDate().toString());
 JTextField orduaTestua = new
JTextField(orain.toLocalTime().withNano(0).toString());

 Object[] mezua = {
 "Langilea:", langileaHautatzailea,
 "Mota:", motaHautatzailea,
 "Data (YYYY-MM-DD):", dataTestua,
 "Ordua (HH:MM:SS):", orduaTestua
 };

 int aukera = JOptionPane.showConfirmDialog(null, mezua, "Fitxaketa
Berria", JOptionPane.OK_CANCEL_OPTION);
 if (aukera == JOptionPane.OK_OPTION &&
langileaHautatzailea.getSelectedItem() != null) {
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 String galdera = "INSERT INTO fitxaketak
(langilea_id, mota, data, ordua) VALUES (?, ?, ?, ?)";
 PreparedStatement sententziaPrestatua =
konexioa.prepareStatement(galdera);
 Langilea hautatua = (Langilea)
langileaHautatzailea.getSelectedItem();
 sententziaPrestatua.setInt(1, hautatua.id);
 sententziaPrestatua.setString(2, (String)
motaHautatzailea.getSelectedItem());
 sententziaPrestatua.setString(3,
dataTestua.getText());
 sententziaPrestatua.setString(4,
orduaTestua.getText());

 if (sententziaPrestatua.executeUpdate() > 0) {
 // JOptionPane.showMessageDialog(this,
"Fitxaketa ongi gehitu da.");
 datuakKargatu();
 }
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea
gordetzean: " + e.getMessage());
 }
 }

 /**
 * Hautatutako fitxaketa editatzeko elkarritzeta-leihoa ireki eta aldaketak
 * gorde.
 */
 private void fitxaketaAldatu() {
 int errenkada = taula.getSelectedRow();
 if (errenkada == -1) {
 JOptionPane.showMessageDialog(this, "Hautatu fitxaketa bat
modifikatzeko.");
 return;
 }
 }
 }
}

```

```

 int id = Integer.parseInt(taula.getValueAt(errenkada,
0).toString());
 int langileaIdHeldua = Integer.parseInt(taula.getValueAt(errenkada,
1).toString());
 String dataHeldua = taula.getValueAt(errenkada, 2).toString();
 String orduaHeldua = taula.getValueAt(errenkada, 3).toString();
 String motaHautatua = taula.getValueAt(errenkada, 4).toString();

 JComboBox<Langilea> langileaHautatzailea = new JComboBox<>
(langileakLortu());
 // Hautatutako langilea ComboBox-ean bilatu
 for (int i = 0; i < langileaHautatzailea.getItemCount(); i++) {
 if (langileaHautatzailea.getItemAt(i).id ==
langileaIdHeldua) {
 langileaHautatzailea.setSelectedIndex(i);
 break;
 }
 }

 String[] aukerak = { "Sarrera", "Irteera" };
 JComboBox<String> motaHautatzailea = new JComboBox<>(aukerak);
 motaHautatzailea.setSelectedItem(motaHautatua);

 JTextField dataTestua = new JTextField(dataHeldua);
 JTextField orduaTestua = new JTextField(orduaHeldua);

 Object[] mezua = {
 "Langilea:", langileaHautatzailea,
 "Mota:", motaHautatzailea,
 "Data (YYYY-MM-DD):", dataTestua,
 "Ordua (HH:MM:SS):", orduaTestua
 };

 int aukera = JOptionPane.showConfirmDialog(null, mezua, "Aldatu
Fitxaketa", JOptionPane.OK_CANCEL_OPTION);
 if (aukera == JOptionPane.OK_OPTION &&
langileaHautatzailea.getSelectedItem() != null) {
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 String galdera = "UPDATE fitxaketak SET langilea_id
= ?, mota = ?, data = ?, ordua = ? WHERE id_fitxaketa = ?";
 PreparedStatement sententziaPrestatua =
konexioa.prepareStatement(galdera);
 Langilea hautatua = (Langilea)
langileaHautatzailea.getSelectedItem();
 sententziaPrestatua.setInt(1, hautatua.id);
 sententziaPrestatua.setString(2, (String)
motaHautatzailea.getSelectedItem());
 sententziaPrestatua.setString(3,
dataTestua.getText());
 sententziaPrestatua.setString(4,
orduaTestua.getText());
 sententziaPrestatua.setInt(5, id);

 if (sententziaPrestatua.executeUpdate() > 0) {
 // JOptionPane.showMessageDialog(this,
 "Fitxaketa ongi egunerau da.");
 datuakKargatu();
 }
 }
 }
 }
}

```

```

 }
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea
eguneratzean: " + e.getMessage());
 }
}

/**
 * Hautatutako fitxaketa ezabatu.
 */
private void fitxaketaEzabatu() {
 int errenkada = taula.getSelectedRow();
 if (errenkada == -1) {
 JOptionPane.showMessageDialog(this, "Hautatu fitxaketa bat
ezabatzeko.");
 return;
 }

 int id = Integer.parseInt(taula.getValueAt(errenkada,
0).toString());
 int berretsi = JOptionPane.showConfirmDialog(this, "Ziur al zaude
fitxaketa hau ezabatu nahi duzula?",
 "Berretsi", JOptionPane.YES_NO_OPTION);

 if (berretsi == JOptionPane.YES_OPTION) {
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 String galdera = "DELETE FROM fitxaketak WHERE
id_fitzaketa = ?";
 PreparedStatement sententziaPrestatua =
konexioa.prepareStatement(galdera);
 sententziaPrestatua.setInt(1, id);

 if (sententziaPrestatua.executeUpdate() > 0) {
 // JOptionPane.showMessageDialog(this,
 "Fitxaketa ezabatu da.");
 datuakKargatu();
 }
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea
ezabatzean: " + e.getMessage());
 }
 }
}

```

# KonponketaXehetasunaElkarritzeta.java

```

package ui;

import db.DB_Konexioa;

```

```
import model.*;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JComboBox;
import javax.swing.JTextArea;
import java.sql.Connection;
import java.sql.PreparedStatement;

/**
 * KonponketaXehetasunaElkarrizketa klasea.
 * Konponketa baten egoera, akatsa eta oharrak editatzeko elkarrizketa-leihoa
 * (JDialog).
 */
public class KonponketaXehetasunaElkarrizketa extends JDialog {

 private static final long serialVersionUID = 1L;
 private final JPanel edukiPanela = new JPanel();
 private JComboBox<String> egoeraHautatzailea;
 private JComboBox<ComboItem> akatsaHautatzailea;
 private JTextArea oharrakTestua;
 private int konponketaId;

 /**
 * KonponketaXehetasunaElkarrizketa eraikitzailea.
 *
 * @param id Konponketaren IDa.
 * @param unekoEgoera Uneko egoera.
 * @param unekoOharrak Uneko oharrak.
 * @param unekoAkatsaId Uneko akatsaren IDa.
 * @param akatsakList Akats posibleen zerrenda.
 */
 public KonponketaXehetasunaElkarrizketa(int id, String unekoEgoera, String
unekoOharrak, int unekoAkatsaId,
 java.util.List<Akatsa> akatsakList) {
 this.konponketaId = id;
 setTitle("Editatu Konponketa" + " (ID: " + id + ")");
 setBounds(100, 100, 450, 400); // Increased height
 getContentPane().setLayout(new BorderLayout());
 edukiPanela.setBorder(new EmptyBorder(5, 5, 5, 5));
 getContentPane().add(edukiPanela, BorderLayout.CENTER);
 edukiPanela.setLayout(null);

 // Egoera
 JLabel egoeraEtiketa = new JLabel("Egoera:");
 egoeraEtiketa.setBounds(20, 30, 80, 14);
 edukiPanela.add(egoeraEtiketa);

 String[] egoerak = { "Prozesuan", "Konponduta", "Konponezina" };
 egoeraHautatzailea = new JComboBox<>(egoerak);
 egoeraHautatzailea.setBounds(110, 27, 200, 22);
 }
}
```

```

egoeraHautatzailea.setSelectedItem(unekoEgoera);
edukiPanela.add(egoeraHautatzailea);

// Akatsa
JLabel akatsaEtiketa = new JLabel("Akatsa:");
akatsaEtiketa.setBounds(20, 70, 80, 14);
edukiPanela.add(akatsaEtiketa);

akatsaHautatzailea = new JComboBox<>();
for (Akatsa a : akatsakList) {
 ComboItem item = new ComboItem(a.getIdAkatsa(), a.getIzena());
 akatsaHautatzailea.addItem(item);
 if (a.getIdAkatsa() == unekoAkatsaId) {
 akatsaHautatzailea.setSelectedItem(item);
 }
}
akatsaHautatzailea.setBounds(110, 67, 200, 22);
edukiPanela.add(akatsaHautatzailea);

// Oharrak
JLabel oharrakEtiketa = new JLabel("Oharrak:");
oharrakEtiketa.setBounds(20, 120, 80, 14);
edukiPanela.add(oharrakEtiketa);

oharrakTestua = new JTextArea();
oharrakTestua.setBounds(110, 115, 280, 100);
oharrakTestua.setText(unekoOharrak);
edukiPanela.add(oharrakTestua);

// Botoiak
 JPanel botoiPanela = new JPanel();
botoiPanela.setLayout(new FlowLayout(FlowLayout.RIGHT));
getContentPane().add(botoiPanela, BorderLayout.SOUTH);

 JButton gordeBotoia = new JButton("GORDE");
gordeBotoia.addActionListener(e -> gordeDatuak());
botoiPanela.add(gordeBotoia);

 JButton utziBotoia = new JButton("Utzi");
utziBotoia.addActionListener(e -> dispose());
botoiPanela.add(utziBotoia);
}

/**
 * Datuak datu-basean gordetzen ditu (UPDATE).
 */
private void gordeDatuak() {
 String egoeraBerria = (String) egoeraHautatzailea.getSelectedItem();
 ComboItem aukeratutakoAkatsa = (ComboItem)
akatsaHautatzailea.getSelectedItem();
 String oharBerriak = oharrakTestua.getText();

 if (aukeratutakoAkatsa == null) {
 JOptionPane.showMessageDialog(this, "Akatsa aukeratu behar da.");
 return;
 }
}

```

```

String galdera = "UPDATE konponketak SET konponketa_egoera = ?, akatsa_id = ?,
 oharrak = ? WHERE id_konponketa = ?";

try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement sententziaPrestatua =
konexioa.prepareStatement(galdera)) {

 sententziaPrestatua.setString(1, egoeraBerria);
 sententziaPrestatua.setInt(2, aukeratutakoAkatsa.getId());
 sententziaPrestatua.setString(3, oharBerriak);
 sententziaPrestatua.setInt(4, konponketaId);

 int emaitza = sententziaPrestatua.executeUpdate();
 if (emaitza > 0) {
 dispose();
 }
}

} catch (Exception ex) {
 ex.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea eguneraztzean: " +
ex.getMessage());
}
}

// Helper class for ComboBox items
/**
 * ComboBox-erako elementu laguntzailea (ID eta Testua).
 */
private static class ComboItem {
 private int id;
 private String label;

 public ComboItem(int id, String label) {
 this.id = id;
 this.label = label;
 }

 public int getId() {
 return id;
 }

 @Override
 public String toString() {
 return label;
 }
}
}

```

## MenuAdministrazioa.java

```
package ui;

import db.DB_Konexioa;
import model.*;

import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.sql.*;

/**
 * MenuAdministrazioa klasea.
 * Administrari langilearen interfaze nagusia.
 * Langileak, sailak, fitxaketak, fakturak, hornitzaleak eta herriak kudeatzeko
 * aukerak eskaintzen ditu.
 */
public class MenuAdministrazioa extends JFrame {

 private static final long serialVersionUID = 1L;
 private JPanel edukiPanela;
 private JTable langileTaula, sailaTaula, fitxaketaTaula, fakturaTaula,
hornitzaleTaula, herriaTaula;
 private JTextField bilatuTestua;
 private TableRowSorter<DefaultTableModel> langileOrdenatzailea,
sailaOrdenatzailea, fitxaketaOrdenatzailea,
 fakturaOrdenatzailea, hornitzaleOrdenatzailea, herriaOrdenatzailea,
 unekoOrdenatzailea;

 // Fitxaketa informazioa
 private JLabel fitxaketaInfoEtiketa;
 private JButton gehituBotoia, editatuBotoia, ezabatuBotoia, ikusiFakturaBotoia,
ikusiKbBotoia;

 // Erabiltzailearen datuak
 // Erabiltzailea (OOP)
 private AdministrariLangilea langilea;

 /**
 * Eraikitzailea eguneratua.
 */
 /**
 * Eraikitzailea parametroarekin.
 *
 * @param oinarrizkoLangilea Saioa hasi duen langilea.
 */
 public MenuAdministrazioa(Langilea oinarrizkoLangilea) {
 this.langilea = new AdministrariLangilea(oinarrizkoLangilea);
 pantailaPrestatu();
 }

 /**
 * Pantailaren osagaiak inizializatu eta kokatu.

```

```

*/
private void pantailaPrestatu() {
 setTitle("Birtek - ADMINISTRAZIOA");
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 setBounds(100, 100, 1050, 650);
 edukiPanela = new JPanel();
 edukiPanela.setBorder(new EmptyBorder(5, 5, 5, 5));
 setContentPane(edukiPanela);
 edukiPanela.setLayout(new BorderLayout(0, 0));

 // --- GOIKO PANELOA ---
 JPanel goikoPanela = new JPanel(new BorderLayout());
 edukiPanela.add(goikoPanela, BorderLayout.NORTH);

 // EZKERRA: Bilatzailea
 JPanel bilatzailePanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
 bilatzailePanela.add(new JLabel("Bilatu: "));
 bilatuTestua = new JTextField(20);
 bilatuTestua.addKeyListener(new KeyAdapter() {
 public void keyReleased(KeyEvent e) {
 filtratu();
 }
 });
 bilatzailePanela.add(bilatuTestua);

 JButton eguneratuBotoia = new JButton("Eguneratu");
 eguneratuBotoia.setBackground(new Color(173, 216, 230)); // Light Blue
 eguneratuBotoia.addActionListener(e -> datuakKargatuOsoa());
 bilatzailePanela.add(eguneratuBotoia);

 goikoPanela.add(bilatzailePanela, BorderLayout.WEST);

 // ESKUINALDEA: Erabiltzailearen informazioa + Fitxaketa + Saioa Itxi
 JPanel eskuinekoPanela = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10,
0));
}

// 1. Erabiltzailearen informazioaren etiketa
JLabel erabiltzaileEtiketa = new JLabel(langilea.getIzena() + " " +
langilea.getAbizena());
erabiltzaileEtiketa.setFont(new Font("SansSerif", Font.BOLD, 12));
erabiltzaileEtiketa.setForeground(new Color(0, 102, 102));

// 2. Fitxaketa Panela
JPanel fitxaketaPanela = new JPanel();
fitxaketaPanela.setLayout(new BoxLayout(fitxaketaPanela,
BoxLayout.Y_AXIS));

JPanel botoiPanela = new JPanel(new FlowLayout(FlowLayout.CENTER, 2, 0));
JButton sarreraBotoia = new JButton("Sarrera");
sarreraBotoia.setBackground(new Color(34, 139, 34));
sarreraBotoia.setForeground(Color.BLACK);
sarreraBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
sarreraBotoia.addActionListener(e -> fitxatu("Sarrera"));

JButton irteeraBotoia = new JButton("Irteera");
irteeraBotoia.setBackground(new Color(255, 140, 0));
irteeraBotoia.setForeground(Color.BLACK);

```

```

irteeraBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
irteeraBotoia.addActionListener(e -> fitxatu("Irteera"));

JButton historialBotoia = new JButton("Historiala");
historialBotoia.setBackground(new Color(100, 149, 237));
historialBotoia.setForeground(Color.BLACK);
historialBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
historialBotoia.addActionListener(e -> ikusiFitzaketaHistoriala());

JButton nireDatuakBotoia = new JButton("Nire Datuak");
nireDatuakBotoia.setBackground(new Color(100, 149, 237));
nireDatuakBotoia.setForeground(Color.BLACK);
nireDatuakBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
nireDatuakBotoia.addActionListener(e -> irekiNireDatuakEditatu());

botoiPanela.add(sarreraBotoia);
botoiPanela.add(irteeraBotoia);
botoiPanela.add(historialBotoia);
botoiPanela.add(nireDatuakBotoia);

fitxaketaInfoEtiketa = new JLabel("Kargatzen...");
fitxaketaInfoEtiketa.setFont(new Font("SansSerif", Font.PLAIN, 9));
fitxaketaInfoEtiketa.setAlignmentX(Component.CENTER_ALIGNMENT);

fitxaketaPanela.add(botoiPanela);
fitxaketaPanela.add(fitxaketaInfoEtiketa);

// 3. Saioa Itxi
JButton saioaItxiBotoia = new JButton("Saioa Itxi");
saioaItxiBotoia.setBackground(new Color(220, 20, 60));
saioaItxiBotoia.setForeground(Color.WHITE);
saioaItxiBotoia.addActionListener(e -> saioaItxi()));

// 4. Atzera botoia (Zuzendaritza bakarrik - hau kentzen dugu momentuz edo
// logika hobetu beharko litzateke)

// Gehitu panelera
eskuinekoPanela.add(erabiltzaileEtiketa);
eskuinekoPanela.add(fitxaketaPanela);
eskuinekoPanela.add(saioaItxiBotoia);

goikoPanela.add(eskuinekoPanela, BorderLayout.EAST);

// --- ERDIKO PANELA ---
JTabbedPane pestainaPanela = new JTabbedPane(JTabbedPane.TOP);
edukiPanela.add(pestainaPanela, BorderLayout.CENTER);

// --- LANGILEAK TAB ---
 JPanel langilePanela = new JPanel(new BorderLayout());
pestainaPanela.addTab("Langileak", null, langilePanela, null);
langileTaula = new JTable();
langilePanela.add(new JScrollPane(langileTaula), BorderLayout.CENTER);

// --- SAILAK TAB ---
 JPanel sailaPanela = new JPanel(new BorderLayout());
pestainaPanela.addTab("Sailak", null, sailaPanela, null);
sailaTaula = new JTable();

```

```

sailaPanela.add(new JScrollPane(sailaTaula), BorderLayout.CENTER);

// --- FITXAKETAK TAB ---
JPanel fitxaketakPanela = new JPanel(new BorderLayout());
pestainaPanela.addTab("Fitzaketak", null, fitxaketakPanela, null);
fitxaketaTaula = new JTable();
fitxaketakPanela.add(new JScrollPane(fitxaketaTaula), BorderLayout.CENTER);

// --- FAKTURAK TAB ---
JPanel fakturaPanela = new JPanel(new BorderLayout());
pestainaPanela.addTab("Fakturak", null, fakturaPanela, null);
fakturaTaula = new JTable();
fakturaPanela.add(new JScrollPane(fakturaTaula), BorderLayout.CENTER);

// --- HORNITZAILEAK TAB ---
JPanel hornitzairePanela = new JPanel(new BorderLayout());
pestainaPanela.addTab("Hornitzaireak", null, hornitzairePanela, null);
hornitzaireTaula = new JTable();
hornitzairePanela.add(new JScrollPane(hornitzaireTaula),
BorderLayout.CENTER);

// --- HERRIAK TAB ---
JPanel herriaPanela = new JPanel(new BorderLayout());
pestainaPanela.addTab("Herriak", null, herriaPanela, null);
herriaTaula = new JTable();
herriaPanela.add(new JScrollPane(herriaTaula), BorderLayout.CENTER);

JPanel botoiCrudPanela = new JPanel(new FlowLayout(FlowLayout.CENTER, 10,
5));
gehituBotoia = new JButton("Gehitu +");
editatuBotoia = new JButton("Editatu");
ezabatuBotoia = new JButton("Ezabatu");
ikusiFakturaBotoia = new JButton("Ikusi Faktura");
ikusiFakturaBotoia.setVisible(false); // Hasieran ezkutatu
ikusiKbBotoia = new JButton("Kurrikuluma Ikusi");
ikusiKbBotoia.setVisible(true); // Langileak hasieran hautatuta dagoenez,
ikusi

gehituBotoia.addActionListener(e ->
gehituElementua(pestainaPanela.getSelectedIndex()));
editatuBotoia.addActionListener(e ->
editatuElementua(pestainaPanela.getSelectedIndex()));
ezabatuBotoia.addActionListener(e ->
ezabatuElementua(pestainaPanela.getSelectedIndex()));
ikusiFakturaBotoia.addActionListener(e -> ikusiFaktura());
ikusiKbBotoia.addActionListener(e -> ikusiKurrikuluma());

botoiCrudPanela.add(gehituBotoia);
botoiCrudPanela.add(editatuBotoia);
botoiCrudPanela.add(ezabatuBotoia);
botoiCrudPanela.add(ikusiFakturaBotoia);
botoiCrudPanela.add(ikusiKbBotoia);

JPanel behekoPanela = new JPanel(new BorderLayout());
behekoPanela.add(botoiCrudPanela, BorderLayout.NORTH);

pestainaPanela.addChangeListener(e -> {

```

```

 bilatuTestua.setText("");
 int index = pestainaPanela.getSelectedIndex();
 ikusiFakturaBotoia.setVisible(index == 3); // 3 = Fakturak
 ikusiKbBotoia.setVisible(index == 0); // 0 = Langileak

 // Fakturak (index 3) denean, ezin da gehitu, editatu edo ezabatu
 // Hornitzaleak (index 4) denean, ezin da gehitu
 boolean fakturakDira = (index == 3);
 boolean hornitzaleakDira = (index == 4);

 gehituBotoia.setEnabled(!fakturakDira && !hornitzaleakDira);
 editatuBotoia.setEnabled(!fakturakDira);
 ezabatuBotoia.setEnabled(!fakturakDira);

 switch (index) {
 case 0:
 unekoOrdenatzailea = langileOrdenatzailea;
 break;
 case 1:
 unekoOrdenatzailea = sailaOrdenatzailea;
 break;
 case 2:
 unekoOrdenatzailea = fitxaketaOrdenatzailea;
 break;
 case 3:
 unekoOrdenatzailea = fakturaOrdenatzailea;
 break;
 case 4:
 unekoOrdenatzailea = hornitzaleOrdenatzailea;
 break;
 case 5:
 unekoOrdenatzailea = herriaOrdenatzailea;
 break;
 }

 });
});

JButton kargatuBotoia = new JButton("Datuak Deskargatu / Kargatu Berriro");
kargatuBotoia.addActionListener(e -> datuakKargatuOsoa());
behekoPanela.add(kargatuBotoia, BorderLayout.CENTER);

edukiPanela.add(behekoPanela, BorderLayout.SOUTH);

if (!java.beans.Beans.isDesignTime()) {
 datuakKargatuOsoa();
 eguneratuFtxaketaEgoera();
}
}

// --- FITXAKETA LOGIKA ---
/**
 * Fitxaketa bat burutzen du (Sarrera edo Irteera).
 *
 * @param mota "Sarrera" edo "Irteera".
 */
private void fitxatu(String mota) {
 try {

```

```

 langilea.fitxatu(mota);
 eguneratuFitxaketaEgoera();
 datuakKargatuOsoa(); // Taulak freskatzu
 JOptionPane.showMessageDialog(this, mota + " ondo erregistratu da.");
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, e.getMessage(), "Errorea",
JOptionPane.WARNING_MESSAGE);
 }
}

/**
 * Fitxaketa egoera etiketa eguneratzen du (Barruan/Kanpoan).
 */
private void eguneratuFitxaketaEgoera() {
 fitxaketaInfoEtiketa.setText(langilea.getFitxaketaEgoera());
 if (fitxaketaInfoEtiketa.getText().contains("BARRUAN")) {
 fitxaketaInfoEtiketa.setForeground(new Color(0, 100, 0));
 } else {
 fitxaketaInfoEtiketa.setForeground(new Color(200, 0, 0));
 }
}

/**
 * Erabiltzailearen fitxaketa historia erakusten duen leihoa ireki.
 */
private void ikusiFitxaketaHistoriala() {
 JDialog elkarrizketa = new JDialog(this, "Nire Fitxaketa Historiala",
true);
 elkarrizketa.setSize(500, 400);
 elkarrizketa.setLocationRelativeTo(this);
 elkarrizketa.setLayout(new BorderLayout());
 String[] zutabeak = { "Mota", "Data", "Ordua" };
 DefaultTableModel eredua = new DefaultTableModel(zutabeak, 0);
 JTable taula = new JTable(eredua);
 elkarrizketa.add(new JScrollPane(taula), BorderLayout.CENTER);

 java.util.List<Fitxaketa> zerrenda = langilea.nireFitxaketakIkusi();
 for (Fitxaketa f : zerrenda) {
 eredua.addRow(new Object[] { f.getMota(), f.getData(), f.getOrdua() });
 }

 JButton itxiBotoia = new JButton("Itxi");
 itxiBotoia.addActionListener(e -> elkarrizketa.dispose());
 JPanel botoiPanela = new JPanel();
 botoiPanela.add(itxiBotoia);
 elkarrizketa.add(botoiPanela, BorderLayout.SOUTH);

 elkarrizketa.setVisible(true);
}

/**
 * Erabiltzailearen datuak editatzeko leihoa ireki.
 */
private void irekiNireDatuakEditatu() {
 NireDatuakDialog dialog = new NireDatuakDialog(this, langilea);
 dialog.setVisible(true);
}

```

```

/**
 * Taula guztiak datuak datu-basetik kargatu eta egunerau.
 */
private void datuakKargatuOsoa() {
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 // Langileak
 try {
 PreparedStatement pstL = konexioa.prepareStatement(
 "SELECT id_langilea, izena, abizena, nan, jaiotza_data,
herria_id, helbidea, posta_kodea, telefonoa, emaila FROM langileak");
 DefaultTableModel mL =
TaulaModelatzalea.ereduEraiki(pstL.executeQuery());
 langileTaula.setModel(mL);
 langileOrdenatzalea = new TableRowSorter<>(mL);
 langileTaula.setRowSorter(langileOrdenatzalea);
 } catch (Exception e) {
 System.err.println("Errorea Langileak kargatzean: " +
e.getMessage());
 e.printStackTrace();
 }

 // Sailak
 try {
 PreparedStatement pstS = konexioa.prepareStatement("SELECT * FROM
langile_sailak");
 DefaultTableModel mS =
TaulaModelatzalea.ereduEraiki(pstS.executeQuery());
 sailaTaula.setModel(mS);
 sailaOrdenatzalea = new TableRowSorter<>(mS);
 sailaTaula.setRowSorter(sailaOrdenatzalea);
 } catch (Exception e) {
 System.err.println("Errorea Sailak kargatzean: " + e.getMessage());
 e.printStackTrace();
 }

 // Fitxaketak (Globala)
 try {
 PreparedStatement pstF = konexioa.prepareStatement(
 "SELECT f.id_fitxaketa, CONCAT(l.izena, ' ', l.abizena) AS
langilea, f.data, CAST(f.ordua AS CHAR) AS ordua, f.mota "
 +
 "FROM fitxaketak f JOIN langileak l ON
f.langilea_id = l.id_langilea ORDER BY f.id_fitxaketa DESC");
 DefaultTableModel mF =
TaulaModelatzalea.ereduEraiki(pstF.executeQuery());
 fitxaketaTaula.setModel(mF);
 fitxaketaOrdenatzalea = new TableRowSorter<>(mF);
 fitxaketaTaula.setRowSorter(fitxaketaOrdenatzalea);
 } catch (Exception e) {
 System.err.println("Errorea Fitxaketak kargatzean: " +
e.getMessage());
 e.printStackTrace();
 }

 // Fakturak
 try {

```

```

 // Hobekuntza: Bezeroaren izena erakutsi eskaera ID hutsaren ordez
 String sqlFakturak = "SELECT e.id_eskaera AS id_faktura,
e.faktura_zerbakia, " +
 "CONCAT(e.id_eskaera, ' - ', b.izena_edo_soziala) AS
eskaera, " +
 "e.data, e.faktura_url AS fitxategia_url " +
 "FROM eskaerak e " +
 "JOIN bezeroak b ON e.bezeroa_id = b.id_bezeroa " +
 "WHERE e.faktura_zerbakia IS NOT NULL AND
e.faktura_zerbakia != '' " +
 "ORDER BY e.id_eskaera DESC";
 PreparedStatement pstFa = konexioa.prepareStatement(sqlFakturak);
 DefaultTableModel mFa =
TaulaModelatzalea.ereduEraiki(pstFa.executeQuery());
 fakturaTaula.setModel(mFa);
 fakturaOrdenatzalea = new TableRowSorter<>(mFa);
 fakturaTaula.setRowSorter(fakturaOrdenatzalea);
 } catch (Exception e) {
 System.err.println("Errorea Fakturak kargatzean: " +
e.getMessage());
 e.printStackTrace();
 }

 // Hornitzaleak
 try {
 PreparedStatement pstH = konexioa.prepareStatement(
 "SELECT id_hornitzalea, izena_soziala, ifz_nan,
kontaktu_pertsona, helbidea, herria_id, posta_kodea, telefonoa FROM
hornitzaleak");
 DefaultTableModel mH =
TaulaModelatzalea.ereduEraiki(pstH.executeQuery());
 hornitzaleTaula.setModel(mH);
 hornitzaleOrdenatzalea = new TableRowSorter<>(mH);
 hornitzaleTaula.setRowSorter(hornitzaleOrdenatzalea);
 } catch (Exception e) {
 System.err.println("Errorea Hornitzaleak kargatzean: " +
e.getMessage());
 e.printStackTrace();
 }

 // Herriak
 try {
 PreparedStatement pstHe = konexioa.prepareStatement("SELECT * FROM
herriak");
 DefaultTableModel mHe =
TaulaModelatzalea.ereduEraiki(pstHe.executeQuery());
 herriaTaula.setModel(mHe);
 herriaOrdenatzalea = new TableRowSorter<>(mHe);
 herriaTaula.setRowSorter(herriaOrdenatzalea);
 } catch (Exception e) {
 System.err.println("Errorea Herriak kargatzean: " +
e.getMessage());
 e.printStackTrace();
 }

 if (unekoOrdenatzalea == null)
 unekoOrdenatzalea = langileOrdenatzalea;

```

```

 } catch (Exception e) {
 e.printStackTrace();
 }
 }

 /**
 * Taulak iragazi bilaketa testuaren arabera.
 */
 private void filtratu() {
 String testua = bilatuTestua.getText();
 if (unekoOrdenatzailea != null) {
 if (testua.trim().length() == 0)
 unekoOrdenatzailea.setRowFilter(null);
 else
 unekoOrdenatzailea.setRowFilter(RowFilter.regexFilter("(?i)" +
testua));
 }
 }

 /**
 * Hautatutako fitxaren elementua ezabatu.
 *
 * @param index Fitxaren indizea (0: Langileak, 1: Sailak, etc.).
 */
 private void ezabatuElementua(int index) {
 JTable unekoTaula = null;

 switch (index) {
 case 0:
 unekoTaula = langileTaula;
 break;
 case 1:
 unekoTaula = sailaTaula;
 break;
 case 2:
 unekoTaula = fitxaketaTaula;
 break;
 case 3:
 unekoTaula = fakturaTaula;
 break;
 case 4:
 unekoTaula = hornitzairenTaula;
 break;
 case 5:
 unekoTaula = herriaTaula;
 break;
 }

 if (unekoTaula == null || unekoTaula.getSelectedRow() == -1) {
 JOptionPane.showMessageDialog(this, "Hautatu errenkada bat
ezabatzeko.");
 return;
 }

 int errenkada = unekoTaula.getSelectedRow();
 int errenkadaModeloa = unekoTaula.convertRowIndexToModel(errenkada);
 }
}

```

```

Object idVal = unekoTaula.getModel().getValueAt(errengadaModeloa, 0);

 int aukera = JOptionPane.showConfirmDialog(this, "Ziur zaude ID " + idVal +
" ezabatu nahi duzula?", "Garbitu",
 JOptionPane.YES_NO_OPTION);
 if (aukera == JOptionPane.YES_OPTION) {
 try {
 int idInt = (idVal instanceof Number) ? ((Number) idVal).intValue()
 : Integer.parseInt(idVal.toString());
 switch (index) {
 case 0:
 langilea.langileaEzabatu(idInt);
 break;
 case 1:
 langilea.langileaSailaEzabatu(idInt);
 break;
 case 2:
 langilea.fitxaketaEzabatu(idInt);
 break;
 case 3:
 // Eskaera/Faktura ezabatu
 // Administrari-k ez du eskaeraEzabatu metodo propiorik,
baina Salmenta-k bai.
 // Erabil dezagun DB zuzena momentuz edo transferitu modelo
administraziora.
 // AdministrariLangilea-n eskaeraIkusi badago, baina
ezabatzea ez.
 // Hala ere, AdministrariLangilea-k dena kudeatzen duenez,
gehitu dugu modeloen
 // orain.
 try (Connection kon = DB_Konexioa.konektatu()) {
 PreparedStatement pst = kon.prepareStatement("DELETE
FROM eskaerak WHERE id_eskaera = ?");
 pst.setInt(1, idInt);
 pst.executeUpdate();
 }
 break;
 case 4:
 langilea.hornitzaleaEzabatu(idInt);
 break;
 case 5:
 langilea.herriaEzabatu(idInt);
 break;
 }
 datuakKargatuOsoa();
 JOptionPane.showMessageDialog(this, "Ezabatuta.");
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea ezabatzean: " +
e.getMessage());
 }
 }
}

/**
 * Elementu berri bat gehitu hautatutako fitxaren arabera.
 *
 * @param index Fitxaren indizea.

```

```

*/
private void gehituElementua(int index) {
 if (index == 0) { // Langileak
 JTextField izenaField = new JTextField();
 JTextField abizenaField = new JTextField();
 JTextField nanField = new JTextField();
 JTextField jaiotzaDataField = new JTextField("YYYY-MM-DD");
 JTextField telefonoaField = new JTextField();
 JTextField emailField = new JTextField();
 JPasswordField passField = new JPasswordField();

 // Sailak ComboBox
 JComboBox<ComboItem> sailaBox = new JComboBox<>();
 for (LangileSaila s : langilea.sailakGuztiakIkusi()) {
 sailaBox.addItem(new ComboItem(s.getIdSaila(), s.getIzena()));
 }

 JTextField helbideaField = new JTextField();

 // Herriak ComboBox
 JComboBox<ComboItem> herriaBox = new JComboBox<>();
 for (Herria h : langilea.herriakIkusi()) {
 herriaBox.addItem(new ComboItem(h.getIdHerria(), h.getIzena()));
 }
 JButton herriBerriaBotoia = new JButton("Herri Berria +");
 herriBerriaBotoia.addActionListener(e -> {
 JTextField hIzena = new JTextField();
 JTextField hLurraldea = new JTextField();
 JTextField hNazioa = new JTextField();
 Object[] hMsg = { "Herria:", hIzena, "Lurraldea:", hLurraldea,
 "Nazioa:", hNazioa };
 int hOpt = JOptionPane.showConfirmDialog(null, hMsg, "Herri Berria
Sortu",
 JOptionPane.OK_CANCEL_OPTION);
 if (hOpt == JOptionPane.OK_OPTION) {
 try {
 langilea.herrirriaSortu(hIzena.getText(),
 hLurraldea.getText(), hNazioa.getText());
 // Herriak kargatu berriro ComboBox-a eguneratzeko (logika
gehigarria behar izan
 // daiteke hemen
 // Baino gutxienez modeloko metodoari deitzen diogu.
 } catch (Exception ex) {
 JOptionPane.showMessageDialog(null, "Errorea herria
sortzean: " + ex.getMessage());
 }
 }
 });
 }

 JTextField postaKodeaField = new JTextField();

 JTextField hizkuntzaField = new JTextField("Euskara");
 JTextField saltoField = new JTextField();
 JCheckBox aktiboBox = new JCheckBox("Aktibo", true);
 JTextField ibanField = new JTextField();

 Object[] message = {

```

```

 "Izena:", izenaField,
 "Abizena:", abizenaField,
 "NAN:", nanField,
 "Jaiotza Data (YYYY-MM-DD):", jaiotzaDataField,
 "Telefonoa:", telefonoaField,
 "Email:", emailField,
 "Pasahitza:", passField,
 "Saila:", sailaBox,
 "Helbidea:", helbideaField,
 "Herria:", herriaBox,
 "", herriBerriaBotoia,
 "Posta Kodea:", postaKodeaField,
 "Hizkuntza (Euskara, Gaztelania, Frantsesa, Ingelesa):",
hizkuntzaField,
 "Salto Txartela UID:", saltoField,
 "Egoera:", aktiboBox,
 "IBAN:", ibanField
 };

 int option = JOptionPane.showConfirmDialog(null, message, "Langile
Berria", JOptionPane.OK_CANCEL_OPTION);
 if (option == JOptionPane.OK_OPTION) {
 try {
 ComboItem selectedSaila = (ComboItem)
sailaBox.getSelectedItem();
 ComboItem selectedHerria = (ComboItem)
herriaBox.getSelectedItem();

 langilea.langileaSortu(
 izenaField.getText(),
 abizenaField.getText(),
 nanField.getText(),
 emailField.getText(),
 new String(passField.getPassword()),
 selectedSaila != null ? selectedSaila.getId() : 0,
 helbideaField.getText(),
 selectedHerria != null ? selectedHerria.getId() : 0,
 postaKodeaField.getText(),
 telefonoaField.getText(),
 jaiotzaDataField.getText(),
 hizkuntzaField.getText(),
 saltoField.getText(),
 aktiboBox.isSelected(),
 ibanField.getText());

 datuakKargatuOsoa();
 JOptionPane.showMessageDialog(this, "Langilea gordeta.");
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " +
e.getMessage());
 }
 }
} else if (index == 1) { // Sailak
 JTextField izenaField = new JTextField();
 JTextField kokapenaField = new JTextField();
 JTextField deskribapenaField = new JTextField();
}

```

```

Object[] message = {
 "Izena:", izenField,
 "Kokapena:", kokapenaField,
 "Deskribapena:", deskribapenaField
};

int option = JOptionPane.showConfirmDialog(null, message, "Saila Berria", JOptionPane.OK_CANCEL_OPTION);
if (option == JOptionPane.OK_OPTION) {
 try {
 langilea.langileSailaSortu(izenField.getText(),
kokapenaField.getText(),
deskribapenaField.getText());
 datuakKargatu0soa();
 JOptionPane.showMessageDialog(this, "Saila gordeta.");
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " +
e.getMessage());
 }
}
} else if (index == 2) { // Fitxaketak
 JTextField langileaIdField = new JTextField();
 String[] motak = { "Sarrera", "Irteera" };
 JComboBox<String> motaBox = new JComboBox<>(motak);
 JTextField dataField = new JTextField("YYYY-MM-DD");
 JTextField orduaField = new JTextField("HH:MM:SS");

 Object[] message = {
 "Langilea ID:", langileaIdField, "Mota:", motaBox,
 "Data (Utzi hutsik gaurkorako):", dataField, "Ordua (Utzi hutsik orainerako):", orduaField
 };
}

int option = JOptionPane.showConfirmDialog(null, message, "Fitxaketa Berria", JOptionPane.OK_CANCEL_OPTION);
if (option == JOptionPane.OK_OPTION) {
 try (Connection kon = DB_Konexioa.konektatu()) {
 String sql = "INSERT INTO fitxaketak (langilea_id, mota, data, ordua) VALUES (?, ?, ?, ?)";
 // Data eta ordua hutsik badaude, NOW() erabili beharko genuke query-an edo
 // Java-n kudeatu.
 // Sinpletasunerako, balioak eskatuko ditugu edo defektuzkoak erabili.
 // SQL moldaketa:
 if (dataField.getText().contains("Y") || dataField.getText().isEmpty())
 sql = sql.replace("?, ?, ?, ?", "?, ?, CURRENT_DATE, CURRENT_TIME");
 }

 PreparedStatement pst = kon.prepareStatement("INSERT INTO fitxaketak (langilea_id, mota) VALUES (?, ?)");
 if (!dataField.getText().contains("Y") && !dataField.getText().isEmpty())
 pst = kon.prepareStatement("INSERT INTO fitxaketak (langilea_id, mota, data, ordua) VALUES (?, ?, ?, ?)");
}

```

```

ordua) VALUES (?, ?, ?, ?, ?)");
 pst.setString(3, dataField.getText());
 pst.setString(4, orduaField.getText());
 }

 pst.setInt(1, Integer.parseInt(langileaIdField.getText()));
 pst.setString(2, (String) motaBox.getSelectedItem());
 pst.executeUpdate();
 datuakKargatuOsoa();
 JOptionPane.showMessageDialog(this, "Fitxaketa gordeta.");
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " +
e.getMessage());
 }
 }
} else if (index == 4) { // Hornitzaireak
 JTextField izenaField = new JTextField();
 JTextField ifzField = new JTextField();
 JTextField emailField = new JTextField();

 Object[] message = {
 "Izena Soziala:", izenaField, "IFZ/NAN:", ifzField, "Emaila:",
emailField
 };
 int option = JOptionPane.showConfirmDialog(null, message, "Hornitzaire
Berria",
 JOptionPane.OK_CANCEL_OPTION);
 if (option == JOptionPane.OK_OPTION) {
 try (Connection kon = DB_Konexioa.konektatu()) {
 // Helbidea, herria eta posta kodea derrigorrezkoak dira DBan.
 // Defektuzko balioak jarriko ditugu hemen erabiltzaileak ez
duelako horiek
 // kudeatu nahi momentuz.
 String sql = "INSERT INTO hornitzaireak (izena_soziala,
ifz_nan, emaila, pasahitza, helbidea, herria_id, posta_kodea) VALUES (?, ?, ?, ?,
'1234', '-', 1, '-')";
 PreparedStatement pst = kon.prepareStatement(sql);
 pst.setString(1, izenaField.getText());
 pst.setString(2, ifzField.getText());
 pst.setString(3, emailField.getText());
 pst.executeUpdate();
 datuakKargatuOsoa();
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " +
e.getMessage());
 }
 }
} else if (index == 5) { // Herriak
 JTextField herriIzenaField = new JTextField();
 JTextField lurrealdeaField = new JTextField();
 JTextField nazioaField = new JTextField();

 Object[] message = { "Herria:", herriIzenaField, "Lurrealdea:",
lurrealdeaField, "Nazioa:", nazioaField };

 int option = JOptionPane.showConfirmDialog(null, message, "Herria
Berria", JOptionPane.OK_CANCEL_OPTION);

```

```

 if (option == JOptionPane.OK_OPTION) {
 try (Connection kon = DB_Konexioa.konektatu()) {
 String sql = "INSERT INTO herriak (izena, lurraldea, nazioa)
VALUES (?, ?, ?);"
 PreparedStatement pst = kon.prepareStatement(sql);
 pst.setString(1, herriIzenaField.getText());
 pst.setString(2, lurraldeaField.getText());
 pst.setString(3, nazioaField.getText());
 pst.executeUpdate();
 datuakKargatuOsoa();
 JOptionPane.showMessageDialog(this, "Herria gordeta.");
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea gordetzean: " +
e.getMessage());
 }
 }
 } else {
 JOptionPane.showMessageDialog(this, "Gehitu funtzioa oraindik ez dago
erabilgarri fitxa honetarako.");
 }
}

/**
 * Hautatutako elementua editatu.
 *
 * @param index Fitxaren indizea.
 */
private void editatuElementua(int index) {
 if (index == 0) { // Langileak
 if (langileTaula.getSelectedRow() == -1)
 return;
 int r = langileTaula.getSelectedRow();
 int rm = langileTaula.convertRowIndexToModel(r);
 Object id = langileTaula.getModel().getValueAt(rm, 0);

 try {
 int idInt = (id instanceof Number) ? ((Number) id).intValue() :
Integer.parseInt(id.toString());
 Langilea l = langilea.langileaIkusi(idInt);
 if (l != null) {
 JTextField izenaField = new JTextField(l.getIzena());
 JTextField abizenaField = new JTextField(l.getAbizena());
 JTextField nanField = new JTextField(l.getNan());
 JTextField jaiotzaDataField = new JTextField(
 l.getJaiotzaData() != null ?
l.getJaiotzaData().toString() : "");
 JTextField telefonoaField = new JTextField(l.getTelefona());
 JTextField emailField = new JTextField(l.getEmaila());
 JTextField helbideaField = new JTextField(l.getHelbidea());
 JTextField postaKodeaField = new JTextField(l.getPostaKodea());

 JComboBox<ComboItem> sailaBox = new JComboBox<>();
 ComboItem selectedSaila = null;
 for (LangileSaila s : langilea.sailakGuztiakIkusi()) {
 ComboItem item = new ComboItem(s.getIdSaila(),
s.getIzena());
 sailaBox.addItem(item);
 }
 }
 }
 }
}

```

```

 if (s.getIdSaila() == l.getIdSaila()) {
 selectedSaila = item;
 }
 }
 if (selectedSaila != null)
 sailaBox.setSelectedItem(selectedSaila);

 JComboBox<ComboItem> herriaBox = new JComboBox<>();
 ComboItem selectedHerria = null;
 for (Herria h : langilea.herrriakIkusi()) {
 ComboItem item = new ComboItem(h.getIdHerria(),
h.getIzena());
 herriaBox.addItem(item);
 if (h.getIdHerria() == l.getIdHerria()) {
 selectedHerria = item;
 }
 }
 if (selectedHerria != null)
 herriaBox.setSelectedItem(selectedHerria);

 JButton herriBerriaBotoia = new JButton("Herri Berria +");
 herriBerriaBotoia.addActionListener(e -> {
 JTextField hIzena = new JTextField();
 JTextField hLurraldea = new JTextField();
 JTextField hNazioa = new JTextField();
 Object[] hMsg = { "Herria:", hIzena, "Lurraldea:",
hLurraldea, "Nazioa:", hNazioa };
 int hOpt = JOptionPane.showConfirmDialog(null, hMsg, "Herri
Berria Sortu",
 JOptionPane.OK_CANCEL_OPTION);
 if (hOpt == JOptionPane.OK_OPTION) {
 try (Connection konH = DB_Konexioa.konektatu()) {
 String sql = "INSERT INTO herriak (izena,
lurraldea, nazioa) VALUES (?, ?, ?)";
 PreparedStatement pstH = konH.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
 pstH.setString(1, hIzena.getText());
 pstH.setString(2, hLurraldea.getText());
 pstH.setString(3, hNazioa.getText());
 pstH.executeUpdate();
 ResultSet rsKey = pstH.getGeneratedKeys();
 if (rsKey.next()) {
 int newItemId = rsKey.getInt(1);
 ComboItem newItem = new ComboItem(newItemId,
hIzena.getText());
 herriaBox.addItem(newItem);
 herriaBox.setSelectedItem(newItem);
 }
 } catch (Exception ex) {
 JOptionPane.showMessageDialog(null, "Errorea herria
sortzean: " + ex.getMessage());
 }
 }
 });
}

JTextField hizkuntzaField = new JTextField(l.getHizkuntza());
JPasswordField passFieldInput = new

```

```

JPasswordField(l.getPasahitz());
JTextField saltoField = new
JTextField(l.getSaltoTxartelaUid());
JCheckBox aktiboBox = new JCheckBox("Aktibo", l.isAktibo());
JTextField ibanField = new JTextField(l.getIban());

Object[] message = {
 "Izena:", izenaField,
 "Abizena:", abizenaField,
 "NAN:", nanField,
 "Jaiotza Data:", jaiotzaDataField,
 "Telefonoa:", telefonoaField,
 "Email:", emailField,
 "Pasahitza:", passFieldInput,
 "Saila:", sailaBox,
 "Helbidea:", helbideaField,
 "Posta Kodea:", postaKodeaField,
 "Herria:", herriaBox,
 "", herriBerriaBotoia,
 "Hizkuntza:", hizkuntzaField,
 "Salto Txartela UID:", saltoField,
 "Egoera:", aktiboBox,
 "IBAN:", ibanField
};

int option = JOptionPane.showConfirmDialog(null, message,
"Editatu Langilea",
JOptionPane.OK_CANCEL_OPTION);
if (option == JOptionPane.OK_OPTION) {
 ComboItem selSaila = (ComboItem)
sailaBox.getSelectedItem();
 ComboItem selHerria = (ComboItem)
herriaBox.getSelectedItem();

 langilea.langileaEditatu(
 idInt,
 izenaField.getText(),
 abizenaField.getText(),
 nanField.getText(),
 emailField.getText(),
 selSaila != null ? selSaila.getId() : 0,
 helbideaField.getText(),
 selHerria != null ? selHerria.getId() : 0,
 postaKodeaField.getText(),
 telefonoaField.getText(),
 jaiotzaDataField.getText(),
 hizkuntzaField.getText(),
 new String(passFieldInput.getPassword()),
 saltoField.getText(),
 aktiboBox.isSelected(),
 ibanField.getText());

 datuakKargatuOsoa();
}
}

} catch (Exception e) {
JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
}

```

```

 }
 } else if (index == 1) { // Sailak
 if (sailaTaula.getSelectedRow() == -1)
 return;
 int r = sailaTaula.getSelectedRow();
 int rm = sailaTaula.convertRowIndexToModel(r);
 Object id = sailaTaula.getModel().getValueAt(rm, 0);

 try (Connection kon = DB_Konexioa.konektatu()) {
 PreparedStatement pstSel = kon.prepareStatement("SELECT * FROM
langile_sailak WHERE id_saila = ?");
 pstSel.setObject(1, id);
 ResultSet rs = pstSel.executeQuery();
 if (rs.next()) {
 JTextField izenaField = new JTextField(rs.getString("izena"));
 JTextField kokapenaField = new
JTextField(rs.getString("kokapena"));
 JTextField deskribapenaField = new
JTextField(rs.getString("deskribapena"));

 Object[] message = {
 "Izena:", izenaField,
 "Kokapena:", kokapenaField,
 "Deskribapena:", deskribapenaField
 };

 int option = JOptionPane.showConfirmDialog(null, message,
"Editatu Saila",
 JOptionPane.OK_CANCEL_OPTION);
 if (option == JOptionPane.OK_OPTION) {
 PreparedStatement pstUpd = kon.prepareStatement(
 "UPDATE langile_sailak SET izena = ?, kokapena = ?,
deskribapena = ? WHERE id_saila = ?");
 pstUpd.setString(1, izenaField.getText());
 pstUpd.setString(2, kokapenaField.getText());
 pstUpd.setString(3, deskribapenaField.getText());
 pstUpd.setObject(4, id);
 pstUpd.executeUpdate();
 datuakKargatuOsoa();
 }
 }
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
 }
 } else if (index == 2) { // Fitxaketak
 if (fitxaketaTaula.getSelectedRow() == -1)
 return;
 int r = fitxaketaTaula.getSelectedRow();
 int rm = fitxaketaTaula.convertRowIndexToModel(r);
 Object id = fitxaketaTaula.getModel().getValueAt(rm, 0);

 try (Connection kon = DB_Konexioa.konektatu()) {
 PreparedStatement pstSel = kon.prepareStatement("SELECT * FROM
fitxaketak WHERE id_fitzaketa = ?");
 pstSel.setObject(1, id);
 ResultSet rs = pstSel.executeQuery();
 if (rs.next()) {

```

```

 JTextField langileaIdField = new
JTextField(String.valueOf(rs.getInt("langilea_id")));
 String[] motak = { "Sarrera", "Irteera" };
 JComboBox<String> motaBox = new JComboBox<>(motak);
 motaBox.setSelectedItem(rs.getString("mota"));
 JTextField dataField = new JTextField(rs.getString("data"));
 JTextField orduaField = new JTextField(rs.getString("ordua"));

 Object[] message = { "Langilea ID:", langileaIdField, "Mota:",
motaBox, "Data:", dataField,
 "Ordua:", orduaField };
 int option = JOptionPane.showConfirmDialog(null, message,
"Editatu Fitxaketa",
 JOptionPane.OK_CANCEL_OPTION);
 if (option == JOptionPane.OK_OPTION) {
 PreparedStatement pstUpd = kon.prepareStatement(
 "UPDATE fitxaketak SET langilea_id = ?, mota = ?,
data = ?, ordua = ? WHERE id_fitxaketa = ?");
 pstUpd.setInt(1,
Integer.parseInt(langileaIdField.getText()));
 pstUpd.setString(2, (String) motaBox.getSelectedItem());
 pstUpd.setString(3, dataField.getText());
 pstUpd.setString(4, orduaField.getText());
 pstUpd.setObject(5, id);
 pstUpd.executeUpdate();
 datuakKargatuOsoa();
 }
 }
} catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
}
} else if (index == 4) { // Hornitzailak
 if (hornitzailaTaula.getSelectedRow() == -1)
 return;
 int r = hornitzailaTaula.getSelectedRow();
 int rm = hornitzailaTaula.convertRowIndexToModel(r);
 Object id = hornitzailaTaula.getModel().getValueAt(rm, 0);

 try (Connection kon = DB_Konexioa.konektatu()) {
 PreparedStatement pstSel = kon
 .prepareStatement("SELECT * FROM hornitzailak WHERE
id_hornitzaila = ?");
 pstSel.setObject(1, id);
 ResultSet rs = pstSel.executeQuery();
 if (rs.next()) {
 JTextField izenaField = new
JTextField(rs.getString("izena_soziala"));
 JTextField ifzField = new JTextField(rs.getString("ifz_nan"));
 JTextField kontaktuField = new
JTextField(rs.getString("kontaktu_pertsona"));
 JTextField helbideaField = new
JTextField(rs.getString("helbidea"));
 JTextField postaKodeaField = new
JTextField(rs.getString("posta_kodea"));
 JTextField telefonoaField = new
JTextField(rs.getString("telefonoa"));
 JTextField emailField = new JTextField(rs.getString("emaila"));

```

```

JComboBox<ComboItem> herriaBox = new JComboBox<>();
PreparedStatement pstHerria = kon.prepareStatement("SELECT
id_herria, izena FROM herriak");
ResultSet rsH = pstHerria.executeQuery();
ComboItem selectedHerria = null;
int currentHerriaId = rs.getInt("herria_id");
while (rsH.next()) {
 ComboItem item = new ComboItem(rsH.getInt("id_herria"),
rsH.getString("izena"));
 herriaBox.addItem(item);
 if (item.getId() == currentHerriaId) {
 selectedHerria = item;
 }
}
if (selectedHerria != null)
 herriaBox.setSelectedItem(selectedHerria);

JButton herriBerriaBotoia = new JButton("Herri Berria +");
herriBerriaBotoia.addActionListener(e -> {
 JTextField hIzena = new JTextField();
 JTextField hLurraldea = new JTextField();
 JTextField hNazioa = new JTextField();
 Object[] hMsg = { "Herria:", hIzena, "Lurraldea:",
hLurraldea, "Nazioa:", hNazioa };
 int hOpt = JOptionPane.showConfirmDialog(null, hMsg, "Herri
Berria Sortu",
 JOptionPane.OK_CANCEL_OPTION);
 if (hOpt == JOptionPane.OK_OPTION) {
 try (Connection konH = DB_Konexioa.konektatu()) {
 String sql = "INSERT INTO herriak (izena,
lurraldea, nazioa) VALUES (?, ?, ?)";
 PreparedStatement pstH = konH.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
 pstH.setString(1, hIzena.getText());
 pstH.setString(2, hLurraldea.getText());
 pstH.setString(3, hNazioa.getText());
 pstH.executeUpdate();
 ResultSet rsKey = pstH.getGeneratedKeys();
 if (rsKey.next()) {
 int newId = rsKey.getInt(1);
 ComboItem newItem = new ComboItem(newId,
hIzena.getText());
 herriaBox.addItem(newItem);
 herriaBox.setSelectedItem(newItem);
 }
 } catch (Exception ex) {
 JOptionPane.showMessageDialog(null, "Errorea herria
sortzean: " + ex.getMessage());
 }
 }
});

Object[] message = {
 "Izena Soziala:", izenaField,
 "IFZ/NAN:", ifzField,
 "Kontaktu Pertsona:", kontaktuField,
}

```

```

 "Helbidea:", helbideaField,
 "Herria:", herriaBox,
 "", herriBerriaBotoia,
 "Posta Kodea:", postaKodeaField,
 "Telefonoa:", telefonoaField,
 "Emaila:", emailField
 };

 int option = JOptionPane.showConfirmDialog(null, message,
"Editatu Hornitzzailea",
JOptionPane.OK_CANCEL_OPTION);
 if (option == JOptionPane.OK_OPTION) {
 ComboItem selHerria = (ComboItem)
herriaBox.getSelectedItem();
 PreparedStatement pst = kon.prepareStatement(
 "UPDATE hornitzzaileak SET izena_soziala = ?,"
ifz_nan = ?, kontaktu_pertsona = ?, helbidea = ?, herria_id = ?, posta_kodea = ?,
telefonoa = ?, emaila = ? WHERE id_hornitzzailea = ?");
 pst.setString(1, izenaField.getText());
 pst.setString(2, ifzField.getText());
 pst.setString(3, kontakuField.getText());
 pst.setString(4, helbideaField.getText());
 pst.setInt(5, selHerria != null ? selHerria.getId() : 0);
 pst.setString(6, postaKodeaField.getText());
 pst.setString(7, telefonoaField.getText());
 pst.setString(8, emailField.getText());
 pst.setObject(9, id);
 pst.executeUpdate();
 datuakKargatuOsoa();
 }
}
} catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
}
} else if (index == 5) { // Herriak
 if (herriaTaula.getSelectedRow() == -1) {
 JOptionPane.showMessageDialog(this, "Hautatu herria editatzeko.");
 return;
 }
 int r = herriaTaula.getSelectedRow();
 int rm = herriaTaula.convertRowIndexToModel(r);
 Object id = herriaTaula.getModel().getValueAt(rm, 0);

 try (Connection kon = DB_Konexioa.konektatu()) {
 PreparedStatement pstSel = kon.prepareStatement("SELECT * FROM"
herriak WHERE id_herria = ?");
 pstSel.setObject(1, id);
 ResultSet rs = pstSel.executeQuery();
 if (rs.next()) {
 JTextField izenaField = new JTextField(rs.getString("izena"));
 JTextField lurrealdeaField = new
JTextField(rs.getString("lurrealdea"));
 JTextField nazioaField = new
JTextField(rs.getString("nazioa"));

 Object[] message = { "Herria:", izenaField, "Lurrealdea:",
lurrealdeaField, "Nazioa:", nazioaField };

```

```

 int option = JOptionPane.showConfirmDialog(null, message,
"Editatu Herria",
 JOptionPane.OK_CANCEL_OPTION);
 if (option == JOptionPane.OK_OPTION) {
 PreparedStatement pstUpd = kon.prepareStatement(
 "UPDATE herriak SET izena = ?, lurrealdea = ?,
nazioa = ? WHERE id_herria = ?");
 pstUpd.setString(1, izenaField.getText());
 pstUpd.setString(2, lurrealdeaField.getText());
 pstUpd.setString(3, nazioaField.getText());
 pstUpd.setObject(4, id);
 pstUpd.executeUpdate();
 datuakKargatuOsoa();
 JOptionPane.showMessageDialog(this, "Herria eguneratuta.");
 }
 }
} catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea egunerezan: " +
e.getMessage());
}
}

/**
 * Hautatutako fakturaren PDF fitxategia ireki.
 */
private void ikusiFaktura() {
 if (fakturaTaula.getSelectedRow() == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu faktura bat
zerrendatik.");
 return;
 }

 int r = fakturaTaula.getSelectedRow();
 int rm = fakturaTaula.convertRowIndexToModel(r);
 // "Fitxategia URL" zutabea 4. posizioan dago (0-tik hasita indizea 3
beharko
 // luke,
 // baina DBko zutabeen ordenaren arabera: id, zenbakia, eskaera_id, data,
url)
 // Taula eredu: id, zenbakia, eskaera_id, data, url -> URL index 4
 // Egiaztatu TaulaModelatzalea.ereduEraiki-k zutabeak nola jartzen
dituen.
 // Normalean SELECT * ordenan. SELECT * FROM bezero_fakturak -> id_faktura,
 // faktura_zenbakia, eskaera_id, data, fitxategia_url.
 // Beraz index 4.

 Object urlObj = fakturaTaula.getModel().getValueAt(rm, 4);

 if (urlObj == null || urlObj.toString().isEmpty()) {
 JOptionPane.showMessageDialog(this, "Faktura honek ez du fitxategirik
lotuta.");
 return;
 }

 String url = urlObj.toString();
 try {

```

```

 java.io.File file = new java.io.File(url);
 if (file.exists()) {
 java.awt.Desktop.getDesktop().open(file);
 } else {
 JOptionPane.showMessageDialog(this, "Fitxategia ez da aurkitu: " +
url);
 }
 } catch (java.io.IOException e) {
 JOptionPane.showMessageDialog(this, "Errorea fitxategia irekitzean: " +
e.getMessage());
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
 }
}

/**
 * Saioa itxi eta hasierako pantailara itzuli.
 */
private void saioaItxi() {
 if (JOptionPane.showConfirmDialog(this, "Ziur zaude?", "Saioa Itxi",
JOptionPane.YES_NO_OPTION) == 0) {
 dispose();
 new SaioaHastekoPanela().setVisible(true);
 }
}

// Helper class for ComboBox items
/**
 * ComboBox-erako klase lagunzailea.
 */
/**
 * Hautatutako langilearen kurrikulumua ikusteko metodoa.
 */
private void ikusiKurrikuluma() {
 int aukeratutakoLerroa = langileTaula.getSelectedRow();
 if (aukeratutakoLerroa == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu langile bat kurrikuluma
ikusteko.");
 return;
 }

 aukeratutakoLerroa =
langileTaula.convertRowIndexToModel(aukeratutakoLerroa);
 Object idObj = langileTaula.getModel().getValueAt(aukeratutakoLerroa, 0);
 int idLangilea = (idObj instanceof Number) ? ((Number) idObj).intValue()
 : Integer.parseInt(idObj.toString());

 try {
 langilea.kurrikulumaIkusi(idLangilea);
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
 }
}

private static class ComboItem {
 private int id;
 private String label;

```

```

public ComboItem(int id, String label) {
 this.id = id;
 this.label = label;
}

public int getId() {
 return id;
}

@Override
public String toString() {
 return label;
}
}
}

```

## MenuLogistika.java

```

package ui;

import db.DB_Konexioa;
import model.*;

import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.io.File;

/**
 * MenuLogistika klasea.
 * Biltegi langilearen interfaze nagusia.
 * Sarrerak, biltegiak, produktuak, eskaerak eta hornitzaleak kudeatzeko
 * aukerak.
 */
public class MenuLogistika extends JFrame {

 private static final long serialVersionUID = 1L;
 private JPanel edukiPanela;
 private JTextField bilatuTestua;

 // Erabiltzailea (OOP)
 private BiltegiLangilea langilea;

 // Fitxaketa informazioa erakusteko etiketa
 private JLabel fitxaketaInfoEtiketa;

```

```
// Taulak
private JTable sarreraTaula;
private JTable biltegiTaula;
private JTable produktuTaula;

// Iragazkia
private JComboBox<String> egoeraIragazkia;

// Fitxak
private JTabbedPane pestainaPanela;

// Sorters
private TableRowSorter<DefaultTableModel> sarreraOrdenatzailea;
private TableRowSorter<DefaultTableModel> biltegiOrdenatzailea;
private TableRowSorter<DefaultTableModel> produktuOrdenatzailea;

// Eskaerak
private JTable eskaeraTaula;
private TableRowSorter<DefaultTableModel> eskaeraOrdenatzailea;
private JComboBox<String> eskaeraEgoeraIragazkia;

/**
 * Eraikitzaileak egeneratua.
 */
/**
 * MenuLogistika eraikitzailea.
 *
 * @param oinarrizkoLangilea Saioa hasi duen langilea.
 */
public MenuLogistika(Langilea oinarrizkoLangilea) {
 this.langilea = new BiltegiLangilea(oinarrizkoLangilea);
 pantailaPrestatu();
}

/**
 * Pantailaren osagaiak prestatu.
 */
private void pantailaPrestatu() {
 setTitle("Birtek - LOGISTIKA");
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 setBounds(100, 100, 1150, 700);

 edukiPanela = new AtzealdekoPanela();
 edukiPanela.setBorder(new EmptyBorder(5, 5, 5, 5));
 setContentPane(edukiPanela);
 edukiPanela.setLayout(new BorderLayout(0, 0));

 // --- GOIKO PANELA ---
 JPanel goikoPanela = new JPanel(new BorderLayout());
 private static final long serialVersionUID = 1L;

 @Override
 protected void paintComponent(Graphics g) {
 super.paintComponent(g);
 g.setColor(new Color(255, 255, 255, 220));
 g.fillRect(0, 0, getWidth(), getHeight());
 }
}
```

```

};

goikoPanela.setOpaque(false);
goikoPanela.setBorder(BorderFactory.createCompoundBorder(
 BorderFactory.createLineBorder(new Color(0, 128, 128), 1),
 BorderFactory.createEmptyBorder(10, 15, 10, 15)));

edukiPanela.add(goikoPanela, BorderLayout.NORTH);

// EZKERRA: Bilatzailea
 JPanel bilatzailePanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
bilatzailePanela.setOpaque(false);
JLabel bilatuEtiketa = new JLabel("Bilatu: ");
bilatuEtiketa.setFont(new Font("SansSerif", Font.BOLD, 12));
bilatzailePanela.add(bilatuEtiketa);

bilatuTestua = new JTextField();
bilatuTestua.setColumns(20);
bilatuTestua.addKeyListener(new KeyAdapter() {
 @Override
 public void keyReleased(KeyEvent e) {
 filtratu();
 }
});
bilatzailePanela.add(bilatuTestua);

 JButton eguneratuBotoia = new JButton("Eguneratu");
eguneratuBotoia.setBackground(new Color(173, 216, 230)); // Light Blue
eguneratuBotoia.addActionListener(e -> {
 sarreraDatuakKargatu();
 biltegiDatuakKargatu();
 produktuDatuakKargatu();
 eskaeraDatuakKargatu();
});
bilatzailePanela.add(eguneratuBotoia);
goikoPanela.add(bilatzailePanela, BorderLayout.WEST);

// ESKUINALDEA: Erabiltzailea + Fitxaketa + Saioa Itxi
 JPanel erabiltzaileInfoPanela = new JPanel(new GridBagLayout());
erabiltzaileInfoPanela.setOpaque(false);

 JLabel erabiltzaileEtiketa = new JLabel(
 langilea.getIzena() + " " + langilea.getAbizena());
erabiltzaileEtiketa.setFont(new Font("SansSerif", Font.BOLD, 14));
erabiltzaileEtiketa.setForeground(new Color(0, 102, 102));

 GridBagConstraints gbcUser = new GridBagConstraints();
gbcUser.insets = new Insets(0, 10, 0, 10);
gbcUser.fill = GridBagConstraints.VERTICAL;
gbcUser.gridx = 0;
gbcUser.gridy = 0;
erabiltzaileInfoPanela.add(erabiltzaileEtiketa, gbcUser);

// Fitxaketa Panela (Etiketa)
fitxaketaInfoEtiketa = new JLabel("Kargatzen...");
fitxaketaInfoEtiketa.setFont(new Font("SansSerif", Font.PLAIN, 10));
GridBagConstraints gbcLabel = new GridBagConstraints();
gbcLabel.gridx = 0;

```

```

gbcLabel.gridx = 1;
erabiltzaileInfoPanela.add(fitxaketaInfoEtiketa, gbcLabel);

// ... Fitxaketa botoiak ...
JPanel botoiPanela = new JPanel(new FlowLayout(FlowLayout.CENTER, 5, 0));
botoiPanela.setOpaque(false);

JButton sarreraBotoia = new JButton("Sarrera");
sarreraBotoia.addActionListener(e -> fitxatu("Sarrera"));

JButton irteeraBotoia = new JButton("Irteera");
irteeraBotoia.addActionListener(e -> fitxatu("Irteera"));

JButton historialBotoia = new JButton("Historiala");
historialBotoia.addActionListener(e -> ikusiFitxaketaHistoriala());

JButton nireDatuakBotoia = new JButton("Nire Datuak");
nireDatuakBotoia.addActionListener(e -> irekiNireDatuakEditatu());

botoiPanela.add(sarreraBotoia);
botoiPanela.add(irteeraBotoia);
botoiPanela.add(historialBotoia);
botoiPanela.add(nireDatuakBotoia);

GridBagConstraints gbcBtns = new GridBagConstraints();
gbcBtns.gridx = 1;
gbcBtns.gridy = 0;
gbcBtns.gridheight = 2;
erabiltzaileInfoPanela.add(botoiPanela, gbcBtns);

// Logout botoia
JButton saioaItxiBotoia = new JButton("Saioa Itxi");
saioaItxiBotoia.setBackground(new Color(220, 20, 60));
saioaItxiBotoia.setForeground(new Color(0, 0, 0));
saioaItxiBotoia.setFont(new Font("SansSerif", Font.BOLD, 12));
saioaItxiBotoia.addActionListener(e -> saioaItxi());

GridBagConstraints gbcLogout = new GridBagConstraints();
gbcLogout.insets = new Insets(0, 10, 0, 10);
gbcLogout.fill = GridBagConstraints.VERTICAL;
gbcLogout.gridx = 2;
gbcLogout.gridy = 0;
gbcLogout.gridheight = 2;
erabiltzaileInfoPanela.add(saioaItxiBotoia, gbcLogout);

goikoPanela.add(erabiltzaileInfoPanela, BorderLayout.EAST);

// PESTAINA PANELA (JTabbedPane)
pestainaPanela = new JTabbedPane(JTabbedPane.TOP);
edukiPanela.add(pestainaPanela, BorderLayout.CENTER);

// Tab-ak sortu
sarreraTabSortu();
biltegiTabSortu();
produktuTabSortu();
eskaeraTabSortu();

```

```

pestainaPanela.addChangeListener(e -> {
 bilatuTestua.setText("");
 int index = pestainaPanela.getSelectedIndex();
 if (index == 0)
 sarreraDatuakKargatu();
 else if (index == 1)
 biltegiDatuakKargatu();
 else if (index == 2)
 produktuDatuakKargatu();
 else if (index == 3)
 eskaeraDatuakKargatu();
});

// Hasierako karga
if (!java.beans.Beans.isDesignTime()) {
 sarreraDatuakKargatu();
 biltegiDatuakKargatu();
 produktuDatuakKargatu();
 eguneratuFitxaketaEgoera();
}
}

// --- FITXAKETA LOGIKA ---
/**
 * Fitxaketa bat egin (Sarrera/Irteera).
 *
 * @param mota Fitxaketa mota.
 */
private void fitxatu(String mota) {
 try {
 langilea.fitxatu(mota);
 eguneratuFitxaketaEgoera();
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, e.getMessage(), "Errorea",
JOptionPane.WARNING_MESSAGE);
 }
}

/**
 * Fitxaketa egoera eguneratu interfazean.
 */
private void eguneratuFitxaketaEgoera() {
 fitxaketaInfoEtiketa.setText(langilea.getFitxaketaEgoera());
 if (fitxaketaInfoEtiketa.getText().contains("BARRUAN")) {
 fitxaketaInfoEtiketa.setForeground(new Color(0, 100, 0));
 } else {
 fitxaketaInfoEtiketa.setForeground(new Color(200, 0, 0));
 }
}

/**
 * Fitxaketa historia erakutsi.
 */
private void ikusiFitxaketaHistoriala() {
 JDialog elkarrizketa = new JDialog(this, "Nire Fitxaketa Historiala",
true);
 elkarrizketa.setSize(500, 400);
}

```

```

elkarritzeta.setLocationRelativeTo(this);
elkarritzeta.setLayout(new BorderLayout());
String[] zutabeak = { "Mota", "Data", "Ordua" };
DefaultTableModel eredua = new DefaultTableModel(zutabeak, 0);
JTable taula = new JTable(eredua);
taula.getTableHeader().setFont(new Font("SansSerif", Font.BOLD, 12));
taula.setRowHeight(25);
elkarritzeta.add(new JScrollPane(taula), BorderLayout.CENTER);

java.util.List<Fitxaketa> zerrenda = langilea.nireFitxaketakIkusi();
for (Fitxaketa f : zerrenda) {
 eredua.addRow(new Object[] { f.getMota(), f.getData(), f.getOrdua() });
}

JButton itxiBotoia = new JButton("Itxi");
itxiBotoia.addActionListener(e -> elkarritzeta.dispose());
 JPanel botoiPanela = new JPanel();
botoiPanela.add(itxiBotoia);
elkarritzeta.add(botoiPanela, BorderLayout.SOUTH);
elkarritzeta.setVisible(true);
}

/**
 * Norberaren datuak editatzeko leihoa ireki.
 */
private void irekiNireDatuakEditatu() {
 NireDatuakDialog dialog = new NireDatuakDialog(this, langilea);
 dialog.setVisible(true);
}

// --- SARRERAK FITXA ---
// --- SARRERAK FITXA ---
/**
 * Sarreren fitxa ("tab") sortu eta konfiguratu.
 */
private void sarreraTabSortu() {
 JPanel sarreraPanela = new JPanel(new BorderLayout());
 sarreraPanela.setOpaque(false);

 JPanel goikoAukeraPanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
 goikoAukeraPanela.setOpaque(false);
 goikoAukeraPanela.add(new JLabel("Egoera Iragazi:"));

 egoeraIragazkia = new JComboBox<>();
 egoeraIragazkia.addItem("Denak");
 egoeraIragazkia.addItem("Bidean");
 egoeraIragazkia.addItem("Jasota");
 egoeraIragazkia.addActionListener(e -> sarreraDatuakKargatu());
 goikoAukeraPanela.add(egoeraIragazkia);

 JButton eguneratuBotoia = new JButton("Egeneratu Zerrenda");
 eguneratuBotoia.addActionListener(e -> sarreraDatuakKargatu());
 goikoAukeraPanela.add(eguneratuBotoia);

 // BOTOI BERRIAK
 JButton ikusiLerroakBotoia = new JButton("Ikusi Lerroak");
 ikusiLerroakBotoia.addActionListener(e -> ikusiSarreraLerroak());
}

```

```

goikoAukeraPanela.add(ikusiLerroakBotoia);

JButton editatuBotoia = new JButton("Editatu Egoera");
editatuBotoia.addActionListener(e -> editatuSarrera());
goikoAukeraPanela.add(editatuBotoia);

JButton ezabatuBotoia = new JButton("Ezabatu");
ezabatuBotoia.setBackground(new Color(255, 100, 100));
ezabatuBotoia.addActionListener(e -> ezabatuSarrera());
goikoAukeraPanela.add(ezabatuBotoia);

JButton sarreraBerriaBotoia = new JButton("Sarrera Berria +");
sarreraBerriaBotoia.setBackground(new Color(50, 205, 50));
sarreraBerriaBotoia.setFont(new Font("SansSerif", Font.BOLD, 12));
sarreraBerriaBotoia.addActionListener(e -> {
 SarreraBerriaDialog dialog = new SarreraBerriaDialog(this, langilea);
 dialog.setVisible(true);
 sarreraDatuakKargatu(); // Freskatu
});
goikoAukeraPanela.add(sarreraBerriaBotoia);

sarreraPanela.add(goikoAukeraPanela, BorderLayout.NORTH);

sarreraTaula = new JTable();
JScrollPane ruli = new JScrollPane(sarreraTaula);
sarreraPanela.add(ruli, BorderLayout.CENTER);

pestainaPanela.addTab("Sarrerak", null, sarreraPanela, null);
}

// --- BILTEGIAK FITXA ---
/**
 * Biltegien fitxa sortu.
 */
private void biltegiTabSortu() {
 JPanel biltegiPanela = new JPanel(new BorderLayout());
 biltegiPanela.setOpaque(false);

 JPanel botoiPanela = new JPanel();
 botoiPanela.setOpaque(false);
 JButton sortuBotoia = new JButton("Sortu Biltegia");
 JButton aldatuBotoia = new JButton("Aldatu");
 JButton ezabatuBotoia = new JButton("Ezabatu");

 sortuBotoia.addActionListener(e -> sortuBiltegia());
 aldatuBotoia.addActionListener(e -> aldatuBiltegia());
 ezabatuBotoia.addActionListener(e -> ezabatuBiltegia());

 botoiPanela.add(sortuBotoia);
 botoiPanela.add(aldatuBotoia);
 botoiPanela.add(ezabatuBotoia);
 biltegiPanela.add(botoiPanela, BorderLayout.NORTH);

 biltegiTaula = new JTable();
 biltegiPanela.add(new JScrollPane(biltegiTaula), BorderLayout.CENTER);

 pestainaPanela.addTab("Biltegiak", null, biltegiPanela, null);
}

```

```

}

// --- PRODUKTUAK FITXA ---
/**
 * Produktuen fitxa sortu.
 */
private void produktuTabSortu() {
 JPanel produktuPanela = new JPanel(new BorderLayout());
 produktuPanela.setOpaque(false);

 JPanel botoiPanela = new JPanel();
 botoiPanela.setOpaque(false);
 JButton aldatuKokapenaBotoia = new JButton("Aldatu Biltegia");
 JButton jasoBotoia = new JButton("Markatu 'Jasota'");
 JButton bideanBotoia = new JButton("Markatu 'Bidean'");

 aldatuKokapenaBotoia.addActionListener(e -> aldatuProduktuarenBiltegia());
 jasoBotoia.addActionListener(e -> markatuProduktuaJasota());
 bideanBotoia.addActionListener(e -> markatuProduktuaBidean());

 botoiPanela.add(aldatuKokapenaBotoia);
 botoiPanela.add(jasoBotoia);
 botoiPanela.add(bideanBotoia);

 JButton oharraBotoia = new JButton("Editatu Oharra");
 oharraBotoia.addActionListener(e -> editatuProduktuOharra());
 botoiPanela.add(oharraBotoia);

 produktuPanela.add(botoiPanela, BorderLayout.NORTH);

 produktuTaula = new JTable();
 produktuPanela.add(new JScrollPane(produktuTaula), BorderLayout.CENTER);

 pestainaPanela.addTab("Produktuak eta Kokapena", null, produktuPanela,
null);
}

// --- ESKAERAK FITXA ---
/**
 * Eskaeren fitxa sortu.
 */
private void eskaeraTabSortu() {
 JPanel eskaeraPanela = new JPanel(new BorderLayout());
 eskaeraPanela.setOpaque(false);

 JPanel goikoAukeraPanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
 goikoAukeraPanela.setOpaque(false);
 goikoAukeraPanela.add(new JLabel("Egoera Iragazi:"));

 eskaeraEgoeraIragazkia = new JComboBox();
 eskaeraEgoeraIragazkia.addItem("Denak");
 eskaeraEgoeraIragazkia.addItem("Prestatzen");
 eskaeraEgoeraIragazkia.addItem("Osatua/Bidalita");
 eskaeraEgoeraIragazkia.addActionListener(e -> eskaeraDatuakKargatu());
 goikoAukeraPanela.add(eskaeraEgoeraIragazkia);

 JButton eguneraBotoia = new JButton("Eguneratu");

```

```

eguneratuBotoia.addActionListener(e -> eskaeraDatuakKargatu());
goikoAukeraPanela.add(eguneratuBotoia);

JButton lerroakBotoia = new JButton("Ikusi Lerroak / Kudeatu");
lerroakBotoia.setFont(new Font("SansSerif", Font.BOLD, 12));
lerroakBotoia.setBackground(new Color(200, 230, 255));
lerroakBotoia.addActionListener(e -> ikusiEskaeraLerroak());
goikoAukeraPanela.add(lerroakBotoia);

eskaeraPanela.add(goikoAukeraPanela, BorderLayout.NORTH);

eskaeraTaula = new JTable();
JScrollPane ruli = new JScrollPane(eskaeraTaula);
eskaeraPanela.add(ruli, BorderLayout.CENTER);

pestainaPanela.addTab("Eskaerak", null, eskaeraPanela, null);
}

// --- SARRERA BERRIA FITXA ---
/**
 * Sarrera berria egiteko fitxa sortu.
 */

/**
 * Sarreren datuak taulan kargatu.
 */
private void sarreraDatuakKargatu() {
 try {
 String filter = (String) egoeraIragazkia.getSelectedItem();
 if ("Denak".equals(filter))
 filter = ""; // Or handle null inside logic if preferred
 java.util.List<Object[]> data = langilea.produktuSarrerakIkusi(filter);

 String[] colNames = { "ID", "Hornitzalea", "Data", "Egoera" };
 DefaultTableModel eredua = new DefaultTableModel(colNames, 0);
 for (Object[] row : data)
 eredua.addRow(row);
 }
 sarreraTaula.setModel(eredua);
 sarreraOrdenatzailea = new TableRowSorter<>(eredua);
 sarreraTaula.setRowSorter(sarreraOrdenatzailea);
} catch (SQLException e) {
 e.printStackTrace();
}
}

/**
 * Biltegien datuak taulan kargatu.
 */
private void biltegiDatuakKargatu() {
 String sql = "SELECT id_biltegia, izena, biltegi_sku FROM biltegiak";
 try (Connection con = DB_Konexioa.konektatu(); PreparedStatement pst =
con.prepareStatement(sql)) {
 DefaultTableModel eredua =
TaulaModelatzalea.ereduaEraiki(pst.executeQuery());
 biltegiTaula.setModel(eredua);
 biltegiOrdenatzailea = new TableRowSorter<>(eredua);
 }
}

```

```

 biltegiTaula.setRowSorter(biltegiOrdenatzailea);
 } catch (Exception e) {
 e.printStackTrace();
 }
}

/***
 * Produktuen datuak taulan kargatu.
 */
private void produktuDatuakKargatu() {
 String sql = "SELECT p.id_produktua, p.izena AS Produktua, b.izena AS
Biltegia, s.id_sarrera AS 'Sarrera ID', sl.sarrera_lerro_egoera AS Egoera, s.data
AS 'Sarrera Data', sl.id_sarrera_lerroa, p.produktu_egoera_oharra AS Oharra"
 + " FROM sarrera_lerroak sl"
 + " JOIN sarrerak s ON sl.sarrera_id = s.id_sarrera"
 + " JOIN produktuak p ON sl.produktua_id = p.id_produktua"
 + " JOIN biltegiak b ON p.biltegi_id = b.id_biltegia ORDER BY
s.data DESC";
 try (Connection con = DB_Konexioa.konektatu(); PreparedStatement pst =
con.prepareStatement(sql)) {
 DefaultTableModel ereduia =
TaulaModelatzaila.ereduaEraiki(pst.executeQuery());
 produktuTaula.setModel(ereduia);
 produktuOrdenatzailea = new TableRowSorter<>(ereduia);
 produktuTaula.setRowSorter(produktuOrdenatzailea);
 } catch (Exception e) {
 e.printStackTrace();
 }
}

/***
 * Taulak iragazi bilaketa testuaren arabera.
 */
private void filtratu() {
 String testua = bilatuTestua.getText();
 TableRowSorter<DefaultTableModel> unekoOrdenatzailea = null;
 int index = pestainaPanela.getSelectedIndex();
 if (index == 0)
 unekoOrdenatzailea = sarreraOrdenatzailea;
 else if (index == 1)
 unekoOrdenatzailea = biltegiOrdenatzailea;
 else if (index == 2)
 unekoOrdenatzailea = produktuOrdenatzailea;
 else if (index == 3)
 unekoOrdenatzailea = eskaeraOrdenatzailea;
 if (unekoOrdenatzailea != null) {
 if (testua.trim().length() == 0)
 unekoOrdenatzailea.setRowFilter(null);
 else
 unekoOrdenatzailea.setRowFilter(RowFilter.regexFilter("(?i)" +
testua));
 }
}

// --- SARRERA LOGIKA DESARROILUA ---

/***

```

```

* Hautatutako sarreraren lerroak ikusi eta haien egoera aldatzeko leihoa.
*/
private void ikusiSarreraLerroak() {
 int row = sarreraTaula.getSelectedRow();
 if (row == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu sarrera bat lehenbizi.");
 return;
 }
 int idSarrera = ((Number) sarreraTaula.getValueAt(row, 0)).intValue();

 JDialog dialog = new JDialog(this, "Sarrera Kodea: " + idSarrera + " - Lerroak", true);
 dialog.setSize(600, 400);
 dialog.setLocationRelativeTo(this);
 dialog.setLayout(new BorderLayout());

 String[] columns = { "ID", "Produktua", "Marka", "Kantitatea", "Egoera" };
 DefaultTableModel model = new DefaultTableModel(columns, 0);
 JTable table = new JTable(model);
 dialog.add(new JScrollPane(table), BorderLayout.CENTER);

 // Logic extracted to method for refreshing
 kargatuSarreraLerroak(idSarrera, model);

 JPanel btnPanel = new JPanel();
 JButton aldatuBtn = new JButton("Aldatu Lerroaren Egoera");
 aldatuBtn.addActionListener(e -> {
 int lRow = table.getSelectedRow();
 if (lRow == -1) {
 JOptionPane.showMessageDialog(dialog, "Aukeratu lerro bat.");
 return;
 }
 int idLerroa = ((Number) table.getValueAt(lRow, 0)).intValue();
 String unekoSt = (String) table.getValueAt(lRow, 4);

 String[] opts = { "Bidean", "Jasota", "Ezabatua" };
 String berria = (String) JOptionPane.showInputDialog(dialog,
 "Aukeratu egoera:", "Egoera Aldatu",
 JOptionPane.QUESTION_MESSAGE, null, opts, unekoSt);

 if (berria != null) {
 try {
 langilea.produktuSarreraLerroEgoeraAldatu(idLerroa, berria);
 kargatuSarreraLerroak(idSarrera, model);
 } catch (SQLException ex) {
 ex.printStackTrace();
 JOptionPane.showMessageDialog(dialog, "Errorea: " +
ex.getMessage());
 }
 }
 });
}

JButton itxiBtn = new JButton("Itxi");
itxiBtn.addActionListener(e -> dialog.dispose());

btnPanel.add(aldatuBtn);
btnPanel.add(itxiBtn);

```

```

 dialog.add(btnPanel, BorderLayout.SOUTH);

 dialog.setVisible(true);
 }

 /**
 * Sarrera baten lerroak taulan kargatu.
 *
 * @param idSarrera Sarreraren IDa.
 * @param model Taula eredu.
 */
 private void kargatuSarreraLerroak(int idSarrera, DefaultTableModel model) {
 model.setRowCount(0);
 try {
 java.util.List<Object[]> lerroak =
langilea.produktuSarreraLerroakIkusi(idSarrera);
 for (Object[] o : lerroak) {
 model.addRow(o);
 }
 } catch (SQLException e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea datuak kargatzean: " +
e.getMessage());
 }
 }

 /**
 * Sarrera baten egoera orokorra editatu.
 */
 private void editatuSarrera() {
 int row = sarreraTaula.getSelectedRow();
 if (row == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu sarrera bat lehenbizi.");
 return;
 }
 int idSarrera = ((Number) sarreraTaula.getValueAt(row, 0)).intValue();
 String unekoEgoera = (String) sarreraTaula.getValueAt(row, 3);

 String[] options = { "Bidean", "Jasota", "Ezabatua" };
 String berria = (String) JOptionPane.showInputDialog(this,
 "Aukeratu Egoera Berria:", "Egoera Aldatu",
 JOptionPane.QUESTION_MESSAGE, null, options, unekoEgoera);

 if (berria != null && !berria.equals(unekoEgoera)) {
 try {
 langilea.produktuSarreraEditatu(idSarrera, berria);
 sarreraDatuakKargatu();
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea egunereztean: " +
e.getMessage());
 }
 }
 }

 /**
 * Sarrera bat eta bere lerro guztiak ezabatu.
 */

```

```

private void ezabatuSarrera() {
 int row = sarreraTaula.getSelectedRow();
 if (row == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu sarrera bat lehenbizi.");
 return;
 }
 int idSarrera = ((Number) sarreraTaula.getValueAt(row, 0)).intValue();

 int confirm = JOptionPane.showConfirmDialog(this,
 "Ziur zaude sarrera hau eta bere lerro guztiak ezabatu nahi dituzula?",
 "Ezabatu", JOptionPane.YES_NO_OPTION);

 if (confirm == JOptionPane.YES_OPTION) {
 try {
 langilea.produktuSarreraEzabatu(idSarrera);
 sarreraDatuakKargatu();
 JOptionPane.showMessageDialog(this, "Sarrera ezabatua.");
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea ezabatzean: " +
e.getMessage());
 }
 }
}

// --- ESKAERA LOGIKA DESARROILUA ---

/**
 * Eskaeren datuak kargatu taulan.
 */
private void eskaeraDatuakKargatu() {
 try {
 String filter = (String) eskaeraEgoeraIragazkia.getSelectedItem();
 java.util.List<Object[]> data = langilea.produktuEskaerakIkusi(filter);

 String[] colNames = { "ID", "Bezeroa", "Data", "Guztira (€)", "Egoera" };
 DefaultTableModel eredua = new DefaultTableModel(colNames, 0);
 for (Object[] row : data) {
 eredua.addRow(row);
 }
 eskaeraTaula.setModel(eredua);
 eskaeraOrdenatzalea = new TableRowSorter<>(eredua);
 eskaeraTaula.setRowSorter(eskaeraOrdenatzalea);
 } catch (SQLException e) {
 e.printStackTrace();
 }
}

/**
 * Eskaera baten lerroak ikusi eta kudeatu.
 */
private void ikusiEskaeraLerroak() {
 int row = eskaeraTaula.getSelectedRow();
 if (row == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu eskaera bat lehenbizi.");
 return;
 }
}

```

```

 }

 int idEskaera = ((Number) eskaeraTaula.getValueAt(row, 0)).intValue();

 JDialog dialog = new JDialog(this, "Eskaera Kodea: " + idEskaera + " - Kudeaketa", true);
 dialog.setSize(700, 500);
 dialog.setLocationRelativeTo(this);
 dialog.setLayout(new BorderLayout());

 String[] columns = { "ID", "Produktua", "Kantitatea", "P.Unit.(€)", "Egoera" };
 DefaultTableModel model = new DefaultTableModel(columns, 0);
 JTable table = new JTable(model);
 dialog.add(new JScrollPane(table), BorderLayout.CENTER);

 // Karga Hasieran
 kargatuEskaeraLerroak(idEskaera, model);

 JPanel btnPanel = new JPanel();
 JButton aldatuBtn = new JButton("Aldatu Lerroaren Egoera");
 aldatuBtn.addActionListener(e -> {
 int lRow = table.getSelectedRow();
 if (lRow == -1)
 return;
 int idLerroa = ((Number) table.getValueAt(lRow, 0)).intValue();
 String unekoSt = (String) table.getValueAt(lRow, 4);

 String[] opts = { "Prestatzen", "Osatua/Bidalita", "Ezabatua" };
 String berria = (String) JOptionPane.showInputDialog(dialog,
 "Aukeratu egoera:", "Egoera Aldatu",
 JOptionPane.QUESTION_MESSAGE, null, opts, unekoSt);

 if (berria != null) {
 try {
 langilea.produktuEskaeraLerroEgoeraAldatu(idLerroa, berria,
 idEskaera);
 kargatuEskaeraLerroak(idEskaera, model); // Freskatu eskaeraDatuakKargatu(); // Freskatu atzeko taula nagusia ere
 } catch (SQLException ex) {
 ex.printStackTrace();
 }
 }
 });
 });

 JButton itxiBtn = new JButton("Itxi");
 itxiBtn.addActionListener(e -> dialog.dispose());

 btnPanel.add(aldatuBtn);
 btnPanel.add(itxiBtn);
 dialog.add(btnPanel, BorderLayout.SOUTH);

 dialog.setVisible(true);
}

/**
 * Eskaera baten lerroak kargatu.
 */

```

```

* @param idEskaera Eskaeraren IDa.
* @param model Taula eredu.
*/
private void kargatuEskaeraLerroak(int idEskaera, DefaultTableModel model) {
 model.setRowCount(0);
 try {
 java.util.List<Object[]> lerroak =
langilea.produktuEskaeraLerroakIkusi(idEskaera);
 for (Object[] o : lerroak) {
 model.addRow(o);
 }
 } catch (SQLException e) {
 e.printStackTrace();
 }
}

/**
* Produktu baten oharra editatu.
*/
private void editatuProduktuOharra() {
 int row = produktuTaula.getSelectedRow();
 if (row == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu produktu bat
lehenbizi.");
 return;
 }

 int idProduktua = ((Number) produktuTaula.getValueAt(row, 0)).intValue();
 // Assuming 'Oharra' is the last column (index 7 based on new SQL)
 String unekoOharra = "";
 Object val = produktuTaula.getValueAt(row, 7);
 if (val != null)
 unekoOharra = val.toString();

 String oharBerria = JOptionPane.showInputDialog(this, "Idatzi oharra:",
unekoOharra);
 if (oharBerria != null) {
 try {
 langilea.produktuEgoeraOharraJarri(idProduktua, oharBerria);
 produktuDatuakKargatu();
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea oharra gordetzean: " +
e.getMessage());
 }
 }
}

/**
* Biltegi berria sortu.
*/
private void sortuBiltegia() {
 JTextField izenaEremua = new JTextField();
 JTextField skuEremua = new JTextField();
 Object[] mezua = { "Biltegi Izena:", izenaEremua, "SKU Kodea:", skuEremua
};
 int aukera = JOptionPane.showConfirmDialog(this, mezua, "Biltegi Berria",
JOptionPane.OK_CANCEL_OPTION);
}

```

```

 if (aukera == JOptionPane.OK_OPTION) {
 try {
 langilea.biltegiaSortu(izenaEremua.getText(), skuEremua.getText());
 biltegiDatuakKargatu();
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
 }
 }
}

/**
 * Biltegi bat ezabatu.
 */
private void ezabatuBiltegia() {
 int row = biltegiTaula.getSelectedRow();
 if (row == -1)
 return;
 int modelRow = biltegiTaula.convertRowIndexToModel(row);
 Object val = biltegiTaula.getModel().getValueAt(modelRow, 0);
 int id = (val instanceof Number) ? ((Number) val).intValue() :
 Integer.parseInt(val.toString());
 try {
 if (JOptionPane.showConfirmDialog(this, "Ziur ezabatu nahi duzula?", "Ezabatu",
 JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION) {
 langilea.biltegiaEzabatu(id);
 biltegiDatuakKargatu();
 }
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, e.getMessage(), "Errorea",
 JOptionPane.ERROR_MESSAGE);
 }
}

/**
 * Biltegi baten datuak (izena, SKU) aldatu.
 */
private void aldatuBiltegia() {
 int row = biltegiTaula.getSelectedRow();
 if (row == -1)
 return;
 int modelRow = biltegiTaula.convertRowIndexToModel(row);
 int id = Integer.parseInt(biltegiTaula.getModel().getValueAt(modelRow,
0).toString());
 String izenaZaharra = (String) biltegiTaula.getModel().getValueAt(modelRow,
1);
 String skuZaharra = (String) biltegiTaula.getModel().getValueAt(modelRow,
2);
 JTextField izenaEremua = new JTextField(izenaZaharra);
 JTextField skuEremua = new JTextField(skuZaharra);
 if (JOptionPane.showConfirmDialog(this, new Object[] { "Izena:", izenaEremua, "SKU:", skuEremua }, "Aldatu",
 JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
 try {
 langilea.biltegiaEditatu(id, izenaEremua.getText(),
 skuEremua.getText());
 biltegiDatuakKargatu();
 }
 }
}

```

```

 } catch (Exception e) {
 e.printStackTrace();
 }
 }

 /**
 * Produkto bat biltegi batetik bestera mugitu.
 */
 private void aldatuProduktuarenBiltegia() {
 int row = produktuTaula.getSelectedRow();
 if (row == -1)
 return;
 int modelRow = produktuTaula.convertRowIndexToModel(row);
 int idProd = Integer.parseInt(produktuTaula.getModel().getValueAt(modelRow,
0).toString());
 JComboBox<BiltegiElementua> hautatzailea = new JComboBox<>();
 try (Connection con = DB_Konexioa.konektatu()) {
 ResultSet rs = con.createStatement().executeQuery("SELECT id_biltegia,
izenan FROM biltegiak");
 while (rs.next())
 hautatzailea.addItem(new BiltegiElementua(rs.getInt(1),
rs.getString(2)));
 } catch (Exception e) {
 e.printStackTrace();
 return;
 }
 if (JOptionPane.showConfirmDialog(this, hautatzailea, "Aukeratu Biltegi
Berria",
JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
 try {
 langilea.produktuarenBiltegiaAldatu(idProd, ((BiltegiElementua)
hautatzailea.getSelectedItem()).id);
 produktuDatuakKargatu();
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 }

 /**
 * Produkto baten lerroa "Jasota" markatu.
 */
 private void markatuProduktuaJasota() {
 produktuEgoeraAldatu("Jasota");
 }

 /**
 * Produkto baten lerroa "Bidean" markatu.
 */
 private void markatuProduktuaBidean() {
 produktuEgoeraAldatu("Bidean");
 }

 /**
 * Produkto sarrera baten lerro egoera aldatu (Helper).
 */

```

```

* @param egoeraBerria Egoera berria.
*/
private void produktuEgoeraAldatu(String egoeraBerria) {
 int row = produktuTaula.getSelectedRow();
 if (row == -1)
 return;
 int modelRow = produktuTaula.convertRowIndexToModel(row);
 Object valSarrera = produktuTaula.getModel().getValueAt(modelRow, 3);
 Object valLerroa = produktuTaula.getModel().getValueAt(modelRow, 6);
 int idSarrera = Integer.parseInt(valSarrera.toString());
 int idLerroa = Integer.parseInt(valLerroa.toString());
 try {
 langilea.produktuSarreraEgoeraAldatu(idLerroa, egoeraBerria,
idSarrera);
 produktuDatuakKargatu();
 sarreraDatuakKargatu();
 } catch (Exception e) {
 e.printStackTrace();
 }
}

/**
 * Saioa itxi.
*/
private void saioaItxi() {
 if (JOptionPane.showConfirmDialog(this, "Ziur zaude saioa itxi nahi
duzula?", "Logout",
 JOptionPane.YES_NO_OPTION) == 0) {
 dispose();
 new SaioaHastekoPanela().setVisible(true);
 }
}

// --- HELPER CLASSES ---
/**
 * Biltegi objektua ComboBox-erako.
*/
static class BiltegiElementua {
 int id;
 String izena;

 public BiltegiElementua(int id, String izena) {
 this.id = id;
 this.izena = izena;
 }

 public String toString() {
 return izena;
 }
}

/**
 * Hornitzaire objektua ComboBox-erako.
*/
static class HornitzaireElementua {
 int id;
 String izena;
}

```

```
public Hornitzairelementua(int id, String izena) {
 this.id = id;
 this.izena = izena;
}

public String toString() {
 return izena;
}
}

/**
 * Kategoria objektua ComboBox-erako.
 */
static class KategoriaElementua {
 int id;
 String izena;

 public KategoriaElementua(int id, String izena) {
 this.id = id;
 this.izena = izena;
 }

 public String toString() {
 return izena;
 }
}

/**
 * Atzealdeko panela irudiarekin.
 */
static class AtzealdekoPanela extends JPanel {
 private static final long serialVersionUID = 1L;
 private Image irudia;

 public AtzealdekoPanela() {
 try {
 java.net.URL imgURL =
MenuLogistika.class.getResource("/birtek_biltegia.png");
 if (imgURL != null)
 irudia = new ImageIcon(imgURL).getImage();
 else if (new File("src/birtek_biltegia.png").exists())
 irudia = new ImageIcon("src/birtek_biltegia.png").getImage();
 } catch (Exception e) {
 e.printStackTrace();
 }
 }

 @Override
 protected void paintComponent(Graphics g) {
 super.paintComponent(g);
 if (irudia != null) {
 int panelW = getWidth();
 int panelH = getHeight();
 int imgW = irudia.getWidth(this);
 int imgH = irudia.getHeight(this);
 if (imgW > 0 && imgH > 0) {
```

```
 double eskalatzea = Math.max((double) panelW / imgW, (double)
panelH / imgH);
 int newW = (int) (imgW * eskalatzea);
 int newH = (int) (imgH * eskalatzea);
 int x = (panelW - newW) / 2;
 int y = (panelH - newH) / 2;
 g.drawImage(irudia, x, y, newW, newH, this);
 } else {
 g.drawImage(irudia, 0, 0, panelW, panelH, this);
 }
 g.setColor(new Color(0, 0, 0, 100));
 g.fillRect(0, 0, panelW, panelH);
} else {
 g.setColor(new Color(245, 245, 245));
 g.fillRect(0, 0, getWidth(), getHeight());
}
}
```

# MenuSalmentak.java

```
package ui;

import db.DB_Konexioa;
import model.*;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.io.File;

/**
 * MenuSalmentak klasea.
 * Salmenta langilearen interfaze nagusia.
 * Bezeroak, eskaerak eta produktuak kudeatzeko aukerak eskaintzen ditu.
 */
public class MenuSalmentak extends JFrame {

 private static final long serialVersionUID = 1L;
 private JTable bezeroTaula, eskaeraTaula, eskaeraLerroTaula, productoTaula;
 private JTextField bilatuTestua;
 private JComboBox<String> categoriaFiltroa, motaFiltroa;
 private TableRowSorter<DefaultTableModel> bezeroOrdenatzailea,
eskaeraOrdenatzailea, productoOrdenatzailea,
 unekoOrdenatzailea;

 // Fitxaketa
```

```
private JLabel fitxaketaInfoEtiketa;

// Erabiltzailearen datuak
// Erabiltzailea (OOP)
private SalmentaLangilea langilea;

/***
 * Eraikitzailea eguneratua.
 */
/***
 * MenuSalmentak eraikitzailea.
 *
 * @param oinarrizkoLangilea Saioa hasi duen langilea.
 */
public MenuSalmentak(Langilea oinarrizkoLangilea) {
 this.langilea = new SalmentaLangilea(oinarrizkoLangilea);

 setTitle("Birtek - SALMENTAK");
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 setBounds(100, 100, 1050, 650);

 JPanel goikoPanela = new JPanel(new BorderLayout());
 getContentPane().add(goikoPanela, BorderLayout.NORTH);

 JPanel bilatzailePanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
 bilatzailePanela.add(new JLabel("Bilatu: "));
 bilatuTestua = new JTextField(20);
 bilatuTestua.addKeyListener(new KeyAdapter() {
 public void keyReleased(KeyEvent e) {
 filtratu();
 }
 });
 bilatzailePanela.add(bilatuTestua);

 JButton eguneratuBotoia = new JButton("Eguneratu");
 eguneratuBotoia.setBackground(new Color(173, 216, 230)); // Light Blue
 eguneratuBotoia.addActionListener(e -> {
 datuakKargatu();
 kargatuKategoriak();
 });
 bilatzailePanela.add(eguneratuBotoia);

 goikoPanela.add(bilatzailePanela, BorderLayout.WEST);

 // ESKUINALDEA: Erabiltzailea + Fitxaketa + Saioa Itxi
 JPanel eskuinekoPanela = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10,
 0));

 JLabel erabiltzaileEtiketa = new JLabel(langilea.getIzena() + " " +
langilea.getAbizena());
 erabiltzaileEtiketa.setFont(new Font("SansSerif", Font.BOLD, 12));
 erabiltzaileEtiketa.setForeground(new Color(0, 102, 102));

 // Fitxaketa Panela
 JPanel fitxaketaPanela = new JPanel();
 fitxaketaPanela.setLayout(new BoxLayout(fitxaketaPanela,
 BoxLayout.Y_AXIS));
```

```

JPanel botoiPanela = new JPanel(new FlowLayout(FlowLayout.CENTER, 2, 0));

 JButton sarreraBotoia = new JButton("Sarrera");
 sarreraBotoia.setBackground(new Color(34, 139, 34));
 sarreraBotoia.setForeground(Color.BLACK);
 sarreraBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
 sarreraBotoia.addActionListener(e -> fitxatu("Sarrera"));

 JButton irteeraBotoia = new JButton("Irteera");
 irteeraBotoia.setBackground(new Color(255, 140, 0));
 irteeraBotoia.setForeground(Color.BLACK);
 irteeraBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
 irteeraBotoia.addActionListener(e -> fitxatu("Irteera"));

 JButton historialBotoia = new JButton("Historiala");
 historialBotoia.setBackground(new Color(100, 149, 237));
 historialBotoia.setForeground(Color.BLACK);
 historialBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
 historialBotoia.addActionListener(e -> ikusiFitxaketaHistoriala());

 JButton nireDatuakBotoia = new JButton("Nire Datuak");
 nireDatuakBotoia.setBackground(new Color(100, 149, 237));
 nireDatuakBotoia.setForeground(Color.BLACK);
 nireDatuakBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
 nireDatuakBotoia.addActionListener(e -> irekiNireDatuakEditatu());

 botoiPanela.add(sarreraBotoia);
 botoiPanela.add(irteeraBotoia);
 botoiPanela.add(historialBotoia);
 botoiPanela.add(nireDatuakBotoia);

 fitxaketaInfoEtiketa = new JLabel("Kargatzen...");
 fitxaketaInfoEtiketa.setFont(new Font("SansSerif", Font.PLAIN, 9));
 fitxaketaInfoEtiketa.setAlignmentX(Component.CENTER_ALIGNMENT);

 fitxaketaPanela.add(botoiPanela);
 fitxaketaPanela.add(fitxaketaInfoEtiketa);

 JButton saioaItxiBotoia = new JButton("Saioa Itxi");
 saioaItxiBotoia.setBackground(Color.RED);
 saioaItxiBotoia.setForeground(Color.WHITE);
 saioaItxiBotoia.addActionListener(e -> saioaItxi());

 eskuinekoPanela.add(erabiltzaileEtiketa);
 eskuinekoPanela.add(fitxaketaPanela);
 eskuinekoPanela.add(saioaItxiBotoia);

 goikoPanela.add(eskuinekoPanela, BorderLayout.EAST);

 pestainaPanelaRef = new JTabbedPane(JTabbedPane.TOP);
 getContentPane().add(pestainaPanelaRef, BorderLayout.CENTER);

 JTabbedPane pestainaPanela = pestainaPanelaRef;

 // BEZEROAK PANELA
 JPanel bezeroPanela = new JPanel(new BorderLayout());
 bezeroTaula = new JTable();

```

```

bezeroPanela.add(new JScrollPane(bezeroTaula), BorderLayout.CENTER);

JPanel bezeroBotoiPanela = new JPanel(new FlowLayout(FlowLayout.CENTER));
JButton bezeroGehituBtn = new JButton("Gehitu");
JButton bezeroEditatuBtn = new JButton("Editatu");
JButton bezeroEzabatuBtn = new JButton("Ezabatu");
JButton bezeroIkusiBtn = new JButton("Ikusi");
JButton bezeroHistorialaBtn = new JButton("Eskaera Historiala Ikusi");

bezeroGehituBtn.addActionListener(e -> bezeroaGehitu());
bezeroEditatuBtn.addActionListener(e -> bezeroaEditatu());
bezeroEzabatuBtn.addActionListener(e -> bezeroaEzabatu());
bezeroIkusiBtn.addActionListener(e -> bezeroaIkusi());
bezeroHistorialaBtn.addActionListener(e -> bezeroHistorialaIkusi());

bezeroBotoiPanela.add(bezeroGehituBtn);
bezeroBotoiPanela.add(bezeroEditatuBtn);
bezeroBotoiPanela.add(bezeroEzabatuBtn);
bezeroBotoiPanela.add(bezeroIkusiBtn);
bezeroBotoiPanela.add(bezeroHistorialaBtn);

bezeroPanela.add(bezeroBotoiPanela, BorderLayout.SOUTH);

pestainaPanela.addTab("Bezeroak", bezeroPanela);

// ESKAERAK PANELA (SplitPane)
eskaeraTaula = new JTable();
eskaeraTaula.getSelectionModel().addListSelectionListener(e -> {
 if (!e.getValueIsAdjusting()) {
 fokatutakoEskaeraKargatu();
 }
});

eskaeraLerroTaula = new JTable();

JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT,
 new JScrollPane(eskaeraTaula),
 new JScrollPane(eskaeraLerroTaula));
splitPane.setDividerLocation(300);
splitPane.setResizeWeight(0.5);

JPanel eskaeraPanela = new JPanel(new BorderLayout());
eskaeraPanela.add(splitPane, BorderLayout.CENTER);
pestainaPanela.addTab("Eskaerak", eskaeraPanela);

// Eskaera botoiak
JPanel eskaeraBotoiPanela = new JPanel(new FlowLayout(FlowLayout.CENTER));
JButton eskaeraGehituBotoia = new JButton("Gehitu");
JButton eskaeraEditatuBotoia = new JButton("Editatu");
JButton eskaeraEzabatuBotoia = new JButton("Ezabatu");
JButton eskaeraFakturaBotoia = new JButton("Faktura");
JButton eskaeraFakturaEzabatuBotoia = new JButton("Faktura Ezabatu");
JButton eskaeraFakturaGuztiakBotoia = new JButton("Faktura Guztiak sortu");

eskaeraGehituBotoia.addActionListener(e -> eskaeraGehitu());
eskaeraEditatuBotoia.addActionListener(e -> eskaeraEditatu());
eskaeraEzabatuBotoia.addActionListener(e -> eskaeraEzabatu());

```

```

eskaeraFakturaBotoia.addActionListener(e -> fakturaSortu());
eskaeraFakturaEzabatuBotoia.addActionListener(e -> fakturaEzabatu());
eskaeraFakturaGuztiakBotoia.addActionListener(e -> fakturaGuztiakSortu());

eskaeraBotoiPanela.add(eskaeraGehituBotoia);
eskaeraBotoiPanela.add(eskaeraEditatuBotoia);
eskaeraBotoiPanela.add(eskaeraEzabatuBotoia);
eskaeraBotoiPanela.add(eskaeraFakturaBotoia);
eskaeraBotoiPanela.add(eskaeraFakturaEzabatuBotoia);
eskaeraBotoiPanela.add(eskaeraFakturaGuztiakBotoia);

eskaeraPanela.add(eskaeraBotoiPanela, BorderLayout.SOUTH);

// PRODUKTUAK PANELA
 JPanel produktuPanela = new JPanel(new BorderLayout());
 produktuTaula = new JTable();
 produktuPanela.add(new JScrollPane(produktuTaula), BorderLayout.CENTER);

 JPanel produktuKontrolPanela = new JPanel(new BorderLayout());
 JPanel produktuFiltroPanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
 kategoriaFiltroa = new JComboBox<>();
 kategoriaFiltroa.addItem("Kategoria Denak");
 motaFiltroa = new JComboBox<>();
 motaFiltroa.addItem("Mota Denak");
 // Motak gehitu (Enum-etik datozen DBan)
 String[] motak = { "Eramangarria", "Mahai-gainekoa", "Mugikorra",
 "Tableta", "Zerbitzaria", "Pantaila",
 "Softwarea" };
 for (String m : motak)
 motaFiltroa.addItem(m);

 kategoriaFiltroa.addActionListener(e -> filtratu());
 motaFiltroa.addActionListener(e -> filtratu());

 produktuFiltroPanela.add(new JLabel("Kategoria:"));
 produktuFiltroPanela.add(kategoriaFiltroa);
 produktuFiltroPanela.add(new JLabel("Mota:"));
 produktuFiltroPanela.add(motaFiltroa);
 produktuKontrolPanela.add(produktuFiltroPanela, BorderLayout.NORTH);

 JPanel produktuBotoiPanela = new JPanel(new FlowLayout(FlowLayout.CENTER));
 JButton prezioaBtn = new JButton("Prezioa Eguneratu");
 JButton salgaiBtn = new JButton("Salgai/Ez");
 JButton eskaintzaBtn = new JButton("Eskaintza Gehitu");

 prezioaBtn.addActionListener(e -> produktuPrezioaAldatu());
 salgaiBtn.addActionListener(e -> produktuSalgaiToggle());
 eskaintzaBtn.addActionListener(e -> produktuEskaintzaJarri());

 produktuBotoiPanela.add(prezioaBtn);
 produktuBotoiPanela.add(salgaiBtn);
 produktuBotoiPanela.add(eskaintzaBtn);
 produktuKontrolPanela.add(produktuBotoiPanela, BorderLayout.SOUTH);

 produktuPanela.add(produktuKontrolPanela, BorderLayout.SOUTH);
 pestainaPanela.addTab("Produktuak", produktuPanela);

```

```

pestainaPanela.addChangeListener(e -> {
 bilatuTestua.setText("");
 int idx = pestainaPanela.getSelectedIndex();
 if (idx == 0)
 unekoOrdenatzailea = bezeroOrdenatzailea;
 else if (idx == 1)
 unekoOrdenatzailea = eskaeraOrdenatzailea;
 else if (idx == 2)
 unekoOrdenatzailea = produktuOrdenatzailea;
});

if (!java.beans.Beans.isDesignTime()) {
 datuakKargatu();
 kargatuKategoriak();
 eguneraFitxaketaEgoera();
}
}

// --- FITXAKETA METODO BERRIAK ---
/**
 * Fitxaketa bat egin (Sarrera/Irteera).
 *
 * @param mota Fitxaketa mota.
 */
private void fitxatu(String mota) {
 try {
 langilea.fitxatu(mota);
 eguneraFitxaketaEgoera();
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, e.getMessage(), "Errorea",
JOptionPane.WARNING_MESSAGE);
 }
}

/**
 * Fitxaketa egoera eguneraFitxaketaEgoera() etiketan.
 */
private void eguneraFitxaketaEgoera() {
 fitxaketaInfoEtiketa.setText(langilea.getFitxaketaEgoera());
 if (fitxaketaInfoEtiketa.getText().contains("BARRUAN")) {
 fitxaketaInfoEtiketa.setForeground(new Color(0, 100, 0));
 } else {
 fitxaketaInfoEtiketa.setForeground(new Color(200, 0, 0));
 }
}

/**
 * Fitxaketa historialaren leioha ireki.
 */
private void ikusiFitxaketaHistoriala() {
 JDialog elkarrizketa = new JDialog(this, "Fitxaketa Historiala", true);
 elkarrizketa.setSize(500, 400);
 elkarrizketa.setLocationRelativeTo(this);
 elkarrizketa.setLayout(new BorderLayout());
 String[] zutabeak = { "Mota", "Data", "Ordua" };
 DefaultTableModel eredu = new DefaultTableModel(zutabeak, 0);
 JTable taula = new JTable(eredu);
}

```

```

elkarritzeta.add(new JScrollPane(taula), BorderLayout.CENTER);

java.util.List<Fitxaketa> zerrenda = langilea.nireFitxaketakIkusi();
for (Fitxaketa f : zerrenda) {
 eredu.addRow(new Object[] { f.getMota(), f.getData(), f.getOrdua() });
}

elkarritzeta.setVisible(true);
}

/**
 * Norberaren datuak editatzeko leihoa ireki.
 */
private void irekiNireDatuakEditatu() {
 NireDatuakDialog dialog = new NireDatuakDialog(this, langilea);
 dialog.setVisible(true);
}

/**
 * Hautatutako eskaeraren lerroak beheko taulan kargatu.
 */
private void fokatutakoEskaeraKargatu() {
 int aukeratutakoLerroa = eskaeraTaula.getSelectedRow();
 if (aukeratutakoLerroa != -1) {
 aukeratutakoLerroa =
eskaeraTaula.convertRowIndexToModel(aukeratutakoLerroa);
 Object idObj = eskaeraTaula.getModel().getValueAt(aukeratutakoLerroa,
0);
 int idEskaera = (idObj instanceof Number) ? ((Number) idObj).intValue()
 : Integer.parseInt(idObj.toString());

 try {
 java.util.List<EskaeraLerroa> lerroak =
langilea.eskaeraLerroakIkusi(idEskaera);
 String[] zutabeak = { "Produktua", "Kantitatea", "P.Unit.",
"Guztira", "Egoera" };
 DefaultTableModel model = new DefaultTableModel(zutabeak, 0);

 for (EskaeraLerroa l : lerroak) {
 Produktua p = langilea.produktuaIkusi(l.getProduktuaId());
 String pIzena = (p != null) ? p.getIzena() : "ID: " +
l.getProduktuaId();
 model.addRow(new Object[] {
 pIzena,
 l.getKantitatea(),
 l.getUnitatePrezioa(),
 l.getUnitatePrezioa().multiply(new
java.math.BigDecimal(l.getKantitatea())),
 l.getEskaeraLerroEgoera()
 });
 }
 eskaeraLerroTaula.setModel(model);
 } catch (Exception e) {
 e.printStackTrace();
 }
 } else {
 eskaeraLerroTaula.setModel(new DefaultTableModel());
 }
}

```

```

 }

 /**
 * Taulak iragazi bilaketa testuaren eta filtroen arabera.
 */
 private void filtratu() {
 int pestainaIdx = pestainaPanelaRef.getSelectedIndex();

 if (pestainaIdx == 2) { // Produktuak
 if (produktuOrdenatzailea != null) {
 java.util.List<RowFilter<Object, Object>> filters = new
java.util.ArrayList<>();

 // Testu iragazkia
 String t = bilatuTestua.getText();
 if (!t.isEmpty()) {
 filters.add(RowFilter.regexFilter("(?i)" + t));
 }

 // Kategoria iragazkia
 String kat = (String) kategoriaFiltroa.getSelectedItem();
 if (kat != null && !"Kategoria Denak".equals(kat)) {
 filters.add(RowFilter.regexFilter("^" + kat + "$", 1)); // 1 =
Kategoria zutabea
 }

 // Mota iragazkia
 String mota = (String) motaFiltroa.getSelectedItem();
 if (mota != null && !"Mota Denak".equals(mota)) {
 filters.add(RowFilter.regexFilter("^" + mota + "$", 4)); // 4 =
Mota zutabea
 }

 produktuOrdenatzailea.setRowFilter(RowFilter.andFilter(filters));
 }
 } else { // Bezeroak edo Eskaerak
 String t = bilatuTestua.getText();
 if (unekoOrdenatzailea != null) {
 if (t.isEmpty())
 unekoOrdenatzailea.setRowFilter(null);
 else
 unekoOrdenatzailea.setRowFilter(RowFilter.regexFilter("(?i)" +
t));
 }
 }
 }

 /**
 * Saioa itxi.
 */
 private void saioaItxi() {
 if (JOptionPane.showConfirmDialog(this, "Irten?", "Saioa Itxi",
JOptionPane.YES_NO_OPTION) == 0) {
 dispose();
 new SaioaHastekoPanela().setVisible(true);
 }
 }
}

```

```

}

/**
 * Datu guztiak (Bezeroak, Eskaerak, Produktuak) datu-basetik kargatu.
 */
private void datuakKargatu() {
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 DefaultTableModel m1 = TaulaModelatzalea.ereduEraiki(konexioa
 .prepareStatement("SELECT id_bezeroa, izena_edo_soziala, emaila
FROM bezeroak").executeQuery());
 bezeroTaula.setModel(m1);
 bezeroOrdenatzalea = new TableRowSorter<>(m1);
 bezeroTaula.setRowSorter(bezeroOrdenatzalea);

 DefaultTableModel m2 = TaulaModelatzalea
 .ereduaEraiki(konexioa.prepareStatement("SELECT * FROM
eskaerak").executeQuery());
 eskaeraTaula.setModel(m2);
 eskaeraOrdenatzalea = new TableRowSorter<>(m2);
 eskaeraTaula.setRowSorter(eskaeraOrdenatzalea);

 produktuDatuakKargatu();

 if (unekoOrdenatzalea == null)
 unekoOrdenatzalea = bezeroOrdenatzalea;
 } catch (Exception e) {
 e.printStackTrace();
 }
}

/***
 * Eskaera berria sortzeko prozesua (Dialogo bidez).
 */
private void eskaeraGehitu() {
 EskaeraDialog dialog = new EskaeraDialog(this, "Gehitu Eskaera", null,
"Prestatzen");
 dialog.setVisible(true);
 if (dialog.isOnartua()) {
 try {
 // Eskaera objektua prestatu
 Eskaera e = new Eskaera(0, dialog.getBezeroaId(),
langilea.getIdLangilea(), null, null,
 dialog.getPrezioTotala(), null, null, dialog.getEgoera());

 // Lerroak prestatu
 java.util.List<EskaeraLerroa> lerroak = new java.util.ArrayList<>
();
 for (Object[] obj : dialog.getLerroak()) {
 lerroak.add(new EskaeraLerroa(0, 0, (int) obj[0], (int) obj[1],
(java.math.BigDecimal) obj[2],
 dialog.getEgoera()));
 }

 // Modeloko metodoari deitu
 int idEskaera = langilea.eskaeraOsoaSortu(e, lerroak);

 // Faktura automatikoa sortu
 }
 }
}

```

```

 if ("Osatua/Bidalita".equalsIgnoreCase(dialog.getEgoera()) &&
idEskaera != -1) {
 try {
 langilea.fakturaSortu(idEskaera);
 } catch (Exception ex) {
 ex.printStackTrace();
 }
 }
 datuakKargatu();
 } catch (SQLException ex) {
 ex.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea eskaera sortzean: " +
ex.getMessage());
 }
}

/**
 * Hautatutako eskaera editatu.
 */
private void eskaeraEditatu() {
 int aukeratutakoLerroa = eskaeraTaula.getSelectedRow();
 if (aukeratutakoLerroa == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu eskaera bat
editatzeko.");
 return;
 }

 // Datuak lortu taulatik
 aukeratutakoLerroa =
eskaeraTaula.convertRowIndexToModel(aukeratutakoLerroa);
 DefaultTableModel model = (DefaultTableModel) eskaeraTaula.getModel();

 Object idObj = model.getValueAt(aukeratutakoLerroa, 0);
 int idEskaera = (idObj instanceof Number) ? ((Number) idObj).intValue()
 : Integer.parseInt(idObj.toString());

 Object bezeroObj = model.getValueAt(aukeratutakoLerroa, 1);
 int bezeroaId = (bezeroObj instanceof Number) ? ((Number)
bezeroObj).intValue()
 : Integer.parseInt(bezeroObj.toString());

 String egoera = (String) model.getValueAt(aukeratutakoLerroa, 8);

 EskaeraDialog dialog = new EskaeraDialog(this, "Editatu Eskaera",
bezeroaId, egoera);

 // Lerro zaharrak kargatu
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(
 "SELECT el.productua_id, p.izena, el.unitate_precioa,
el.kantitatea " +
 "FROM eskaera_lerroak el JOIN produktuak p ON
el.productua_id = p.id_productua " +
 "WHERE el.eskaera_id = ?")) {
 pst.setInt(1, idEskaera);
 ResultSet rs = pst.executeQuery();

```

```

 while (rs.next()) {
 dialog.addZuzeneanLerroa(
 rs.getInt("produktua_id"),
 rs.getString("izena"),
 rs.getBigDecimal("unitate_presioa"),
 rs.getInt("kantitatea"),
 0.0);
 }
 } catch (SQLException e) {
 e.printStackTrace();
 }

 dialog.setVisible(true);

 if (dialog.isOnartua()) {
 try {
 // Eskaera objektua prestatu
 Eskaera e = new Eskaera(idEskaera, dialog.getBezeroaId(),
langilea.getIdLangilea(), null, null,
 dialog.getPrezioTotala(), null, null, dialog.getEgoera()));

 // Lerroak prestatu
 java.util.List<EskaeraLerroa> lerroak = new java.util.ArrayList<>
();
 for (Object[] obj : dialog.getLerroak()) {
 lerroak.add(new EskaeraLerroa(0, idEskaera, (int) obj[0], (int)
obj[1],
 (java.math.BigDecimal) obj[2], dialog.getEgoera())));
 }

 // Modeloko metodoari deitu
 langilea.eskaeraOsoaEditatu(e, lerroak);

 // Faktura automatikoa sortu
 if ("Osatua/Bidalita".equalsIgnoreCase(dialog.getEgoera())) {
 try {
 langilea.fakturaSortu(idEskaera);
 } catch (Exception ex) {
 ex.printStackTrace();
 }
 }
 datuakKargatu();
 } catch (SQLException ex) {
 ex.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea eskaera editatzean: "
+ ex.getMessage());
 }
 }
}

/**
 * Hautatutako eskaera ezabatu.
 */
private void eskaeraEzabatu() {
 int aukeratutakoLerroa = eskaeraTaula.getSelectedRow();
 if (aukeratutakoLerroa == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu eskaera bat

```

```

ezabatzeko.");
 return;
}

aukeratutakoLerroa =
eskaeraTaula.convertRowIndexToModel(aukeratutakoLerroa);
Object idObj = eskaeraTaula.getModel().getValueAt(aukeratutakoLerroa, 0);
int idEskaera = (idObj instanceof Number) ? ((Number) idObj).intValue() :
Integer.parseInt(idObj.toString());

if (JOptionPane.showConfirmDialog(this, "Ziur zaude eskaera hau ezabatu
nahi duzula?", "Ezabatu",
JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION) {
try {
 langilea.eskaeraEzabatu(idEskaera);
 datuakKargatu();
} catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea eskaera ezabatzean: "
+ e.getMessage());
}
}

/**
 * Hautatutako eskaeraren faktura sortu (PDF).
 */
private void fakturaSortu() {
 int aukeratutakoLerroa = eskaeraTaula.getSelectedRow();
 if (aukeratutakoLerroa == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu eskaera bat faktura
sortzeko.");
 return;
 }

 aukeratutakoLerroa =
eskaeraTaula.convertRowIndexToModel(aukeratutakoLerroa);
 DefaultTableModel model = (DefaultTableModel) eskaeraTaula.getModel();

 Object idObj = model.getValueAt(aukeratutakoLerroa, 0);
 int idEskaera = (idObj instanceof Number) ? ((Number) idObj).intValue() :
Integer.parseInt(idObj.toString());
 String egoera = (String) model.getValueAt(aukeratutakoLerroa, 8);

 if (!"Osatua/Bidalita".equalsIgnoreCase(egoera)) {
 JOptionPane.showMessageDialog(this,
 "Faktura bakarrik 'Osatua/Bidalita' egoeran dauden
eskaeretarako sor daiteke.");
 return;
 }

 try {
 File fakturaFitxategia = langilea.fakturaSortu(idEskaera);
 if (fakturaFitxategia != null) {
 JOptionPane.showMessageDialog(this,
 "Faktura PDF sortu eta gordeta: " +
fakturaFitxategia.getAbsolutePath());
 }
}

```

```

 java.awt.Desktop.getDesktop().open(fakturaFitxategia);
 }
} catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea faktura sortzean: " +
e.getMessage());
}
}

/**
 * Osatua/Bidalita dauden eskaera guztien fakturak sortu.
*/
private void fakturaGuztiakSortu() {
 String sql = "SELECT id_eskaera FROM eskaerak WHERE eskaera_egoera =
'Osatua/Bidalita'";
 java.util.List<Integer> idZerrenda = new java.util.ArrayList<>();
 int kontagailua = 0;

 // 1. Fasea: ID-ak lortu (Konexioa ireki eta itxi egiten da bloke honetan)
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 PreparedStatement pst = konexioa.prepareStatement(sql);
 ResultSet rs = pst.executeQuery();

 while (rs.next()) {
 idZerrenda.add(rs.getInt("id_eskaera"));
 }
 } catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea eskaerak bilatzean: " +
e.getMessage());
 return;
 }

 // 2. Fasea: Fakturak sortu (Bakoitzak bere konexio kudeaketa dauka
barruan)
 for (int idEskaera : idZerrenda) {
 try {
 if (langilea.fakturaSortu(idEskaera) != null) {
 kontagailua++;
 }
 } catch (Exception e) {
 e.printStackTrace();
 }
 }

 JOptionPane.showMessageDialog(this, kontagailua + " faktura sortu dira
zuzen.");
}

/**
 * Eskaera baten faktura logikoa ezabatu (fitxategia mantendu daiteke).
*/
private void fakturaEzabatu() {
 int aukeratutakoLerroa = eskaeraTaula.getSelectedRow();
 if (aukeratutakoLerroa == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu eskaera bat faktura
ezabatzeko.");
 }
}
```

```

 return;
 }

 aukeratutakoLerroa =
eskaeraTaula.convertRowIndexToModel(aukeratutakolLerroa);
 DefaultTableModel model = (DefaultTableModel) eskaeraTaula.getModel();

 Object idObj = model.getValueAt(aukeratutakoLerroa, 0);
 int idEskaera = (idObj instanceof Number) ? ((Number) idObj).intValue()
 : Integer.parseInt(idObj.toString());

 if (JOptionPane.showConfirmDialog(this, "Ziur zaude eskaera honen faktura
ezabatu nahi duzula?",
 "Faktura Ezabatu", JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION) {
 try {
 langilea.fakturaEzabatu(idEskaera);
 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Faktura ondo ezabatu da.");
 } catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea faktura ezabatzean: "
+ e.getMessage());
 }
 }

 /**
 * Bezero baten xehetasunak ikusi.
 */
 private void bezeroaIkusi() {
 int r = bezeroTaula.getSelectedRow();
 if (r == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu bezero bat ikusteko.");
 return;
 }
 int rm = bezeroTaula.convertRowIndexToModel(r);
 Object idObj = bezeroTaula.getModel().getValueAt(rm, 0);
 int idBezeroa = Integer.parseInt(idObj.toString());

 try {
 Bezeroa b = langilea.bezeroaIkusi(idBezeroa);
 if (b != null) {
 BezeroaDialog dialog = new BezeroaDialog(this, "Bezeroa Ikusi", b,
langilea);
 dialog.setViewMode(true);
 dialog.setVisible(true);
 }
 } catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
 }
 }

 /**
 * Bezero berria sortu.
 */

```

```

private void bezeroaGehitu() {
 BezeroaDialog dialog = new BezeroaDialog(this, "Bezeroa Berria", null,
langilea);
 dialog.setVisible(true);
 if (dialog.isOnartua()) {
 try {
 langilea.bezeroBerriaSortu(dialog.getBezeroa());
 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Bezeroa ondo sortu da.");
 } catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea bezeroa sortzean: " +
e.getMessage());
 }
 }
}

/**
 * Bezero baten datuak editatu.
 */
private void bezeroaEditatu() {
 int r = bezeroTaula.getSelectedRow();
 if (r == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu bezero bat editatzeko.");
 return;
 }
 int rm = bezeroTaula.convertRowIndexToModel(r);
 Object idObj = bezeroTaula.getModel().getValueAt(rm, 0);
 int idBezeroa = Integer.parseInt(idObj.toString());

 try {
 Bezeroa b = langilea.bezeroaIkusi(idBezeroa);
 if (b != null) {
 Bezeroa bBerria = dialog.getBezeroa();
 // IDa mantendu
 bBerria.setIdBezeroa(idBezeroa);
 langilea.bezeroaEditatu(bBerria);
 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Bezeroa eguneratu da.");
 }
 }
} catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
}
}

/**
 * Bezero bat ezabatu.
 */
private void bezeroaEzabatu() {
 int r = bezeroTaula.getSelectedRow();
 if (r == -1) {

```

```

 JOptionPane.showMessageDialog(this, "Aukeratu bezero bat ezabatzeko.");
 return;
 }
 int rm = bezeroTaula.convertRowIndexToModel(r);
 Object idObj = bezeroTaula.getModel().getValueAt(rm, 0);
 int idBezeroa = Integer.parseInt(idObj.toString());

 if (JOptionPane.showConfirmDialog(this, "Ziur zaude bezero hau ezabatu nahi
duzula?", "Ezabatu",
 JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION) {
 try {
 langilea.bezeroaKendu(idBezeroa);
 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Bezeroa ezabatu da.");
 } catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage(),
"Ezin da ezabatu",
 JOptionPane.ERROR_MESSAGE);
 }
 }
}

// Klaseko aldagaien JTabbedPane gordetzeko
private JTabbedPane pestainaPanelaRef;

/**
 * Bezero baten eskaera historia ikusteko iragazkia aktibatu eskaeren fitxan.
 */
private void bezeroHistorialaIkusi() {
 int r = bezeroTaula.getSelectedRow();
 if (r == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu bezero bat historiala
ikusteko.");
 return;
 }
 int rm = bezeroTaula.convertRowIndexToModel(r);
 Object idObj = bezeroTaula.getModel().getValueAt(rm, 0);
 // Bezeroaren izena ere lortu dezakegu filtroan erakusteko, baina IDa
nahikoa da
 String idStr = idObj.toString();

 if (pestainaPanelaRef != null) {
 pestainaPanelaRef.setSelectedIndex(1); // Eskaerak tab

 if (eskaeraOrdenatzailea != null) {
 // Zehatz mehatz ID hori dutenak (regex hasiera eta amaiera ^ $)
 // eskaerak taulako 2. zutabea (index 1) da bezeroa_id
 eskaeraOrdenatzailea.setRowFilter(RowFilter.regexFilter("^" + idStr
+ "$", 1));
 bilatuTestua.setText("Bezero ID: " + idStr); // Erabiltzaileari
abisatu
 }
 }
}

private void kargatuKategoriak() {

```

```

 if (kategoriaFiltroa != null) {
 kategoriaFiltroa.removeAllItems();
 kategoriaFiltroa.addItem("Kategoria Denak");
 }
 try (Connection konexioa = DB_Konexioa.konektatu();
 Statement stmt = konexioa.createStatement();
 ResultSet rs = stmt.executeQuery("SELECT izena FROM
produktu_kategoriak")) {
 while (rs.next()) {
 kategoriaFiltroa.addItem(rs.getString("izena"));
 }
 } catch (SQLException e) {
 e.printStackTrace();
 }
}

private void produktuDatuakKargatu() {
 String sql = "SELECT p.id_produktua, pk.izena as kategoria, p.izena,
p.marka, p.mota, p.salmenta_prezioa, p.stock, p.salgai, p.eskaintza "
 + "FROM produktuak p JOIN produktu_kategoriak pk ON p.kategoria_id
= pk.id_kategoria";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement pst = konexioa.prepareStatement(sql)) {
 DefaultTableModel model =
TaulaModelatzalea.ereduEraiki(pst.executeQuery());
 produktuTaula.setModel(model);
 produktuOrdenatzalea = new TableRowSorter<>(model);
 produktuTaula.setRowSorter(produktuOrdenatzalea);
 } catch (Exception e) {
 e.printStackTrace();
 }
}

private void produktuPrezioaAldatu() {
 int row = produktuTaula.getSelectedRow();
 if (row == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu produktu bat.");
 return;
 }
 row = produktuTaula.convertRowIndexToModel(row);
 Object idObj = produktuTaula.getModel().getValueAt(row, 0);
 int idProd = (idObj instanceof Number) ? ((Number) idObj).intValue() :
Integer.parseInt(idObj.toString());
 String izena = (String) produktuTaula.getModel().getValueAt(row, 2);

 String berria = JOptionPane.showInputDialog(this, izena + " - Sartu prezio
berria:");
 produktuTaula.getModel().setValueAt(berria, row, 5);
 if (berria != null && !berria.isEmpty()) {
 try {
 langilea.produktuariPrezioaAldatu(idProd, new
java.math.BigDecimal(berria));
 produktuDatuakKargatu();
 JOptionPane.showMessageDialog(this, "Prezioa eguneratuta.");
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
 }
 }
}

```

```

 }

 }

 private void produktuSalgaiToggle() {
 int row = produktuTaula.getSelectedRow();
 if (row == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu produktu bat.");
 return;
 }
 row = produktuTaula.convertRowIndexToModel(row);
 Object idObj = produktuTaula.getModel().getValueAt(row, 0);
 int idProd = (idObj instanceof Number) ? ((Number) idObj).intValue() :
 Integer.parseInt(idObj.toString());

 Object salgaiObj = produktuTaula.getModel().getValueAt(row, 7);
 boolean salgai = false;
 if (salgaiObj instanceof Boolean) {
 salgai = (Boolean) salgaiObj;
 } else if (salgaiObj instanceof Number) {
 salgai = ((Number) salgaiObj).intValue() != 0;
 } else if (salgaiObj != null) {
 salgai = "true".equalsIgnoreCase(salgaiObj.toString()) ||
 "1".equals(salgaiObj.toString());
 }

 try {
 // Salgai jarri/kendu logic
 String sql = "UPDATE produktuak SET salgai = ? WHERE id_produktua = ?";
 try (Connection kon = DB_Konexioa.konektatu();
 PreparedStatement pst = kon.prepareStatement(sql)) {
 pst.setBoolean(1, !salgai);
 pst.setInt(2, idProd);
 pst.executeUpdate();
 }
 produktuDatuakKargatu();
 JOptionPane.showMessageDialog(this, "Egoera aldatuta.");
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
 }
 }

 private void produktuEskaintzaJarri() {
 int row = produktuTaula.getSelectedRow();
 if (row == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu produktu bat.");
 return;
 }
 row = produktuTaula.convertRowIndexToModel(row);
 Object idObj = produktuTaula.getModel().getValueAt(row, 0);
 int idProd = (idObj instanceof Number) ? ((Number) idObj).intValue() :
 Integer.parseInt(idObj.toString());

 String eskaintza = JOptionPane.showInputDialog(this, "Sartu eskaintza
 (ehuneko, adib. 10):", "0");
 if (eskaintza != null && !eskaintza.isEmpty()) {
 try {
 langilea.produktuariEskaintzaAldatzeko(idProd, new

```

```

 java.math.BigDecimal(eskaintza));
 produktuDatuakKargatu();
 JOptionPane.showMessageDialog(this, "Eskaintza eguneratuta.");
 } catch (Exception e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
 }
}
}

```

## MenuTeknikoa.java

```

package ui;

import db.DB_Konexioa;
import model.*;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.ArrayList;

/**
 * MenuTeknikoa klasea.
 * Teknikari langilearen interfaze nagusia (SAT).
 * Konponketak, produktuak eta akatsak kudeatzeko aukerak.
 */
public class MenuTeknikoa extends JFrame {

 private static final long serialVersionUID = 1L;
 private JTable konponketaTaula, productoTaula, akatsTaula;
 private JTextField bilatuTestua;
 private TableRowSorter<DefaultTableModel> konponketaOrdenatzailea,
productoOrdenatzailea, akatsOrdenatzailea,
unekoOrdenatzailea;

 // Fitxaketa
 private JLabel fitxaketaInfoEtiketa;

 // Erabiltzaile datuak
 private TeknikariLangilea langilea;

 /**
 * Eraikitzailea eguneratua.
 */
 /**
 * MenuTeknikoa eraikitzailea.
 */

```

```

*
* @param oinarrizkoLangilea Saioa hasi duen langilea.
*/
public MenuTeknikoa(Langilea oinarrizkoLangilea) {
 this.langilea = new TeknikariLangilea(oinarrizkoLangilea);

 setTitle("Birtek - TEKNIKOA (SAT)");
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 setBounds(100, 100, 1050, 650);

 JPanel goikoPanela = new JPanel(new BorderLayout());
 getContentPane().add(goikoPanela, BorderLayout.NORTH);

 JPanel bilatzailePanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
 bilatzailePanela.add(new JLabel("Bilatu: "));
 bilatuTestua = new JTextField(20);
 bilatuTestua.addKeyListener(new KeyAdapter() {
 public void keyReleased(KeyEvent e) {
 filtratu();
 }
 });
 bilatzailePanela.add(bilatuTestua);

 JButton eguneratuBotoia = new JButton("Eguneratu");
 eguneratuBotoia.setBackground(new Color(173, 216, 230)); // Light Blue
 eguneratuBotoia.addActionListener(e -> datuakKargatu());
 bilatzailePanela.add(eguneratuBotoia);

 goikoPanela.add(bilatzailePanela, BorderLayout.WEST);

 // ESKUINALDEA: Erabiltzailea + Fitxaketa + Saioa Itxi
 JPanel eskuinekoPanela = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10,
0));
 JLabel erabiltzaileEtiketa = new JLabel(langilea.getSailaId() + " | " +
langilea.getIzena() + " "
 + langilea.getAbizena());
 erabiltzaileEtiketa.setFont(new Font("SansSerif", Font.BOLD, 12));
 erabiltzaileEtiketa.setForeground(new Color(0, 102, 102));

 // Fitxaketa
 JPanel fitxaketaPanela = new JPanel();
 fitxaketaPanela.setLayout(new BoxLayout(fitxaketaPanela,
BoxLayout.Y_AXIS));
 JPanel botoiPanela = new JPanel(new FlowLayout(FlowLayout.CENTER, 2, 0));

 JButton sarreraBotoia = new JButton("Sarrera");
 sarreraBotoia.setBackground(new Color(34, 139, 34));
 sarreraBotoia.setForeground(Color.BLACK);
 sarreraBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
 sarreraBotoia.addActionListener(e -> fitxatu("Sarrera"));

 JButton irteeraBotoia = new JButton("Irteera");
 irteeraBotoia.setBackground(new Color(255, 140, 0));
 irteeraBotoia.setForeground(Color.BLACK);
 irteeraBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
 irteeraBotoia.addActionListener(e -> fitxatu("Irteera"));

```

```

 JButton historialBotoia = new JButton("Historiala");
 historialBotoia.setBackground(new Color(100, 149, 237));
 historialBotoia.setForeground(Color.BLACK);
 historialBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
 historialBotoia.addActionListener(e -> ikusiFitxaketaHistoriala()));

 JButton nireDatuakBotoia = new JButton("Nire Datuak");
 nireDatuakBotoia.setBackground(new Color(100, 149, 237));
 nireDatuakBotoia.setForeground(Color.BLACK);
 nireDatuakBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
 nireDatuakBotoia.addActionListener(e -> irekiNireDatuakEditatu());

 botoiPanela.add(sarreraBotoia);
 botoiPanela.add(irteeraBotoia);
 botoiPanela.add(historialBotoia);
 botoiPanela.add(nireDatuakBotoia);

 fitxaketaInfoEtiketa = new JLabel("Kargatzen...");
 fitxaketaInfoEtiketa.setFont(new Font("SansSerif", Font.PLAIN, 9));
 fitxaketaInfoEtiketa.setAlignmentX(Component.CENTER_ALIGNMENT);

 fitxaketaPanela.add(botoiPanela);
 fitxaketaPanela.add(fitxaketaInfoEtiketa);

 JButton saioaItxiBotoia = new JButton("Saioa Itxi");
 saioaItxiBotoia.setBackground(Color.RED);
 saioaItxiBotoia.setForeground(Color.WHITE);
 saioaItxiBotoia.addActionListener(e -> saioaItxi()));

 // Atzera botoia logikoki hemen egon daiteke

 eskuinekoPanela.add(erabiltzaileEtiketa);
 eskuinekoPanela.add(fitxaketaPanela);
 eskuinekoPanela.add(saioaItxiBotoia);

 goikoPanela.add(eskuinekoPanela, BorderLayout.EAST);

 JTabbedPane pestainaPanela = new JTabbedPane(JTabbedPane.TOP);
 getContentPane().add(pestainaPanela, BorderLayout.CENTER);

 // --- KONPONKETAK TAB ---
 JPanel konponketaPanela = new JPanel(new BorderLayout());
 konponketaTaula = new JTable();

 konponketaTaula.addMouseListener(new MouseAdapter() {
 @Override
 public void mouseClicked(MouseEvent e) {
 if (e.getClickCount() == 2) {
 int errenkada = konponketaTaula.getSelectedRow();
 if (errenkada != -1) {
 int errenkadaModeloa =
 konponketaTaula.convertRowIndexToModel(errenkada);
 Object idObj =
 konponketaTaula.getModel().getValueAt(errenkadaModeloa, 0);
 if (idObj != null) {
 int idKonponketa = Integer.parseInt(idObj.toString());
 }
 }
 }
 }
 });

```

```

 irekiKonponketaXehetasuna(idKonponketa);
 }
 }
 });
}

JLabel informazioEtiketa = new JLabel(
 "Klik bikoitza egin konponketa bat kudeatzeko");
informazioEtiketa.setHorizontalTextPosition(SwingConstants.CENTER);
informazioEtiketa.setBorder(BorderFactory.createEmptyBorder(5, 0, 5, 0));
konponketaPanela.add(informazioEtiketa, BorderLayout.SOUTH);
konponketaPanela.add(new JScrollPane(konponketaTaula),
BorderLayout.CENTER);
pestainaPanela.addTab("Konponketak", konponketaPanela);

// --- PRODUKTUAK TAB ---
 JPanel produktuPanela = new JPanel(new BorderLayout());
produktuTaula = new JTable();
produktuPanela.add(new JScrollPane(produktuTaula), BorderLayout.CENTER);
pestainaPanela.addTab("Produktuak", produktuPanela);

// --- AKATSAK TAB ---
 JPanel akatsPanela = new JPanel(new BorderLayout());
akatsTaula = new JTable();
akatsPanela.add(new JScrollPane(akatsTaula), BorderLayout.CENTER);
pestainaPanela.addTab("Akatsak", akatsPanela);

pestainaPanela.addChangeListener(e -> {
 bilatuTestua.setText("");
 int index = pestainaPanela.getSelectedIndex();
 if (index == 0)
 unekoOrdenatzailea = konponketaOrdenatzailea;
 else if (index == 1)
 unekoOrdenatzailea = produktuOrdenatzailea;
 else if (index == 2)
 unekoOrdenatzailea = akatsOrdenatzailea;
});

// CRUD Botoiak
 JPanel botoiCrudPanela = new JPanel(new FlowLayout(FlowLayout.CENTER, 10,
5));
 JButton gehituBotoia = new JButton("Gehitu +");
 JButton editatuBotoia = new JButton("Editatu");
 JButton ezabatuBotoia = new JButton("Ezabatu");

 gehituBotoia.addActionListener(e ->
gehituElementua(pestainaPanela.getSelectedIndex()));
 editatuBotoia.addActionListener(e ->
editatuElementua(pestainaPanela.getSelectedIndex()));
 ezabatuBotoia.addActionListener(e ->
ezabatuElementua(pestainaPanela.getSelectedIndex()));

botoiCrudPanela.add(gehituBotoia);
botoiCrudPanela.add(editatuBotoia);
botoiCrudPanela.add(ezabatuBotoia);

```

```

 JButton prezioaBotoia = new JButton("Prezioa Ezarri");
 prezioaBotoia.addActionListener(e ->
prezioaEzarriElementua(pestainaPanela.getSelectedIndex()));
 botoiCrudPanela.add(prezioaBotoia);

 JPanel behekoPanela = new JPanel(new BorderLayout());
 behekoPanela.add(botoiCrudPanela, BorderLayout.NORTH);

 JButton kargatuBotoia = new JButton("Kargatu / Berritu");
 kargatuBotoia.addActionListener(e -> datuakKargatu());
 behekoPanela.add(kargatuBotoia, BorderLayout.CENTER);

 getContentPane().add(behekoPanela, BorderLayout.SOUTH);

 pestainaPanela.addChangeListener(e -> {
 bilatuTestua.setText("");
 int index = pestainaPanela.getSelectedIndex();
 if (index == 0)
 unekoOrdenatzailea = konponketaOrdenatzailea;
 else if (index == 1)
 unekoOrdenatzailea = produktuOrdenatzailea;
 else if (index == 2)
 unekoOrdenatzailea = akatsOrdenatzailea;
 });

 if (!java.beans.Beans.isDesignTime()) {
 datuakKargatu();
 egeneratuFitxaketaEgoera();
 }
}

// --- FITXAKETA METODOAK ---
/***
 * Fitxaketa bat egin.
 *
 * @param mota Fitxaketa mota (Sarrera/Irteera).
 */
private void fitxatu(String mota) {
 try {
 langilea.fitxatu(mota);
 egeneratuFitxaketaEgoera();
 datuakKargatu(); // Refrescar tableros
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, e.getMessage(), "Errorea",
JOptionPane.WARNING_MESSAGE);
 }
}

/**
 * Fitxaketa egoera egunerau interfazean.
 */
private void egeneratuFitxaketaEgoera() {
 fitxaketaInfoEtiketa.setText(langilea.getFitxaketaEgoera());
 if (fitxaketaInfoEtiketa.getText().contains("BARRUAN")) {
 fitxaketaInfoEtiketa.setForeground(new Color(0, 100, 0));
 } else {
 fitxaketaInfoEtiketa.setForeground(new Color(200, 0, 0));
 }
}

```

```

 }

 /**
 * Fitxaketa historia ikusi.
 */
 private void ikusiFitxaketaHistoriala() {
 JDialog elkarritzeta = new JDialog(this, "Nire Fitxaketa Historiala",
true);
 elkarritzeta.setSize(500, 400);
 elkarritzeta.setLocationRelativeTo(this);
 elkarritzeta.setLayout(new BorderLayout());
 String[] zutabeak = { "Mota", "Data", "Ordua" };
 DefaultTableModel eredua = new DefaultTableModel(zutabeak, 0);
 JTable taula = new JTable(eredua);
 taula.getTableHeader().setFont(new Font("SansSerif", Font.BOLD, 12));
 taula.setRowHeight(25);
 elkarritzeta.add(new JScrollPane(taula), BorderLayout.CENTER);

 java.util.List<Fitxaketa> zerrenda = langilea.nireFitxaketakIkusi();
 for (Fitxaketa f : zerrenda) {
 eredua.addRow(new Object[] { f.getMota(), f.getData(), f.getOrdua() });
 }

 JButton itxiBotoia = new JButton("Itxi");
 itxiBotoia.addActionListener(e -> elkarritzeta.dispose());
 JPanel botoiPanela = new JPanel();
 botoiPanela.add(itxiBotoia);
 elkarritzeta.add(botoiPanela, BorderLayout.SOUTH);

 elkarritzeta.setVisible(true);
 }

 /**
 * Norberaren datuak editatzeko leihoa ireki.
 */
 private void irekiNireDatuakEditatu() {
 NireDatuakDialog dialog = new NireDatuakDialog(this, langilea);
 dialog.setVisible(true);
 }

 /**
 * Datu guztiak (Konponketak, Produktuak, Akatsak) kargatu datu-basetik.
 */
 private void datuakKargatu() {
 try (Connection konexioa = DB_Konexioa.konektatu()) {
 DefaultTableModel m1 = TaulaModelatzalea
 .ereduaEraiki(konexioa.prepareStatement("SELECT * FROM
konponketak").executeQuery());
 konponketaTaula.setModel(m1);
 konponketaOrdenatzalea = new TableRowSorter<>(m1);
 konponketaTaula.setRowSorter(konponketaOrdenatzalea);

 DefaultTableModel m2 = TaulaModelatzalea.ereduaEraiki(konexioa
 .prepareStatement(
 "SELECT id_produktua, izena, produktu_egoera,
salmenta_prezioa, eskaintza FROM produktuak")

```

```

 .executeQuery());
produktuTaula.setModel(m2);
produktuOrdenatzailea = new TableRowSorter<>(m2);
produktuTaula.setRowSorter(produktuOrdenatzailea);

DefaultTableModel m3 = TaulaModelatzalea.ereduEraiki(konexioa
 .prepareStatement("SELECT * FROM akatsak").executeQuery());
akatsTaula.setModel(m3);
akatsOrdenatzailea = new TableRowSorter<>(m3);
akatsTaula.setRowSorter(akatsOrdenatzailea);

if (unekoOrdenatzailea == null)
 unekoOrdenatzailea = konponketaOrdenatzailea;
} catch (Exception e) {
 e.printStackTrace();
}
}

/**
 * Hautatutako elementua ezabatu fitxaren arabera.
 *
 * @param index Fitxaren indizea.
 */
private void ezabatuElementua(int index) {
 JTable t = (index == 0) ? konponketaTaula : (index == 1) ? produktuTaula :
akatsTaula;

 if (t.getSelectedRow() == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu elementu bat
ezabatzeko.");
 return;
 }

 int rm = t.convertRowIndexToModel(t.getSelectedRow());
 Object idObj = t.getModel().getValueAt(rm, 0);
 int id = Integer.parseInt(idObj.toString());

 if (JOptionPane.showConfirmDialog(this, "Ziur ID " + id + " ezabatu nahi
duzula?", "Ezabatu",
 JOptionPane.YES_NO_OPTION) == 0) {
 try {
 if (index == 0) { // Konponketak
 langilea.konponketaEzabatu(id);
 } else if (index == 1) { // Produktuak
 langilea.produktuaBorratu(id);
 } else if (index == 2) { // Akatsak
 langilea.akatsaEzabatu(id);
 }
 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Elementua ezabatu da.");
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
 }
 }
}

/**
 */

```

```

* Elementu berria gehitu fitxaren arabera.
*
* @param index Fitxaren indizea.
*/
private void gehituElementua(int index) {
 if (index == 0) { // Konponketak
 try {
 // 1. Datuak lortu
 java.util.List<Produktua> produktuak = langilea.produktuakIkusi();
 java.util.List<Akatsa> akatsak = langilea.akatsaIkusi();

 if (produktuak.isEmpty()) {
 JOptionPane.showMessageDialog(this, "Ez dago produkturik
sisteman.");
 return;
 }
 if (akatsak.isEmpty()) {
 JOptionPane.showMessageDialog(this, "Ez dago akats motarik
sisteman. Sortu bat lehenik.");
 return;
 }

 // 2. UI Konponenteak
 JComboBox<ComboItem> produktuBox = new JComboBox<>();
 for (Produktua p : produktuak) {
 produktuBox.addItem(new ComboItem(p.getIdProduktua(),
p.getIzena() + " (" + p.getMarka() + ")"));
 }

 JComboBox<ComboItem> akatsBox = new JComboBox<>();
 for (Akatsa a : akatsak) {
 akatsBox.addItem(new ComboItem(a.getIdAkatsa(), a.getIzena()));
 }

 JTextArea oharrakArea = new JTextArea(5, 20);

 Object[] message = {
 "Produktua:", produktuBox,
 "Akatsa:", akatsBox,
 "Oharrak:", new JScrollPane(oharrakArea)
 };

 int option = JOptionPane.showConfirmDialog(this, message, "Sortu
Konponketa Berria",
 JOptionPane.OK_CANCEL_OPTION);

 if (option == JOptionPane.OK_OPTION) {
 ComboItem selectedProd = (ComboItem)
produktuBox.getSelectedItem();
 ComboItem selectedAkats = (ComboItem)
akatsBox.getSelectedItem();
 String oharrak = oharrakArea.getText();

 if (selectedProd == null || selectedAkats == null)
 return;

 // 3. Konponketa sortu

```

```

Konponketa k = new Konponketa(
 0, // ID
 selectedProd.getId(),
 langilea.getIdLangilea(),
 new Timestamp(System.currentTimeMillis()), // Hasiera
data
 null, // Amaiera data
 "Prozesuan",
 selectedAkats.getId(),
 oharrak,
 null);

langilea.konponketaEgin(k);
datuakKargatu();
JOptionPane.showMessageDialog(this, "Konponketa ondo sortu
da!");
}
} catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea: " + e.toString() +
"\n" + e.getMessage(), "Errorea",
 JOptionPane.ERROR_MESSAGE);
}

} else if (index == 1) { // Produktuak
 try (Connection kon = DB_Konexioa.konektatu()) {
 // 1. Datuak lortu (Kategoriak eta Hornitzailak)
 java.util.List<ProduktuKategoria> kategoriak = new ArrayList<>();
 ResultSet rsK = kon.prepareStatement("SELECT * FROM
produktu_kategoriarak").executeQuery();
 while (rsK.next()) {
 kategoriak.add(new
ProduktuKategoria(rsK.getInt("id_kategoria"), rsK.getString("izena")));
 }

 java.util.List<Hornitzaila> hornitzailak = new ArrayList<>();
 ResultSet rsH = kon.prepareStatement("SELECT id_hornitzaila,
izena_soziala FROM hornitzailak")
 .executeQuery();
 while (rsH.next()) {
 hornitzailak.add(new
Hornitzaila(rsH.getInt("id_hornitzaila"), rsH.getString("izena_soziala"),
 null, null, null, 1, null, null, null, null, null,
true, null));
 }

 java.util.List<Biltegia> biltegiak = new ArrayList<>();
 ResultSet rsB = kon.prepareStatement("SELECT id_biltegia, izena
FROM biltegiak").executeQuery();
 while (rsB.next()) {
 biltegiak.add(new Biltegia(rsB.getInt("id_biltegia"),
rsB.getString("izena"), ""));
 }

 // 2. UI Konponenteak
 JTextField izenaField = new JTextField();
 JTextField markaField = new JTextField();

```

```

 JTextField motaField = new JTextField();
 JComboBox<ComboItem> kategoriaBox = new JComboBox<>();
 for (ProduktuKategoria pk : kategoriak) {
 kategoriaBox.addItem(new ComboItem(pk.getIdKategoria(),
pk.getIzena())));
 }
 JComboBox<ComboItem> hornitzaireBox = new JComboBox<>();
 for (Hornitzairea h : hornitzaireak) {
 hornitzaireBox.addItem(new ComboItem(h.getIdHornitzairea(),
h.getIzenaSoziala())));
 }
 JComboBox<ComboItem> biltegiBox = new JComboBox<>();
 for (Biltegia b : biltegiak) {
 biltegiBox.addItem(new ComboItem(b.getIdBiltegia(),
b.getIzena())));
 }

 JTextField stockField = new JTextField("0");
 String[] egoerak = { "Berria", "Berritua A", "Berritua B",
"Hondatua", "Zehazteko" };
 JComboBox<String> egoeraBox = new JComboBox<>(egoerak);
 JTextArea deskribapenaArea = new JTextArea(3, 20);
 JCheckBox salgaiBox = new JCheckBox("Salgai", true);

 Object[] message = {
 "Izena:", izenaField,
 "Marka:", markaField,
 "Mota:", motaField,
 "Kategoria:", kategoriaBox,
 "Hornitzairea:", hornitzaireBox,
 "Biltegia:", biltegiBox,
 "Stock:", stockField,
 "Egoera:", egoeraBox,
 "Deskribapena:", new JScrollPane(deskribapenaArea),
 "Salgai:", salgaiBox
 };
 }

 int option = JOptionPane.showConfirmDialog(this, message, "Produktu
Berria Sortu",
JOptionPane.OK_CANCEL_OPTION);

 if (option == JOptionPane.OK_OPTION) {
 ComboItem selKat = (ComboItem) kategoriaBox.getSelectedItem();
 ComboItem selHorItem = (ComboItem)
hornitzaireBox.getSelectedItem();
 ComboItem selBilItem = (ComboItem)
biltegiBox.getSelectedItem();

 Produktua p = new Produktua(
 0, // ID
 selHorItem != null ? selHorItem.getId() : 0,
 selKat != null ? selKat.getId() : 0,
 izenaField.getText(),
 markaField.getText(),
 motaField.getText(),
 deskribapenaArea.getText(),
 "", // Irudia URL
 ""
);
 }
}

```

```

 selBilItem != null ? selBilItem.getId() : null, //
Biltegi ID
 (String) egoeraBox.getSelectedItem(),
 "", // Egoera oharra
 salgaiBox.isSelected(),
 java.math.BigDecimal.ZERO, // Prezioa
 0, // Stock (set below)
 java.math.BigDecimal.ZERO, // Eskaintza
 new java.math.BigDecimal("21.00"), // Zergak
 null, null) {
 };
 p.setStock(Integer.parseInt(stockField.getText()));

 langilea.produktuBatSortu(p);
 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Produktua ondo sortu
da!");
}
} catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea produktua sortzean: "
+ e.getMessage());
}

} else if (index == 2) { // Akatsak
 JTextField izenaField = new JTextField();
 JTextArea deskribapenaArea = new JTextArea(5, 20);

 Object[] message = {
 "Izena (Derrigorrezkoa):", izenaField,
 "Deskribapena:", new JScrollPane(deskribapenaArea)
 };

 int option = JOptionPane.showConfirmDialog(this, message, "Gehitu
Akatsa", JOptionPane.OK_CANCEL_OPTION);

 if (option == JOptionPane.OK_OPTION) {
 String izena = izenaField.getText().trim();
 String deskribapena = deskribapenaArea.getText().trim();

 if (izena.isEmpty()) {
 JOptionPane.showMessageDialog(this, "Izena derrigorrezko da.",
"Errorea",
 JOptionPane.ERROR_MESSAGE);
 return;
 }

 try {
 Akatsa a = new Akatsa(0, izena, deskribapena);
 langilea.akatsaSortu(a);
 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Akatsa ondo sortu da.");
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea sortzean: " +
e.getMessage(), "Errorea",
 JOptionPane.ERROR_MESSAGE);
 }
 }
}

```

```

 }
 } else {
 JOptionPane.showMessageDialog(this, "Funtzio hau oraindik ez dago
erabilgarri.");
 }
}

/**
 * Hautatutako elementua editatu fitxaren arabera.
 *
 * @param index Fitxaren indizea.
 */
private void editatuElementua(int index) {
 if (index == 0) { // Konponketak
 int r = konponketaTaula.getSelectedRow();
 if (r == -1)
 return;
 }

irekiKonponketaXehetasuna(Integer.parseInt(konponketaTaula.getValueAt(r,
0).toString()));
 } else if (index == 1) { // Produktuak
 int r = produktuTaula.getSelectedRow();
 if (r == -1)
 return;
 int rm = produktuTaula.convertRowIndexToModel(r);
 Object idObj = produktuTaula.getModel().getValueAt(rm, 0);
 int id = Integer.parseInt(idObj.toString());

 try (Connection kon = DB_Konexioa.konektatu()) {
 // Gaur egungo datuak lortu
 PreparedStatement pst = kon.prepareStatement("SELECT * FROM
produktuak WHERE id_produktua = ?");
 pst.setInt(1, id);
 ResultSet rs = pst.executeQuery();
 if (rs.next()) {
 JTextField izenaField = new JTextField(rs.getString("izena"));
 JTextField markaField = new JTextField(rs.getString("marka"));
 JTextField egoeraField = new
JTextField(rs.getString("produktu_egoera"));
 JCheckBox salgaiBox = new JCheckBox("Salgai",
rs.getBoolean("salgai"));
 JTextField prezioaField = new
JTextField(rs.getString("salmenta_precioa"));

 Object[] message = {
 "Izena:", izenaField,
 "Marka:", markaField,
 "Egoera (Berria, Berritua A, Berritua B, Hondatua,
Zehazteko):", egoeraField,
 "Salgai:", salgaiBox,
 "Prezioa (€):", prezioaField
 };

 int option = JOptionPane.showConfirmDialog(null, message,
"Editatu Produktua",
 JOptionPane.OK_CANCEL_OPTION);
 if (option == JOptionPane.OK_OPTION) {

```

```

 // Update basic info
 PreparedStatement upst = kon.prepareStatement(
 "UPDATE produktuak SET izena = ?, marka = ?,"
 + "produktu_egoera = ?, salgai = ?, salmenta_prezioa = ? WHERE id_produktua = ?");
 upst.setString(1, izenaField.getText());
 upst.setString(2, markaField.getText());
 upst.setString(3, egoeraField.getText());
 upst.setBoolean(4, salgaiBox.isSelected());
 upst.setBigDecimal(5, new
java.math.BigDecimal(prezioaField.getText().replace(",",".")));
 upst.setInt(6, id);
 upst.executeUpdate();

 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Produktua eguneratu
da.");
 }
}

} catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea editatzean: " +
e.getMessage());
}
} else if (index == 2) { // Akatsak
 int r = akatsTaula.getSelectedRow();
 if (r == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu akats bat
editatzeko.");
 return;
 }

 // Ereduko balioak lortu (View -> Model konbertsioa)
 int rm = akatsTaula.convertRowIndexToModel(r);
 // Suposatzen dugu zutabeen ordena: [0]=ID, [1]=Izena, [2]=Deskribapena
 // Egiaztatu behar da datuakKargatu() metodoan zutabeen ordena
 Object idObj = akatsTaula.getModel().getValueAt(rm, 0);
 Object izenaObj = akatsTaula.getModel().getValueAt(rm, 1);
 Object deskribapenaObj = akatsTaula.getModel().getValueAt(rm, 2);

 int idAkatsa = Integer.parseInt(idObj.toString());
 String unekoIzena = izenaObj != null ? izenaObj.toString() : "";
 String unekoDeskr = deskribapenaObj != null ?
deskribapenaObj.toString() : "";

 JTextField izenaField = new JTextField(unekoIzena);
 JTextArea deskribapenaArea = new JTextArea(unekoDeskr);
 deskribapenaArea.setRows(5);
 deskribapenaArea.setColumns(20);

 Object[] message = {
 "Izena:", izenaField,
 "Deskribapena:", new JScrollPane(deskribapenaArea)
 };

 int option = JOptionPane.showConfirmDialog(this, message, "Editatu
Akatsa", JOptionPane.OK_CANCEL_OPTION);
}

```

```

 if (option == JOptionPane.OK_OPTION) {
 String izenaBerria = izenaField.getText().trim();
 String deskribapenaBerria = deskribapenaArea.getText().trim();

 if (izenaBerria.isEmpty()) {
 JOptionPane.showMessageDialog(this, "Izena ezin da hutsik
egon.", "Errorea",
 JOptionPane.ERROR_MESSAGE);
 return;
 }

 try {
 Akatsa a = new Akatsa(idAkatsa, izenaBerria,
deskribapenaBerria);
 langilea.akatsaEditatu(a);
 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Akatsa eguneratu da.");
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea editatzean: " +
e.getMessage(), "Errorea",
 JOptionPane.ERROR_MESSAGE);
 }
 }
 } else {
 JOptionPane.showMessageDialog(this, "Funtzio hau oraindik ez dago
erabilgarri.");
 }
}

/**
 * Produktu baten prezioa eta eskaintza ezarri.
 *
 * @param index Fitxaren indizea (Produktuak fitxa izan behar du).
 */
private void prezioaEzarriElementua(int index) {
 if (index != 1) { // 1 = Produktuak
 JOptionPane.showMessageDialog(this, "Aukera hau produktuetan bakarrik
dago erabilgarri.");
 return;
 }

 int r = productoTaula.getSelectedRow();
 if (r == -1) {
 JOptionPane.showMessageDialog(this, "Aukeratu produktu bat prezioa
ezartzeko.");
 return;
 }

 int rm = productoTaula.convertRowIndexToModel(r);
 Object idObj = productoTaula.getModel().getValueAt(rm, 0);
 int idProduktua = Integer.parseInt(idObj.toString());

 JTextField prezioaField = new JTextField();
 JTextField eskaintzaField = new JTextField();
 Object[] message = {
 "Salmenta Prezioa (€):", prezioaField,
 "Eskaintza (€) (Aukerazkoa):", eskaintzaField
 }
}

```

```

};

 int option = JOptionPane.showConfirmDialog(this, message, "Ezarri Prezioa",
JOptionPane.OK_CANCEL_OPTION);
 if (option == JOptionPane.OK_OPTION) {
 try {
 String pStr = prezioaField.getText().replace(", ", ".").trim();
 String eStr = eskaintzaField.getText().replace(", ", ".").trim();

 if (pStr.isEmpty()) {
 JOptionPane.showMessageDialog(this, "Prezioa derrigorrezkoa
da.");
 return;
 }

 java.math.BigDecimal prezioa = new java.math.BigDecimal(pStr);
 java.math.BigDecimal eskaintza = null;
 if (!eStr.isEmpty()) {
 eskaintza = new java.math.BigDecimal(eStr);
 }

 langilea.prezioaEzarri(idProduktua, prezioa, eskaintza);
 datuakKargatu();
 JOptionPane.showMessageDialog(this, "Prezioa eguneratu da.");

 } catch (NumberFormatException e) {
 JOptionPane.showMessageDialog(this, "Mesedez, sartu zenbaki
baliodunak (adib. 10.50).", "Errorea",
 JOptionPane.ERROR_MESSAGE);
 } catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage(),
"Errorea", JOptionPane.ERROR_MESSAGE);
 }
 }

}

/**
 * Konponketa baten xehetasunak editatzeko leihoa ireki.
 *
 * @param idKonponketa Konponketaren IDa.
 */
private void irekiKonponketaXehetasuna(int idKonponketa) {
 try {
 // Balioak lortu taulatik
 int errenkada = konponketaTaula.getSelectedRow();
 String egoera = konponketaTaula.getValueAt(errenkada, 5).toString();
 // 6. zutabea akatsa_id da, 7.a oharrak
 int akatsaId = Integer.parseInt(konponketaTaula.getValueAt(errenkada,
6).toString());
 Object oharrakObj = konponketaTaula.getValueAt(errenkada, 7);
 String oharrak = (oharrakObj != null) ? oharrakObj.toString() : "";

 java.util.List<Akatsa> zerrenda = langilea.akatsaIkusi();

 KonponketaXehetasunaElkarritzeta elkarritzeta = new
KonponketaXehetasunaElkarritzeta(

```

```

 idKonponketa, egoera, oharrak, akatsaId, zerrenda);
elkarrikzeta.setModal(true);
elkarrikzeta.setVisible(true);
datuakKargatu();

} catch (SQLException e) {
 JOptionPane.showMessageDialog(this, "Errorea datuak kargatzean: " +
e.getMessage());
} catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea: " + e.getMessage());
}
}

/**
 * Taulak iragazi bilaketa testuaren arabera.
 */
private void filtratu() {
 String t = bilatuTestua.getText();
 if (unekoOrdenatzailea != null) {
 if (t.isEmpty())
 unekoOrdenatzailea.setRowFilter(null);
 else
 unekoOrdenatzailea.setRowFilter(RowFilter.regexFilter("(?i)" + t));
 }
}

/**
 * Saioa itxi.
 */
private void saioaItxi() {
 if (JOptionPane.showConfirmDialog(this, "Irten?", "Saioa Itxi",
JOptionPane.YES_NO_OPTION) == 0) {
 dispose();
 new SaioaHastekoPanela().setVisible(true);
 }
}

// Helper class for ComboBox items
/**
 * ComboBox-erako elementu laguntzailea.
 */
private static class ComboItem {
 private int id;
 private String label;

 public ComboItem(int id, String label) {
 this.id = id;
 this.label = label;
 }

 public int getId() {
 return id;
 }

 @Override
 public String toString() {

```

```
 return label;
 }
}
}
```

# MenuZuzendaritza.java

```
package ui;

import db.DB_Konexioa;
import model.*;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.*;

/**
 * MenuZuzendaritza klasea.
 * Zuzendaritza edo Super Admin erabiltzaileen interfazea.
 * Beste menu guztieta sartzeko aukera ematen du.
 */
public class MenuZuzendaritza extends JFrame {

 private static final long serialVersionUID = 1L;
 private Langilea langilea;
 private String erabiltzaileSaila;

 // Fitxaketa
 private JLabel fitxaketaInfoEtiketa;

 /**
 * MenuZuzendaritza eraikitzailea.
 *
 * @param id Erabiltzailearen IDa.
 * @param izena Izena.
 * @param abizena Abizena.
 * @param saila Sailaren izena.
 */
 public MenuZuzendaritza(int id, String izena, String abizena, String saila) {
 this.langilea = new Langilea(id, izena, abizena, "", null, 0, "", "", "",
 "", "ES", "", "", null, null, true,
 1, "", null);
 this.erabiltzaileSaila = saila;

 setTitle("Birtek - SISTEMAK (Super Admin)");
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 setBounds(100, 100, 800, 500);
 getContentPane().setLayout(new BorderLayout());
 }

 // HEADER
```

```

 JPanel goikoPanela = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10, 5));

 JLabel erabiltzaileEtiketa = new JLabel(erabiltzaileSaila + " | " +
langilea.getIzena() + " "
+ langilea.getAbizena() + " (ID: " + langilea.getIdLangilea() +
")");
erabiltzaileEtiketa.setFont(new Font("SansSerif", Font.BOLD, 12));
erabiltzaileEtiketa.setForeground(new Color(0, 102, 102));

// ... (fitxaketa panela eta besteak berdin mantentzen dira, baina
fitxatu()
// metodoak langilea erabiliko du)
// Fitxaketa Panela
JPanel fitxaketaPanela = new JPanel();
fitxaketaPanela.setLayout(new BoxLayout(fitxaketaPanela,
BoxLayout.Y_AXIS));
JPanel botoiPanela = new JPanel(new FlowLayout(FlowLayout.CENTER, 2, 0));

JButton sarreraBotoia = new JButton("Sarrera");
sarreraBotoia.setBackground(new Color(34, 139, 34));
sarreraBotoia.setForeground(Color.BLACK);
sarreraBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
sarreraBotoia.addActionListener(e -> fitxatu("Sarrera"));

JButton irteeraBotoia = new JButton("Irteera");
irteeraBotoia.setBackground(new Color(255, 140, 0));
irteeraBotoia.setForeground(Color.BLACK);
irteeraBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
irteeraBotoia.addActionListener(e -> fitxatu("Irteera"));

JButton historialBotoia = new JButton("Historiala");
historialBotoia.setBackground(new Color(100, 149, 237));
historialBotoia.setForeground(Color.BLACK);
historialBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
historialBotoia.addActionListener(e -> ikusiFitxaketaHistoriala()));

botoiPanela.add(sarreraBotoia);
botoiPanela.add(irteeraBotoia);
botoiPanela.add(historialBotoia);

JButton nireDatuakBotoia = new JButton("Nire Datuak");
nireDatuakBotoia.setBackground(new Color(100, 149, 237));
nireDatuakBotoia.setForeground(Color.BLACK);
nireDatuakBotoia.setFont(new Font("SansSerif", Font.BOLD, 10));
nireDatuakBotoia.addActionListener(e -> irekiNireDatuakEditatu());
botoiPanela.add(nireDatuakBotoia);

fitxaketaInfoEtiketa = new JLabel("Kargatzen...");
fitxaketaInfoEtiketa.setFont(new Font("SansSerif", Font.PLAIN, 9));
fitxaketaInfoEtiketa.setAlignmentX(Component.CENTER_ALIGNMENT);

fitxaketaPanela.add(botoiPanela);
fitxaketaPanela.add(fitxaketaInfoEtiketa);

goikoPanela.add(erabiltzaileEtiketa);
goikoPanela.add(fitxaketaPanela);
getContentPane().add(goikoPanela, BorderLayout.NORTH);

```

```

// BOTONES CENTRALES
JPanel botoiPanelaNagusia = new JPanel(new GridLayout(3, 2, 15, 15));
botoiPanelaNagusia.setBorder(BorderFactory.createEmptyBorder(20, 20, 20,
20));

JButton adminBotoia = new JButton("ADMINISTRAZIOA");
adminBotoia.addActionListener(e -> {
 new MenuAdministrazioa(langilea).setVisible(true);
});

JButton teknikariBotoia = new JButton("TEKNIKOA");
teknikariBotoia.addActionListener(e -> {
 new MenuTeknikoa(langilea).setVisible(true);
});

JButton salmentaBotoia = new JButton("SALMENTAK");
salmentaBotoia.addActionListener(e -> {
 new MenuSalmentak(langilea).setVisible(true);
});

JButton logistikaBotoia = new JButton("LOGISTIKA");
logistikaBotoia.addActionListener(e -> {
 new MenuLogistika(langilea).setVisible(true);
});

JButton probaBotoia = new JButton("DB CHECK");
probaBotoia.addActionListener(e -> JOptionPane.showMessageDialog(this,
"Konexioa OK"));

JButton saioaItxiBotoia = new JButton("SAIOA ITXI");
saioaItxiBotoia.setBackground(Color.RED);
saioaItxiBotoia.setForeground(Color.WHITE);
saioaItxiBotoia.addActionListener(e -> saioaItxi());

botoiPanelaNagusia.add(adminBotoia);
botoiPanelaNagusia.add(teknikariBotoia);
botoiPanelaNagusia.add(salmentaBotoia);
botoiPanelaNagusia.add(logistikaBotoia);
botoiPanelaNagusia.add(probaBotoia);
botoiPanelaNagusia.add(saioaItxiBotoia);

getContentPane().add(botoiPanelaNagusia, BorderLayout.CENTER);

if (!java.beans.Beans.isDesignTime()) {
 eguneratuFitxaketaEgoera();
}
}

// --- FITXAKETA LOGIKA (Berdina) ---
/**
 * Fitxaketa egin.
 *
 * @param mota Fitxaketa mota.
 */
private void fitxatu(String mota) {
 try {

```

```

 langilea.fitxatu(mota);
 eguneratuFitxaketaEgoera();
 } catch (SQLException e) {
 JOptionPane.showMessageDialog(this, e.getMessage(), "Errorea",
JOptionPane.WARNING_MESSAGE);
 }
}

/**
 * Fitxaketa egoera eguneratu.
 */
private void eguneratuFitxaketaEgoera() {
 String galdera = "SELECT mota, data, ordua FROM fitxaketak WHERE
langilea_id = ? ORDER BY id_fitxaketa DESC LIMIT 1";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement sententzia = konexioa.prepareStatement(galdera))
{
 sententzia.setInt(1, langilea.getIdLangilea());
 ResultSet rs = sententzia.executeQuery();
 if (rs.next()) {
 String mota = rs.getString("mota");
 Time ordua = rs.getTime("ordua");
 if ("Sarrera".equals(mota)) {
 fitxaketaInfoEtiketa.setText("BARRUAN (" + ordua + ")");
 fitxaketaInfoEtiketa.setForeground(new Color(0, 100, 0));
 } else {
 fitxaketaInfoEtiketa.setText("KANPOAN (" + ordua + ")");
 fitxaketaInfoEtiketa.setForeground(new Color(200, 0, 0));
 }
 } else {
 fitxaketaInfoEtiketa.setText("Ez dago erregistrorik.");
 fitxaketaInfoEtiketa.setForeground(Color.GRAY);
 }
} catch (SQLException e) {
 e.printStackTrace();
}
}

/**
 * Fitxaketa historia ikusi.
 */
private void ikusiFitxaketaHistoriala() {
 JDialog elkarrizketa = new JDialog(this, "Fitxaketa Historiala", true);
 elkarrizketa.setSize(500, 400);
 elkarrizketa.setLocationRelativeTo(this);
 elkarrizketa.setLayout(new BorderLayout());
 String[] zutabeak = { "Mota", "Data", "Ordua" };
 DefaultTableModel eredu = new DefaultTableModel(zutabeak, 0);
 JTable taula = new JTable(eredu);
 elkarrizketa.add(new JScrollPane(taula), BorderLayout.CENTER);

 String galdera = "SELECT mota, data, ordua FROM fitxaketak WHERE
langilea_id = ? ORDER BY id_fitxaketa DESC";
 try (Connection konexioa = DB_Konexioa.konektatu();
 PreparedStatement sententzia = konexioa.prepareStatement(galdera))
{
 sententzia.setInt(1, langilea.getIdLangilea());
}
}

```

```

 ResultSet rs = sententzia.executeQuery();
 while (rs.next()) {
 eredu.addRow(new Object[] { rs.getString("mota"),
rs.getDate("data"), rs.getTime("ordua") });
 }
 } catch (SQLException e) {
 e.printStackTrace();
 }
 elkarritzeta.setVisible(true);
}

/**
 * Norberaren datuak editatzeko leihoa ireki.
 */
private void irekiNireDatuakEditatu() {
 // MenuZuzendaritza uses separate fields, so we need to construct a
Langilea
 // object
 // Query DB to get full details properly if needed, but for now we create a
 // temporary one with available data.
 // Ideally, MenuZuzendaritza should also hold a Langilea object like
others.

 // We need current password and other details. Let's fetch them first or
create
 // a Langilea and refresh after.
 Langilea l = new Langilea(langilea.getIdLangilea(), langilea.getIzena(),
langilea.getAbizena(), "", null, 0, "",
 "", "", "", "ES", "", "", null, null, true, 1, "", null);

 // Retrieve actual data from DB to ensure we have the latest (password,
town,
 // etc.)
 try (Connection k = DB_Konexioa.konektatu();
 PreparedStatement ps = k.prepareStatement("SELECT * FROM langileak
WHERE id_langilea = ?")) {
 ps.setInt(1, langilea.getIdLangilea());
 ResultSet rs = ps.executeQuery();
 if (rs.next()) {
 l = new Langilea(
 rs.getInt("id_langilea"),
 rs.getString("izena"),
 rs.getString("abizena"),
 rs.getString("nan_ifz"),
 rs.getDate("jaiotze_data"),
 rs.getInt("herria_id"),
 rs.getString("helbidea"),
 rs.getString("posta_kodea"),
 "", // telefonoa (missing in DB query or handled by
pertsona?) - Defaulting to empty
 rs.getString("emaila"),
 rs.getString("hizkuntza"),
 rs.getString("pasahitza"),
 "", // saltoTxartelaUid
 rs.getTimestamp("kontratazio_data"), // altaData
 rs.getTimestamp("eguneratze_data"),

```

```

 rs.getBoolean("aktibo"),
 rs.getInt("saila_id"),
 "", // iban
 null // kurrikuluma
);
 }
 } catch (Exception e) {
 e.printStackTrace();
 }

 NireDatuakDialog dialog = new NireDatuakDialog(this, 1);
 dialog.setVisible(true);
}

/**
 * Saioa itxi.
 */
private void saioaItxi() {
 if (JOptionPane.showConfirmDialog(this, "Irten?", "Saioa Itxi",
JOptionPane.YES_NO_OPTION) == 0) {
 dispose();
 new SaioaHastekoPanela().setVisible(true);
 }
}
}

```

## NireDatuakDialog.java

```

package ui;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.sql.SQLException;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;

import model.Herria;
import model.Langilea;

/**

```

```
* NireDatuakDialog klasea.
* Erabiltzaileak bere datu pertsonalak (Pasahitza, Hizkuntza, Herria,
* Telefona, Helbidea) editatzeko elkarritzeta leihoa.
*/
public class NireDatuakDialog extends JDialog {

 private static final long serialVersionUID = 1L;
 private final JPanel contentPanel = new JPanel();

 private JPasswordField pasahitzaField;
 private JComboBox<String> hizkuntzaBox;
 private JComboBox<Herria> herriaBox;
 private JButton herriaGehituBtn;
 private JTextField telefonoaField;
 private JTextField helbideaField;

 // Modeloa
 private Langilea langilea;

 /**
 * Create the dialog.
 */
 /**
 * Eraikitzailea.
 *
 * @param parent Guraso leihoa.
 * @param langilea Editatu beharreko langile objektua.
 */
 public NireDatuakDialog(JFrame parent, Langilea langilea) {
 super(parent, "Nire Datuak Editatu", true);
 this.langilea = langilea;

 setBounds(100, 100, 450, 300);
 getContentPane().setLayout(new BorderLayout());
 contentPanel.setBorder(new EmptyBorder(10, 10, 10, 10));
 getContentPane().add(contentPanel, BorderLayout.CENTER);
 contentPanel.setLayout(new GridLayout(0, 2, 10, 10));

 // --- PASAHITZA ---
 contentPanel.add(new JLabel("Pasahitza Berria:"));
 pasahitzaField = new JPasswordField();
 contentPanel.add(pasahitzaField);

 // --- HIZKUNTA ---
 contentPanel.add(new JLabel("Hizkuntza:"));
 String[] hizkuntzak = { "Euskara", "Gaztelania", "Frantsesa", "Ingelesa" };
 hizkuntzaBox = new JComboBox<>(hizkuntzak);
 contentPanel.add(hizkuntzaBox);

 // --- HERRIA ---
 contentPanel.add(new JLabel("Herria:"));
 JPanel herriaPanela = new JPanel(new BorderLayout());
 herriaBox = new JComboBox<>();
 herriaGehituBtn = new JButton("+");
 herriaGehituBtn.setToolTipText("Gehitu Herria");
 herriaPanela.add(herriaBox, BorderLayout.CENTER);
 herriaPanela.add(herriaGehituBtn, BorderLayout.EAST);
```

```

contentPanel.add(herriaPanela);

// --- TELEFONOA ---
contentPanel.add(new JLabel("Telefonoa:"));
telefonoaField = new JTextField();
contentPanel.add(telefonoaField);

// --- HELBIDEA ---
contentPanel.add(new JLabel("Helbidea:"));
helbideaField = new JTextField();
contentPanel.add(helbideaField);

// --- LOGIKA ---
herriaGehituBtn.addActionListener(e -> herriaBerriaGehitu());

// Datuak kargatu
hasierakoDatuakJarri();

// --- BOTOIAK ---
JPanel buttonPane = new JPanel();
buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));
getContentPane().add(buttonPane, BorderLayout.SOUTH);

JButton okButton = new JButton("Gorde Aldaketak");
okButton.addActionListener(e -> gordeDatuak());
buttonPane.add(okButton);
getRootPane().setDefaultButton(okButton);

JButton cancelButton = new JButton("Ezeztatu");
cancelButton.addActionListener(e -> dispose());
buttonPane.add(cancelButton);
}

/**
 * Hasierako datuak kargatu interfazean (DBtik freskatuz).
 */
private void hasierakoDatuakJarri() {
 // DBtik datuak freskatzu ziurtatzeko
 try (java.sql.Connection kon = db.DB_Konexioa.konektatu();
 java.sql.PreparedStatement pst = kon
 .prepareStatement("SELECT * FROM langileak WHERE
id_langilea = ?")) {
 pst.setInt(1, langilea.getIdLangilea());
 java.sql.ResultSet rs = pst.executeQuery();
 if (rs.next()) {
 pasahitzaField.setText(rs.getString("pasahitza"));
 hizkuntzaBox.setSelectedItem(rs.getString("hizkuntza"));
 telefonoaField.setText(rs.getString("telefonoa"));
 helbideaField.setText(rs.getString("helbidea"));

 // Update local object to match DB
 langilea.setPasahitza(rs.getString("pasahitza"));
 langilea.setHizkuntza(rs.getString("hizkuntza"));
 langilea.setTelefonoa(rs.getString("telefonoa"));
 langilea.setHelbidea(rs.getString("helbidea"));
 langilea.setHerriaId(rs.getInt("herria_id"));
 } else {

```

```

 // Fallback if DB fail (shouldn't happen)
 pasahitzField.setText(langilea.getPasahitz());
 hizkuntzaBox.setSelectedItem(langilea.getHizkuntza());
 telefonoaField.setText(langilea.getTelefona());
 helbideaField.setText(langilea.getHelbidea());
 }
} catch (Exception e) {
 e.printStackTrace();
 // Fallback
 pasahitzField.setText(langilea.getPasahitz());
 hizkuntzaBox.setSelectedItem(langilea.getHizkuntza());
 telefonoaField.setText(langilea.getTelefona());
 helbideaField.setText(langilea.getHelbidea());
}
}

// Herriak DBtik kargatu
herriakKargatu();

// Langilearen herria hautatu ID bidez
int hId = langilea.getHerriaId();
for (int i = 0; i < herriaBox.getItemCount(); i++) {
 if (herriaBox.getItemAt(i).getIdHerria() == hId) {
 herriaBox.setSelectedIndex(i);
 break;
 }
}
}

/**
 * Herrien zerrenda kargatu ComboBoxean.
 */
private void herriakKargatu() {
 herriaBox.removeAllItems();
 if (langilea != null) {
 List<Herria> zerrenda = langilea.herriakLortu();
 for (Herria h : zerrenda) {
 herriaBox.addItem(h);
 }
 }
}

/**
 * Herri berria gehitzeko leihoa ireki eta sortu.
 */
private void herriaBerriaGehitu() {
 JTextField izenaF = new JTextField();
 JTextField lurrealdeaF = new JTextField();
 JTextField nazioaF = new JTextField();

 Object[] mezua = {
 "Herria:", izenaF,
 "Lurrealdea:", lurrealdeaF,
 "Nazioa:", nazioaF
 };

 int opt = JOptionPane.showConfirmDialog(this, mezua, "Gehitu Herria",
JOptionPane.OK_CANCEL_OPTION);
}

```

```

 if (opt == JOptionPane.OK_OPTION) {
 if (izenaf.getText().trim().isEmpty()) {
 JOptionPane.showMessageDialog(this, "Izena derrigorrezkoa da.");
 return;
 }
 try {
 Herria hBerria = new Herria(0, izenaf.getText(),
lurraldeaF.getText(), nazioaF.getText());
 langilea.herriaSortu(hBerria);
 herriakKargatu();
 // Aukeratu berria
 for (int i = 0; i < herriaBox.getItemCount(); i++) {
 Herria h = herriaBox.getItemAt(i);
 if (h.getIzena().equalsIgnoreCase(izenaf.getText())) {
 herriaBox.setSelectedIndex(i);
 break;
 }
 }
 } catch (Exception e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea herria sortzean: " +
e.getMessage());
 }
 }
}

/**
 * Datuak gorde datu-basean.
 */
private void gordeDatuak() {
 String pass = new String(pasahitzaField.getPassword());
 String hizkunza = (String) hizkunzaBox.getSelectedItem();
 Herria aukeratua = (Herria) herriaBox.getSelectedItem();
 String tlf = telefonoaField.getText().trim();
 String helbidea = helbideaField.getText().trim();

 if (pass.isEmpty()) {
 JOptionPane.showMessageDialog(this, "Pasahitza ezin da hutsik egon.");
 return;
 }
 if (aukeratua == null) {
 JOptionPane.showMessageDialog(this, "Herria aukeratu behar da.");
 return;
 }

 try {
 langilea.nireLangileDatuakEditatu(pass, hizkunza,
aukeratua.getIdHerria(), tlf, helbidea);
 JOptionPane.showMessageDialog(this, "Datuak ondo eguneratu dira.");
 dispose();
 } catch (SQLException e) {
 e.printStackTrace();
 JOptionPane.showMessageDialog(this, "Errorea gordetzean: " +
e.getMessage());
 }
}

```

```
}
```

# SaioaHastekoPanela.java

```
package ui;

import db.DB_Konexioa;
import model.*;

import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.SQLException;
import java.awt.Font;
import java.awt.Color;
import java.awt.Image;
import java.io.File;

/**
 * SaioaHastekoPanela klasea.
 * Aplikazioaren sarrera leihoa (Login).
 * Erabiltzaileak posta eta pasahitza sartuz saioa hasteko aukera ematen du.
 */
public class SaioaHastekoPanela extends JFrame {

 private static final long serialVersionUID = 1L;
 private JPanel edukiPanela;

 // JComboBox erabiltzen dugu emailak aukeratzeko
 private JComboBox<String> postaEremua;

 private JPasswordField pasahitzaEremua;

 // Testu etiketak
 private JLabel izenburuEtiketa;
 private JLabel postaEtiketa;
 private JLabel pasahitzaEtiketa;
 private JLabel irudiEtiketa;
 private JButton saioaHasiBotoia;

 /**
 * SaioaHastekoPanela eraikitzailea.
 * Leihoa osagaiak inizializatzen ditu.
 */
 public SaioaHastekoPanela() {
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 // Leihoa zabalagoa egin dugu elementuak ondo sartzeko
```

```

setBounds(100, 100, 900, 400);

edukiPanela = new JPanel();
edukiPanela.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(edukiPanela);
edukiPanela.setLayout(null);

// --- IRUDIA (EZKERRALDEAN) ---
irudiEtiketa = new JLabel("");
irudiEtiketa.setBounds(0, 0, 350, 360);
try {
 ImageIcon originalaIcon = null;
 // 1. Saiakera: Classpath bidez
 java.net.URL irundiURL =
SaioaHastekoPanela.class.getResource("/birtek1.jpeg");

 if (irundiURL != null) {
 originalaIcon = new ImageIcon(irundiURL);
 } else {
 // 2. Saiakera: Fitxategi-sisteman zuzenean
 // ALDAKETA: Irudiak 'irudiak' karpetan daude
 if (new File("irudiak/birtek1.jpeg").exists()) {
 originalaIcon = new ImageIcon("irudiak/birtek1.jpeg");
 }
 }
}

if (originalaIcon != null) {
 Image img = originalaIcon.getImage();
 Image imgBerria = img.getScaledInstance(350, 360,
Image.SCALE_SMOOTH);
 irudiEtiketa.setIcon(new ImageIcon(imgBerria));
} else {
 throw new Exception("Irudia ez da aurkitu");
}

} catch (Exception e) {
 irudiEtiketa.setText("Irudia ez da aurkitu / No image found");
 irudiEtiketa.setHorizontalAlignment(SwingConstants.CENTER);
 irudiEtiketa.setBorder(BorderFactory.createLineBorder(Color.GRAY));
 System.err.println("Errorea irudia kargatzean: Ziurtatu 'birtek1.jpeg'
src karpetan dagoela.");
}
edukiPanela.add(irudiEtiketa);

// --- LOGIN FORMULARIOA (ESKUINALDEAN) ---
int desplazamendua = 360;

// Titulua
izenburuEtiketa = new JLabel();
izenburuEtiketa.setFont(new Font("Tahoma", Font.BOLD, 18));
izenburuEtiketa.setBounds(desplazamendua + 50, 50, 300, 30);
edukiPanela.add(izenburuEtiketa);

// Emaila
postaEtiketa = new JLabel();
postaEtiketa.setBounds(desplazamendua + 50, 110, 100, 14);
edukiPanela.add(postaEtiketa);

```

```

postaEremua = new JComboBox();
postaEremua.setBounds(desplazamendua + 150, 107, 300, 25);
postaEremua.setEditable(false);
edukiPanela.add(postaEremua);

// Datuak kargatu ComboBox-ean
if (!java.beans.Beans.isDesignTime()) {
 postaKargatu();
}

// Pasahitza
pasahitzaEtiketa = new JLabel();
pasahitzaEtiketa.setBounds(desplazamendua + 50, 150, 100, 14);
edukiPanela.add(pasahitzaEtiketa);

pasahitzaEremua = new JPasswordField();
pasahitzaEremua.setBounds(desplazamendua + 150, 147, 300, 25);
edukiPanela.add(pasahitzaEremua);

// Sartu Botoia
saioaHasiBotoia = new JButton();
saioaHasiBotoia.setBackground(new Color(0, 128, 128));
saioaHasiBotoia.setForeground(new Color(0, 0, 0));
saioaHasiBotoia.addActionListener(e -> saioaHasi());
saioaHasiBotoia.setBounds(desplazamendua + 150, 200, 120, 30);
edukiPanela.add(saioaHasiBotoia);

// Testuak hasieratu
eguneratuTestuak();
}

// METODOA posta-helbideak eta sailak kargatzeko
/**
 * Posta helbideak eta sailak DBtik kargatu ComboBox-era.
 */
private void postaKargatu() {
 String galdera = "SELECT l.emaila, s.izenan AS saila_izena " +
 "FROM langileak l " +
 "JOIN langile_sailak s ON l.saila_id = s.id_saila " +
 "ORDER BY l.emaila ASC";

 Connection konexioa = DB_Konexioa.konektatu();
 if (konexioa == null) {
 postaEremua.addItem("Errorea: Ezin da konektatu DB-ra");
 return;
 }

 try (Statement stmt = konexioa.createStatement();
 ResultSet rs = stmt.executeQuery(galdera)) {

 postaEremua.removeAllItems();

 while (rs.next()) {
 String email = rs.getString("emaila");
 String saila = rs.getString("saila_izena");
 // "emaila (Saila)" formatua erakutsi
 }
 }
}

```

```

 postaEremua.addItem(email + " (" + saila + ")");
 }

 } catch (Exception ex) {
 ex.printStackTrace();
 postaEremua.addItem("Errorea datuak kargatzean");
 } finally {
 try {
 if (konexioa != null)
 konexioa.close();
 } catch (SQLException e) {
 e.printStackTrace();
 }
 }
 }

 /**
 * Testu etiketak eguneraatu.
 */
 private void eguneraatuTestuak() {
 setTitle("Saioa Hasi");
 izenburuEtiketa.setText("Birtek Kudeaketa");
 postaEtiketa.setText("Emaila:");
 pasahitzaEtiketa.setText("Pasahitza:");
 saioaHasiBotoia.setText("Sartu");
 }

 /**
 * Saioa hasteko prozesua.
 * Erabiltzailea eta pasahitza egiaztatzen ditu eta dagokion menua irekitzen
du.
 */
 private void saioaHasi() {
 String aukeratutakoa = (String) postaEremua.getSelectedItem();

 if (aukeratutakoa == null || aukeratutakoa.trim().isEmpty()) {
 JOptionPane.showMessageDialog(null, "Aukeratu erabiltzaile bat / Seleccione un usuario");
 return;
 }

 // "emaila (Saila)" formatutik emaila bakarrik atera
 String email = aukeratutakoa.split(" \\(")[0];

 String pasahitza = new String(pasahitzaEremua.getPassword());

 String galdera = "SELECT l.id_langilea, l.izena, l.abizena, l.saila_id,
s.izena AS saila_izena " +
 "FROM langileak l " +
 "JOIN langile_sailak s ON l.saila_id = s.id_saila " +
 "WHERE l.emaila = ? AND l.pasahitza = ?";

 Connection konexioa = DB_Konexioa.konektatu();
 if (konexioa == null) {
 JOptionPane.showMessageDialog(null, "Errorea: Ezin da konektatu datu-
basera (Driverra falta da?)", "Error",
 JOptionPane.ERROR_MESSAGE);
 }
 }
}

```

```

 return;
 }

 try (PreparedStatement pst = konexioa.prepareStatement(galdera)) {
 pst.setString(1, email);
 pst.setString(2, pasahitza);
 ResultSet rs = pst.executeQuery();

 if (rs.next()) {
 // Langilea objektua sortu saioaren datuekin
 Langilea l = new Langilea(
 rs.getInt("id_langilea"),
 rs.getString("izena"),
 rs.getString("abizena"),
 "", // NAN (ez dugu behar login-erako)
 null, // Jaiotza data
 0, // Herria ID
 "", // Helbidea
 "", // Posta kodea
 "", // Telefonoa
 email,
 "ES", // Hizkuntza lehenetsia
 pasahitza,
 "", // Salto UID
 null, // Alta data
 null, // Egeneratze data
 true, // Aktibo
 rs.getInt("saila_id"),
 "", // IBAN
 null // Kurrikuluma
);
 }

 String sailaIzena = rs.getString("saila_izena");

 irekiSailMenua(l, sailaIzena);
 dispose();
 } else {
 JOptionPane.showMessageDialog(null, "Login Error: Pasahitza okerra / Contraseña incorrecta", "Error",
 JOptionPane.ERROR_MESSAGE);
 }
} catch (Exception ex) {
 ex.printStackTrace();
 JOptionPane.showMessageDialog(null, "Errorea konexioan / Error de conexión");
} finally {
 try {
 if (konexioa != null)
 konexioa.close();
 } catch (SQLException e) {
 e.printStackTrace();
 }
}
}

/**
 * Sailaren araberako menua ireki.

```

```

*
* @param l Saioa hasi duen langilea.
* @param sailaIzena Sailaren izena.
*/
private void irekiSailMenua(Langilea l, String sailaIzena) {
 switch (l.getSailaId()) {
 case 1:
 new MenuZuzendaritza(l.getIdLangilea(), l.getIzena(),
l.getAbizena(), sailaIzena).setVisible(true);
 break;
 case 2:
 new MenuAdministrazioa(l).setVisible(true);
 break;
 case 3:
 new MenuSalmentak(l).setVisible(true);
 break;
 case 4:
 new MenuTeknikoa(l).setVisible(true);
 break;
 case 5:
 new MenuLogistika(l).setVisible(true);
 break;
 default:
 JOptionPane.showMessageDialog(null, "ID Ezezaguna / Desconocido");
 }
}
}

```

## SarreraBerriaDialog.java

```

package ui;

import db.DB_Konexioa;
import model.*;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.io.File;
import java.nio.file.Files;
import java.nio.file.StandardCopyOption;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

/**
 * SarreraBerriaDialog klasea.
 * Produktu sarrera berriak sartzeko elkarrizketa-leihoa.

```

```

*/
public class SarreraBerriaDialog extends JDialog {

 private BiltegiLangilea langilea;
 private JComboBox<MenuLogistika.Hornitzairelementua> hornitzaireHautatzailea;
 private JCheckBox hornitzaireBerriaAukera;
 private JTextField izenaBerriaTestua, postaBerriaTestua, ifzBerriaTestua;
 private JTextField produktuIzenaTestua, markaTestua, kantitateTestua,
deskribapenaTestua, irudiaUrlTestua,
 egoeraOharraTestua;
 private JComboBox<MenuLogistika.KategoriaElementua> kategoriaHautatzailea;
 private JComboBox<String> motaHautatzailea;
 private JComboBox<MenuLogistika.BiltegiElementua> biltegiHautatzaileaSarrera;
 private JTable lerroBerriTaula;
 private DefaultTableModel lerroBerriEredua;
 private JFileChooser fitxategiHautatzailea;

 public SarreraBerriaDialog(Frame parent, BiltegiLangilea langilea) {
 super(parent, "Sarrera Berria", true);
 this.langilea = langilea;
 pantailaPrestatu();
 sarreraHautatzaileakKargatu();
 }

 private void pantailaPrestatu() {
 setSize(900, 700);
 setLocationRelativeTo(getParent());
 setLayout(new BorderLayout());

 JPanel formularioPanela = new JPanel(new BorderLayout());
 formularioPanela.setBorder(BorderFactory.createTitledBorder("Sarrera eta
Produktu Berriaren Datuak"));

 // Hornitzaire atala
 JPanel hornitzairePanela = new JPanel(new GridLayout(2, 1));
 JPanel hornAukeratuPanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
 hornitzaireHautatzailea = new JComboBox<>();
 hornitzaireBerriaAukera = new JCheckBox("Hornitzaire Berria Sortu?");
 hornitzaireBerriaAukera.addActionListener(e -> hornitzaireModuaAldatu());
 hornAukeratuPanela.add(new JLabel("Hornitzairea Aukeratu: "));
 hornAukeratuPanela.add(hornitzaireHautatzailea);
 hornAukeratuPanela.add(hornitzaireBerriaAukera);

 JPanel hornBerriaPanela = new JPanel(new FlowLayout(FlowLayout.LEFT));
 izenaBerriaTestua = new JTextField(15);
 postaBerriaTestua = new JTextField(15);
 ifzBerriaTestua = new JTextField(10);
 hornBerriaPanela.add(new JLabel("Izena:"));
 hornBerriaPanela.add(izenaBerriaTestua);
 hornBerriaPanela.add(new JLabel("Emaila:"));
 hornBerriaPanela.add(postaBerriaTestua);
 hornBerriaPanela.add(new JLabel("IFZ:"));
 hornBerriaPanela.add(ifzBerriaTestua);
 hornitzaireBerriaGaitu(false);

 hornitzairePanela.add(hornAukeratuPanela);
 hornitzairePanela.add(hornBerriaPanela);

 formularioPanela.add(hornitzairePanela, BorderLayout.NORTH);
 formularioPanela.add(hornBerriaPanela, BorderLayout.SOUTH);
 add(formularioPanela);
 }
}

```

```
formularioPanela.add(hornitzalePanela, BorderLayout.NORTH);

// Produktu eremuak
JPanel formPanela = new JPanel(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(5, 5, 5, 5);
gbc.fill = GridBagConstraints.HORIZONTAL;

produktuIzenaTestua = new JTextField(15);
markaTestua = new JTextField(15);
kategoriaHautatzailea = new JComboBox<>();
motaHautatzailea = new JComboBox<>(new String[] { "Eramangarria", "Mahai-gainekoa", "Mugikorra", "Tableta",
 "Zerbitzaria", "Pantaila", "Softwarea" });
biltegiHautatzaileaSarrera = new JComboBox<>();
kantitateTestua = new JTextField(5);
deskribapenaTestua = new JTextField(30);
irudiaUrlTestua = new JTextField(20);
egoeraOharraTestua = new JTextField(20);

gbc.gridx = 0;
gbc.gridy = 0;
formPanela.add(new JLabel("Prod. Izena:"), gbc);
gbc.gridx = 1;
formPanela.add(produktuIzenaTestua, gbc);
gbc.gridx = 2;
formPanela.add(new JLabel("Marka:"), gbc);
gbc.gridx = 3;
formPanela.add(markaTestua, gbc);

gbc.gridx = 0;
gbc.gridy = 1;
formPanela.add(new JLabel("Kategoria:"), gbc);
gbc.gridx = 1;
formPanela.add(kategoriaHautatzailea, gbc);
gbc.gridx = 2;
formPanela.add(new JLabel("Mota:"), gbc);
gbc.gridx = 3;
formPanela.add(motaHautatzailea, gbc);

gbc.gridx = 0;
gbc.gridy = 2;
formPanela.add(new JLabel("Biltegia:"), gbc);
gbc.gridx = 1;
formPanela.add(biltegiHautatzaileaSarrera, gbc);
gbc.gridx = 2;
formPanela.add(new JLabel("Kantitatea:"), gbc);
gbc.gridx = 3;
formPanela.add(kantitateTestua, gbc);

gbc.gridx = 0;
gbc.gridy = 3;
formPanela.add(new JLabel("Deskribapena:"), gbc);
gbc.gridx = 1;
gbc.gridwidth = 3;
formPanela.add(deskribapenaTestua, gbc);
gbc.gridwidth = 1;
```

```

gbc.gridx = 0;
gbc.gridy = 4;
formPanela.add(new JLabel("Irudia URL:"), gbc);
 JPanel irudiPanela = new JPanel(new BorderLayout());
irudiPanela.add(irudiaUrlTestua, BorderLayout.CENTER);
 JButton igoBotoia = new JButton("Igo");
igoBotoia.addActionListener(e -> igoIrudia());
irudiPanela.add(igoBotoia, BorderLayout.EAST);
gbc.gridx = 1;
formPanela.add(irudiPanela, gbc);
gbc.gridx = 2;
formPanela.add(new JLabel("Oharra:"), gbc);
gbc.gridx = 3;
formPanela.add(egoeraOharraTestua, gbc);

 JButton gehituBotoia = new JButton("Gehitu Zerrendara +");
gehituBotoia.addActionListener(e -> gehituLerroaTaulara());
 JPanel botoiHustuPanela = new JPanel();
botoiHustuPanela.add(gehituBotoia);

 JPanel erdikoFormPanela = new JPanel(new BorderLayout());
erdikoFormPanela.add(formPanela, BorderLayout.CENTER);
erdikoFormPanela.add(botoiHustuPanela, BorderLayout.SOUTH);

 formularioPanela.add(erdikoFormPanela, BorderLayout.CENTER);
add(formularioPanela, BorderLayout.NORTH);

 String[] zutabeIzenak = { "Izena", "Marka", "Kategoria", "Mota",
"Biltegia", "Kantitatea", "Deskribapena",
 "Irudia", "Oharra" };
lerroBerriEredua = new DefaultTableModel(zutabeIzenak, 0);
lerroBerriTaula = new JTable(lerroBerriEredua);
add(new JScrollPane(lerroBerriTaula), BorderLayout.CENTER);

 JButton gordeBotoia = new JButton("GORDE SARRERA ETA SORTU PRODUKTUAK");
gordeBotoia.setBackground(new Color(0, 128, 0));
gordeBotoia.setForeground(Color.WHITE);
gordeBotoia.setFont(new Font("Arial", Font.BOLD, 14));
gordeBotoia.addActionListener(e -> gordeSarrera0soa());
add(gordeBotoia, BorderLayout.SOUTH);
}

private void hornitzaireModuaAldatu() {
 boolean berriaDa = hornitzaireBerriaAukera.isSelected();
 hornitzaireBerriaGaitu(berriaDa);
 hornitzaireHautatzailea.setEnabled(!berriaDa);
}

private void hornitzaireBerriaGaitu(boolean gaitu) {
 izenaBerriaTestua.setEnabled(gaitu);
 postaBerriaTestua.setEnabled(gaitu);
 ifzBerriaTestua.setEnabled(gaitu);
}

private void sarreraHautatzaileakKargatu() {
 try (Connection konexioa = DB_Konexioa.konektatu()) {

```

```

 Statement sententzia = konexioa.createStatement();
 ResultSet rsH = sententzia.executeQuery("SELECT id_hornitzairea,
izenasoziala FROM hornitzaireak");
 while (rsH.next())
 hornitzaireHautatzailea
 .addItem(new
MenuLogistika.HornitzaireElementua(rsH.getInt(1), rsH.getString(2)));
 ResultSet rsK = sententzia.executeQuery("SELECT id_kategoria, izena
FROM produktukategoriak");
 while (rsK.next())
 kategoriaHautatzailea.addItem(new
MenuLogistika.KategoriaElementua(rsK.getInt(1), rsK.getString(2)));
 ResultSet rsB = sententzia.executeQuery("SELECT id_biltegia, izena FROM
biltegiak");
 while (rsB.next())
 biltegiHautatzaileaSarrera.addItem(new
MenuLogistika.BiltegiElementua(rsB.getInt(1), rsB.getString(2)));
 } catch (Exception e) {
 e.printStackTrace();
 }
}

private void gehituLerroaTaulara() {
 String izena = produktuIzenaTestua.getText();
 String marka = markaTestua.getText();
 MenuLogistika.KategoriaElementua kat = (MenuLogistika.KategoriaElementua)
kategoriaHautatzailea
 .getSelectedItem();
 String mota = (String) motaHautatzailea.getSelectedItem();
 MenuLogistika.BiltegiElementua bilt = (MenuLogistika.BiltegiElementua)
biltegiHautatzaileaSarrera
 .getSelectedItem();
 String kantiStr = kantitateTestua.getText();
 if (izena.isEmpty() || marka.isEmpty() || kantiStr.isEmpty() || kat == null
|| bilt == null) {
 JOptionPane.showMessageDialog(this, "Mesedez, bete produktuaren eremu
guztiak.");
 return;
 }
 try {
 int kanti = Integer.parseInt(kantiStr);
 lerroBerriEredua.addRow(new Object[] { izena, marka, kat, mota, bilt,
kanti, deskribapenaTestua.getText(),
 irudiaUrlTestua.getText(), egoeraOharraTestua.getText() });
 produktuIzenaTestua.setText("");
 markaTestua.setText("");
 kantitateTestua.setText("");
 deskribapenaTestua.setText("");
 irudiaUrlTestua.setText("");
 egoeraOharraTestua.setText("");
 } catch (NumberFormatException ex) {
 JOptionPane.showMessageDialog(this, "Kantitateak zenbakia izan behar
du.");
 }
}

private void igoIrudia() {

```

```

 if (fitxategiHautatzailea == null) {
 fitxategiHautatzailea = new JFileChooser();
 fitxategiHautatzailea.setFileFilter(
 new javax.swing.filechooser.FileNameExtensionFilter("Irudiak",
"jpg", "jpeg", "png"));
 }
 if (fitxategiHautatzailea.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
 File selectedFile = fitxategiHautatzailea.getSelectedFile();
 try {
 String baseDir = "src/birtek_interfaze_grafikoa/irudiak/";
 File destFile = new File(new File(baseDir),
selectedFile.getName().replaceAll("\\\\s+", "_"));
 if (!destFile.getParentFile().exists())
 destFile.getParentFile().mkdirs();
 Files.copy(selectedFile.toPath(), destFile.toPath(),
StandardCopyOption.REPLACE_EXISTING);
 irudiaUrlTestua.setText("irudiak/" + destFile.getName());
 } catch (Exception ex) {
 JOptionPane.showMessageDialog(this, "Errorea irudia igotzean: " +
ex.getMessage());
 }
 }
}

private void gordeSarreraOsoa() {
 if (lerroBerriEredua.getRowCount() == 0) {
 JOptionPane.showMessageDialog(this, "Ez dago produkturik zerrendan.");
 return;
 }
 try {
 int hornitzaleaId = -1;
 if (hornitzaleBerriaAukera.isSelected()) {
 hornitzaleaId =
langilea.hornitzaleBerriaSortu(izenaBerriaTestua.getText(),
ifzBerriaTestua.getText(),
postaBerriaTestua.getText());
 } else {
 MenuLogistika.HornitzaleElementua item =
(MenuLogistika.HornitzaleElementua) hornitzaleHautatzailea
 .getSelectedItem();
 if (item != null)
 hornitzaleaId = item.id;
 }
 if (hornitzaleaId == -1)
 return;
 }

 List<Produktua> produktuList = new ArrayList<>();
 List<SarreraLerroa> lerroList = new ArrayList<>();
 for (int i = 0; i < lerroBerriEredua.getRowCount(); i++) {
 MenuLogistika.KategoriaElementua kat =
(MenuLogistika.KategoriaElementua) lerroBerriEredua.getValueAt(i,
2);
 MenuLogistika.BiltegiElementua bilt =
(MenuLogistika.BiltegiElementua) lerroBerriEredua.getValueAt(i,
4);
 int kanti = (Integer) lerroBerriEredua.getValueAt(i, 5);

```

# TaulaModelatzzailea.java

```
package ui;

import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.Vector;
import javax.swing.table.DefaultTableModel;

/**
 * TaulaModelatzalea klasea.
 * ResultSet batetik JTable baterako DefaultTableModel sortzeko utilitatea.
 */
public class TaulaModelatzalea {

 /**
 * ResultSet datuekin TableModel bat eraikitzen du.
 *
 * @param emaitza SQL kontsultaren emaitza (ResultSet).
 * @return Datuak dituen DefaultTableModel objektua.
 * @throws SQLException SQL errorea gertatzen bada.
 */
 public static DefaultTableModel ereduEraiki(ResultSet emaitza) throws
SQLException {
 ResultSetMetaData metaDatuak = emaitza.getMetaData();
```

```
// 1. Zutabeen izenak lortu
Vector<String> zutabeIzenak = new Vector<String>();
int zutabeKopurua = metaDatuak.getColumnCount();
for (int zutabea = 1; zutabea <= zutabeKopurua; zutabea++) {
 zutabeIzenak.add(metaDatuak.getColumnName(zutabea));
}

// 2. Errenkaden datuak lortu
Vector<Vector<Object>> datuak = new Vector<Vector<Object>>();
while (emaitza.next()) {
 Vector<Object> lerroa = new Vector<Object>();
 for (int zutabeIndizea = 1; zutabeIndizea <= zutabeKopurua;
zutabeIndizea++) {
 lerroa.add(emaitza.getObject(zutabeIndizea));
 }
 datuak.add(lerroa);
}

return new DefaultTableModel(datuak, zutabeIzenak) {
 @Override
 public Class<?> getColumnClass(int columnIndex) {
 if (getRowCount() > 0 && getValueAt(0, columnIndex) != null) {
 return getValueAt(0, columnIndex).getClass();
 }
 return Object.class;
 }
};
}
```