Project Εξαμήνου
Σύστημα Μισθοδοσίας Πανεπιστημίου Κρήτης

Φροντιστήριο
Βοηθός Μαθήματος: Μιχαήλ Γαλλιάκης
(email: csdp1413@csd.uoc.gr)
HY-360 Αρχεία και Βάσεις Δεδομένων
10/12/2025

Η δομή του σημερινού φροντιστηρίου

- Μέρος Α: Ανάλυση Απαιτήσεων & Παραδοτέα (Τι φτιάχνουμε;)
- Μέρος Β: Setup Εργαλείων & Περιβάλλοντος (Πώς το φτιάχνουμε;)
- Μέρος Γ: Live Demo & Implementation Guide (Παράδειγμα στην πράξη)
- Q&A - Ερωτήσεις

Τι καλούμαστε να φτιάξουμε;

- Ο Ρόλος μας: IT Support Τμήματος Μισθοδοσίας.
- Ο Χρήστης (User): Διαχειριστής στο τμήμα μισθοδοσίας.
- Ο Στόχος: Ανάπτυξη εφαρμογής (συστήματος) για:
  - Διαχείριση Προσλήψεων & Συμβάσεων.
  - Αυτόματο Υπολογισμό Μισθοδοσίας.
  - Εξαγωγή Αναφορών & Ιστορικού.

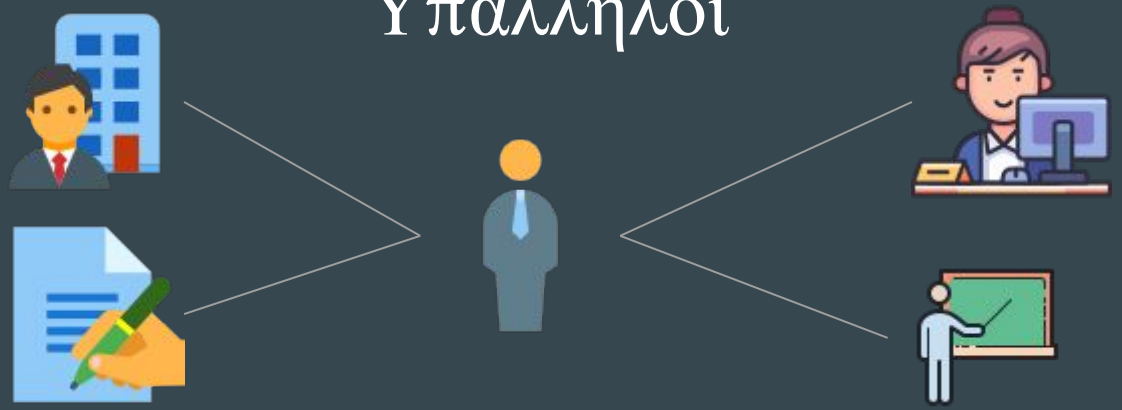
Αρχιτεκτονική

Data Layer (DB): Η αποθήκευση των δεδομένων (πχ MariaDB/MySQL).

Business Logic (Java): Οι υπολογισμοί και οι κανόνες (Java Code).

Presentation (UI): Αυτό που βλέπει ο χρήστης (πχ Java Swing / Servlets).

Υπάλληλοι



- Δεδομένα Υπαλλήλων: περιλαμβάνουν το όνομα κάθε υπαλλήλου, την οικογενειακή του κατάσταση (άγαμος / έγγαμος, αριθμός παιδιών και ηλικίες), την κατηγορία προσωπικού στην οποία ανήκει, το τμήμα στο οποίο εργάζεται, την ημερομηνία έναρξης εργασίας του στο Πανεπιστήμιο, διεύθυνση κατοικίας, τηλέφωνο, τον αριθμό τραπεζικού λογαριασμού και το όνομα της τράπεζας.

Μισθοδοσία

Βασικό μισθό + 15% για κάθε χρόνο

Μισθό σύμβασης

Βασικό μισθό + 15% για κάθε χρόνο
+ επίδομα έρευνας

Μισθό σύμβασης
+ επίδομα βιβλιοθήκης

+ 5% του μισθού, για σύζυγο και κάθε παιδί

Λοιπές οντότητες

- Δεδομένα Καταβολών Μισθοδοσίας: περιλαμβάνουν το όνομα του υπαλλήλου, την ημερομηνία και το ποσό της πληρωμής. Υποθέστε ότι η καταβολή της μισθοδοσίας γίνεται την τελευταία μέρα κάθε μήνα.

❖ Μπορείτε να συμπεριλάβετε επιπλέον πληροφορία αν το κρίνετε απαραίτητο. Σε αυτή την περίπτωση θα πρέπει να τεκμηριωσετε την επιλογή σας.

Βασικές διαδικασίες

- Πρόσληψη νέου μόνιμου υπαλλήλου. Η πρόσληψη έχει πάντα ισχύ από την πρώτη μέρα του μήνα και δεν μπορεί να γίνεται αναδρομικά.
- Σύναψη σύμβασης με νέο υπάλληλο. Υποθέστε ότι η ανανέωση συμβάσεων θα γίνεται πάντα με τη σύναψη νέας σύμβασης. Μια νέα σύμβαση ξεκινά πάντα την πρώτη μέρα του μήνα και δεν μπορεί να γίνεται αναδρομικά.
- Αλλαγή στοιχείων υπαλλήλων. Πρέπει να υποστηρίζεται η αλλαγή οποιουδήποτε στοιχείου αφορά τον υπάλληλο (πχ., αλλαγή διεύθυνσης, αλλαγή οικογενειακής κατάστασης κλπ).
- Μεταβολή βασικών μισθών και επιδομάτων. Η διαδικασία αυτή μεταβάλει τα ποσά των βασικών μισθών (για κάποιες ή ολες τις κατηγορίες) ή των επιδομάτων. Δεν πρέπει να επιτρέπεται μείωση μισθών ή επιδομάτων.
- Απόλυση ή συνταξιοδότηση μόνιμου υπαλλήλου. Θεωρείστε ότι απολύσεις ή συνταξιοδοτήσεις γίνονται μόνο την τελευταία μέρα του μήνα και επομένως πρέπει να καταβληθεί όλο το ποσό της μισθοδοσίας.

Βασικές διαδικασίες

- Καταβολή μισθοδοσίας. Γίνεται η πληρωμή όλων των υπαλλήλων με κατάθεση του ποσού στον τραπεζικό τους λογαριασμό. Πρέπει να δημιουργείται μια κατάσταση μισθοδοσίας η οποία θα δείχνει αναλυτικά την καταβολή μισθών και επιδομάτων ανά κατηγορία υπαλλήλου.
- Ερωτήσεις: Το σύστημα πρέπει να υποστηρίζει αυθαίρετες ερωτήσεις προς τη βάση δεδομένων μέσω κατάλληλης διεπαφής χρήσης. Θα πρέπει συγχρόνως να υποστηρίζει και τις ακόλουθες προκαθορισμένες ερωτήσεις
    - Κατάσταση μισθοδοσίας ανά κατηγορία προσωπικού
    - Μεγιστος, ελάχιστος και μέσος μισθός ανά κατηγορία προσωπικού
    - Μέση αύξηση μισθών και επιδομάτων ανά χρονική περίοδο
    - Στοιχεία και μισθοδοσία συγκεκριμένου υπαλλήλου
    - Συνολικό ύψος μισθοδοσίας ανά κατηγορία προσωπικού

Χρήση Όψεων (Views)

- Τι είναι: "Εικονικοί πίνακες" (Αποθηκευμένα πολύπλοκα SQL queries).
- Όφελος: Απλοποιεί τον κώδικα της Java (SELECT * FROM View αντί για πχ 5 JOINs).

Φάση Ι (Σχεδιασμός)

- Ένα πλήρες διάγραμμα οντοτήτων-σχέσεων για την εταιρία
- Τα γνωρίσματα (όνομα, τύπος) όλων των οντοτήτων και σχέσεων
- Τα πρωτεύοντα κλειδιά
- Επεξηγήσεις για τα μη προφανή γνωρίσματα και τις μη-προφανείς σχέσεις
- Περιορισμούς πληθικότητας
- Τη μετάφραση του μοντέλου σας στο σχεσιακό μοντέλο
- Τις εντολές της γλώσσας ορισμού δεδομένων για τις σχέσεις που προκύπτουν
- Περιορισμούς ακεραιότητας και συναρτησιακές εξαρτήσεις
- Καθορισμό κλειδιών των σχέσεων βάσει των συναρτησιακών εξαρτήσεων
- Μετατροπή του μοντέλου σε τρίτη κανονική μορφή με διατήρηση των συναρτησιακών εξαρτήσεων και χωρίς απώλεια πληροφορίας.
- Περιγραφή των ερωτήσεων προς τη βάση δεδομένων με SQL
- Περιγραφή σε ψευδοκώδικα των διαδικασιών.

Φάση ΙΙ (Υλοποίηση)

- Ενδεικτικά αποτελέσματα από την εκτέλεση των διαδικασιών
- Ένα εγχειρίδιο χρήσης της εφαρμογής.
- Περιγραφή των περιορισμών της υλοποίησής σας και των δυνατοτήτων βελτίωσής του.
- Τον κώδικα των προγραμμάτων που θα υλοποιούν τις διαδικασίες που καθορίστηκαν στην προηγούμενη φάση.

Το Tech Stack που προτείνεται για την εργασία

- Database: MariaDB ή MySQL (Relational DB).
- Backend Logic: Java (με JDBC για σύνδεση).
- Frontend:
  - Java Swing (Desktop App).
  - Java Servlets (Web App).
- Σημαντικό: Το μάθημα είναι "Βάσεις Δεδομένων". Βαθμολογείστε κυρίως για τη σωστή Βάση, τα Queries και τις διαδικασίες, όχι για το πόσο όμορφα είναι τα user interfaces.

Υλοποίηση & Ομάδες

- Ομάδες: 3 Ατόμων.
- Δήλωση ομάδων στο Email: hy360@csd.uoc.gr (Ονόματα/ΑΜ μέχρι 15/12).
- Deadline: 23 Ιανουαρίου 2026.
- Εξέταση προς το τέλος της εξεταστικής: Υποχρεωτική παρουσία όλων.

Τέλος πρώτου μέρους

Ερωτήσεις;

Δεύτερο μέρος

Για τη βάση δεδομένων

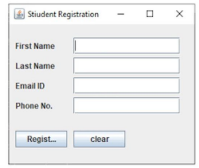
Για την εφαρμογή

Java

Servlets & Jsp

IntelliJ IDEA
JETBRAINS IDE

JAVA Swing



Apache NetBeans

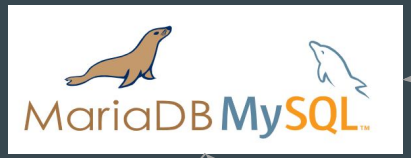
mysql/mysql-connector-j
MySQL Connector/J

Maven

Προτεινόμενη Αρχιτεκτονική Συστήματος

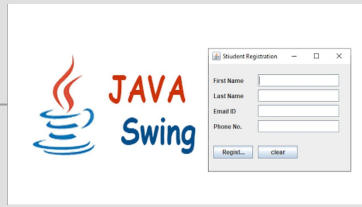
Data Layer (DB).

Business Logic (Java) UIs (πχ Java Swing / Servlets).



mysql/**mysql-connector-j**

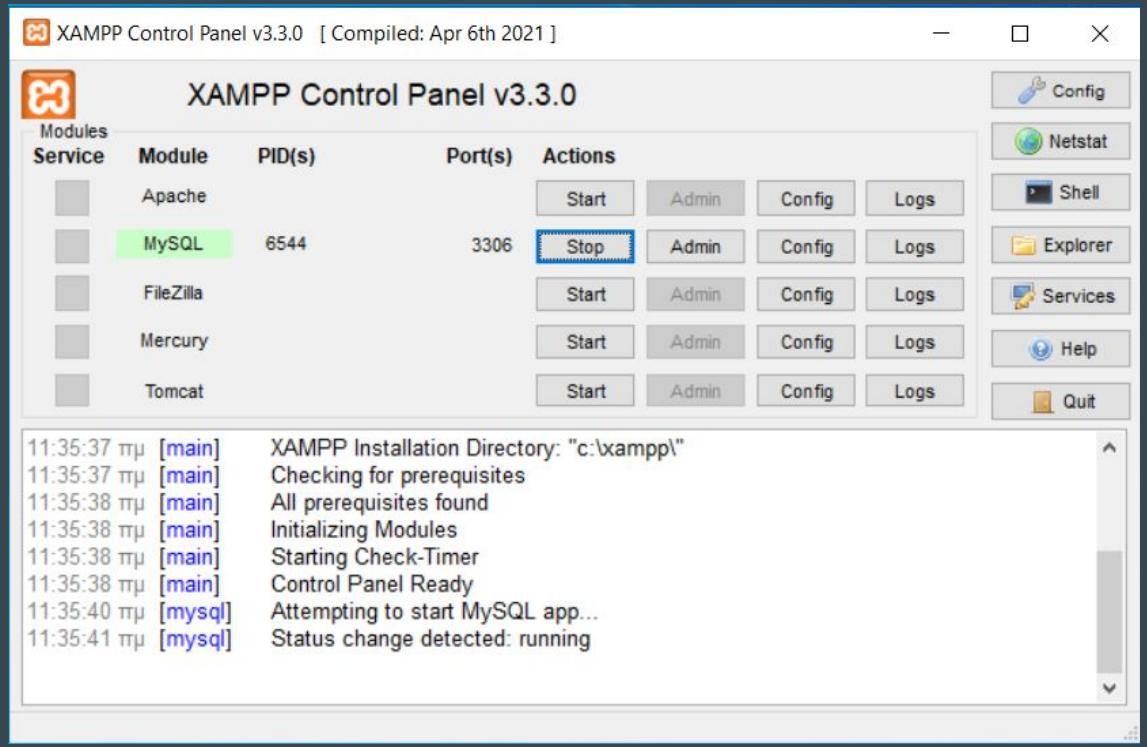
MySQL Connector/J



Οδηγίες εγκατάστασης και links

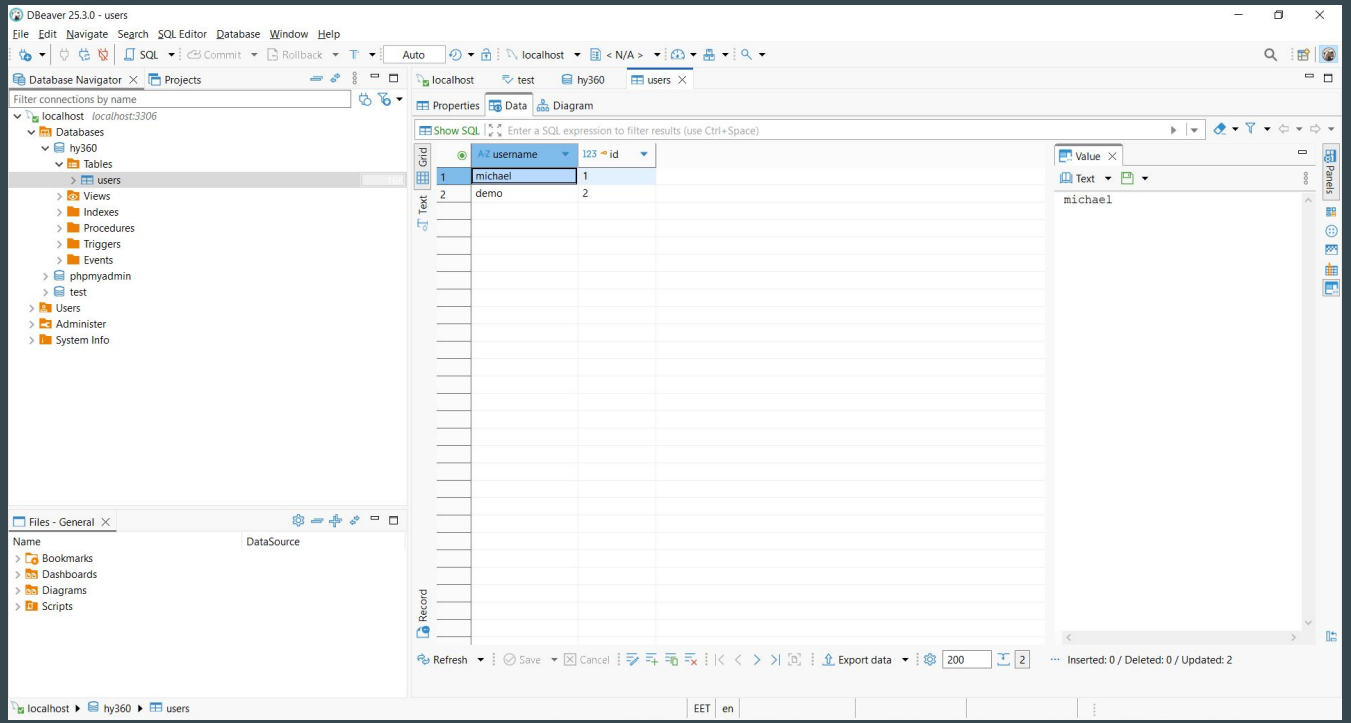
XAMPP	https://www.apachefriends.org/download.html
DBeaver	https://dbeaver.io/download/
Netbeans	https://netbeans.apache.org/front/main/download/
mysql-connector-j	https://repo1.maven.org/maven2/com/mysql/mysql-connector-j/ https://repo1.maven.org/maven2/com/mysql/mysql-connector-j/8.1.0/mysql-connector-j-8.1.0.jar

XAMPP

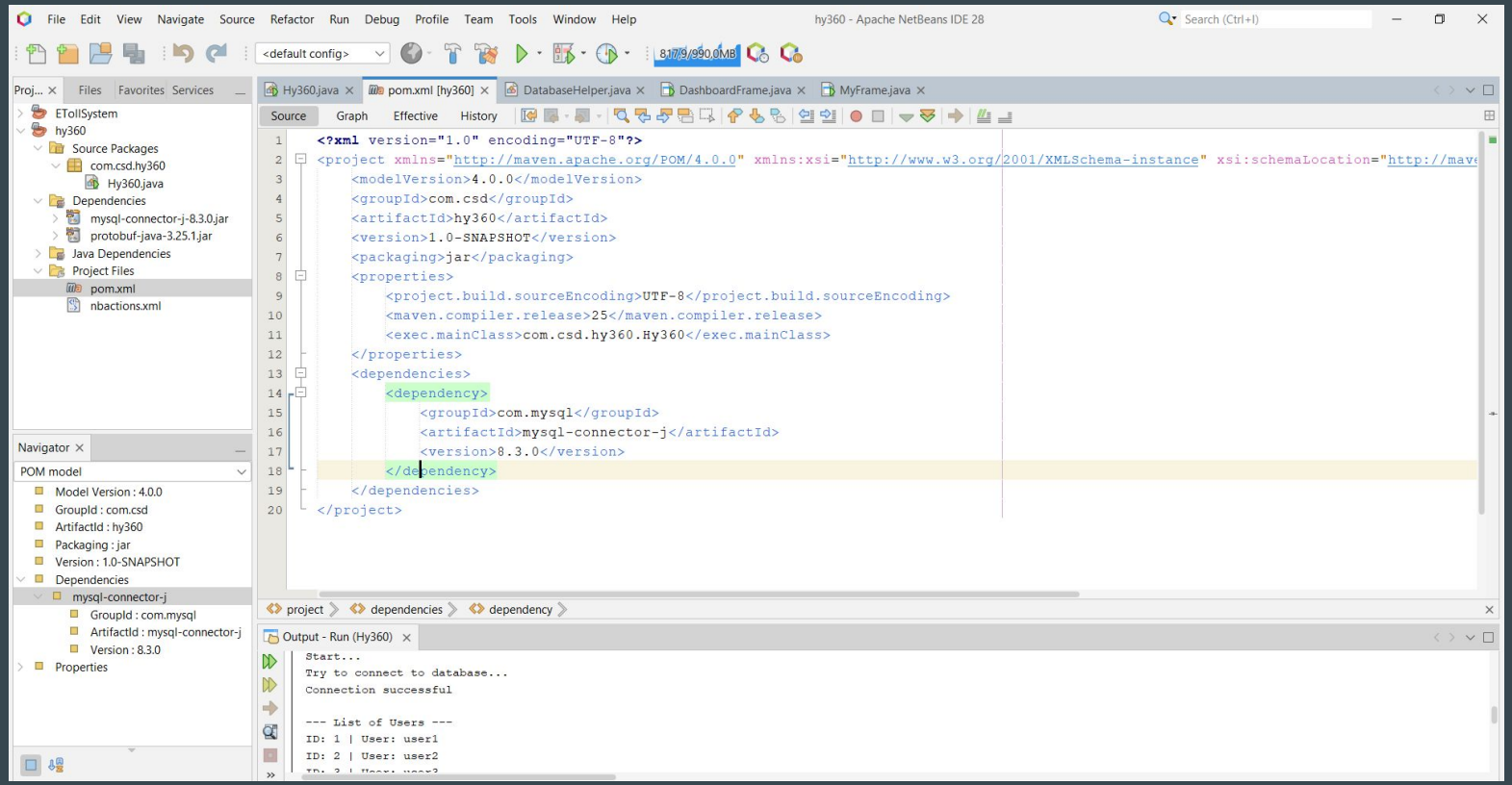


Απλά πατάς Start στο MySQL

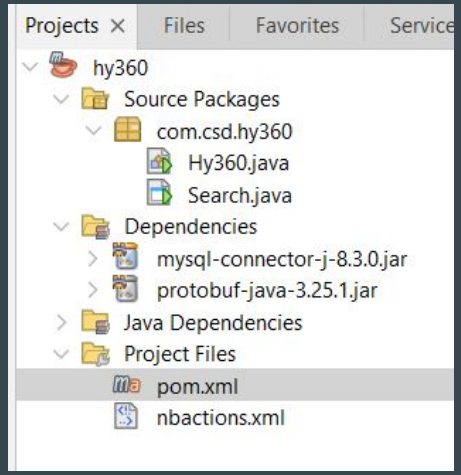
DBeaver

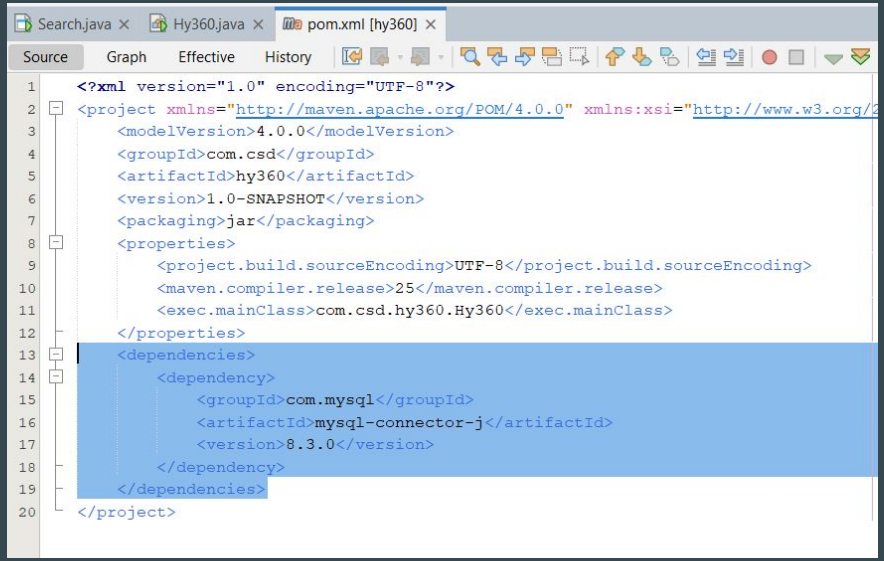

DBeaver 25.3.0 - users
File Edit Navigate Search SQL Editor Database Window Help
Database Navigator
Projects
Filter connections by name
localhost localhost:3306
Databases
hy360
Tables
users
Views
Indexes
Procedures
Triggers
Events
phpmyadmin
test
Users
Administer
System Info
Files - General
Name
DataSource
Bookmarks
Dashboards
Diagrams
Scripts
localhost
test
hy360
users
Properties
Data
Diagram
Show SQL
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
Record
username
id
1 michael 1
2 demo 2
Value
michael
Refresh
Save
Cancel
Export data
200
Inserted: 0 / Deleted: 0 / Updated: 2
EET
en

Netbeans


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
hy360 - Apache NetBeans IDE 28
Search (Ctrl+I)
<default config>
Proj... Files Favorites Services
ETollSystem
hy360
Source Packages
com.csd.hy360
Hy360.java
Dependencies
mysql-connector-j-8.3.0.jar
protobuf-java-3.25.1.jar
Java Dependencies
Project Files
pom.xml
nbactions.xml
Navigator
POM model
Model Version : 4.0.0
GroupId : com.csd
ArtifactId : hy360
Packaging : jar
Version : 1.0-SNAPSHOT
Dependencies
mysql-connector-j
GroupId : com.mysql
ArtifactId : mysql-connector-j
Version : 8.3.0
Properties
Hy360.java
pom.xml [hy360]
DatabaseHelper.java
DashboardFrame.java
MyFrame.java
Source Graph Effective History
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://mave
<modelVersion>4.0.0</modelVersion>
<groupId>com.csd</groupId>
<artifactId>hy360</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.release>25</maven.compiler.release>
<exec.mainClass>com.csd.hy360.Hy360</exec.mainClass>
</properties>
<dependencies>
<dependency>
<groupId>com.mysql</groupId>
<artifactId>mysql-connector-j</artifactId>
<version>8.3.0</version>
</dependency>
</dependencies>
</project>
project dependencies dependency
Output - Run (Hy360)
Start...
Try to connect to database...
Connection successful
--- List of Users ---
ID: 1 | User: user1
ID: 2 | User: user2

Εγκατάσταση του mysql-connector-j

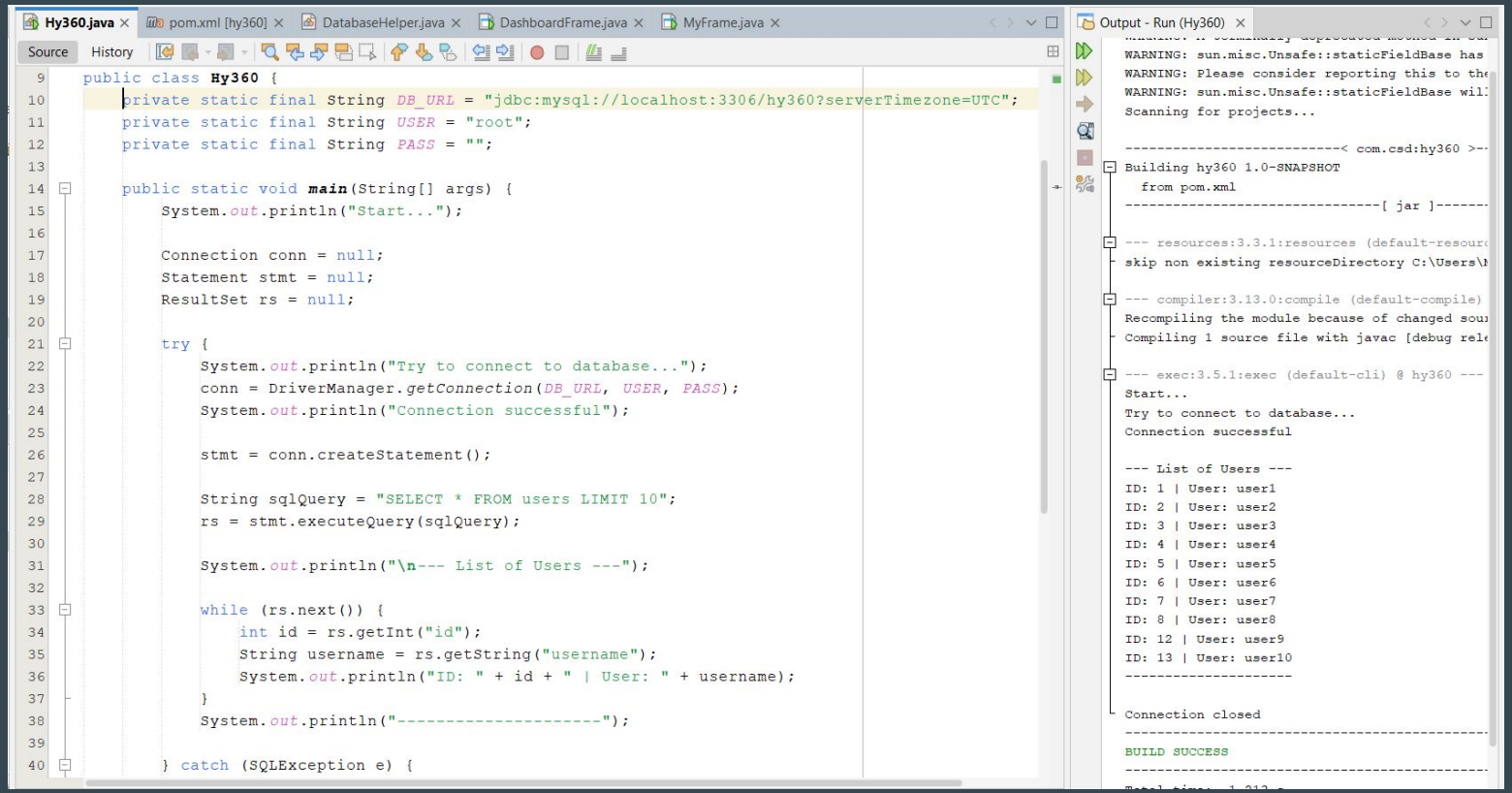




Αν το project είναι Maven (Προτείνεται), τότε προσθήκη το δίπλα στα dependencies, αλλιώς κατεβάζουμε το mysql-connector-j.jar αρχείο και το κάνουμε επικόλληση στο φάκελο Dependencies

```
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>8.3.0</version>
</dependency>
```

Σύνδεση στη βάση και το πρώτο query από Java.



Ας τα δούμε λίγο στη πράξη

Τέλος δεύτερου μέρους

Ερωτήσεις;

Τρίτο μέρος

Σύστημα Ηλεκτρονικών Διοδίων (E-Toll)

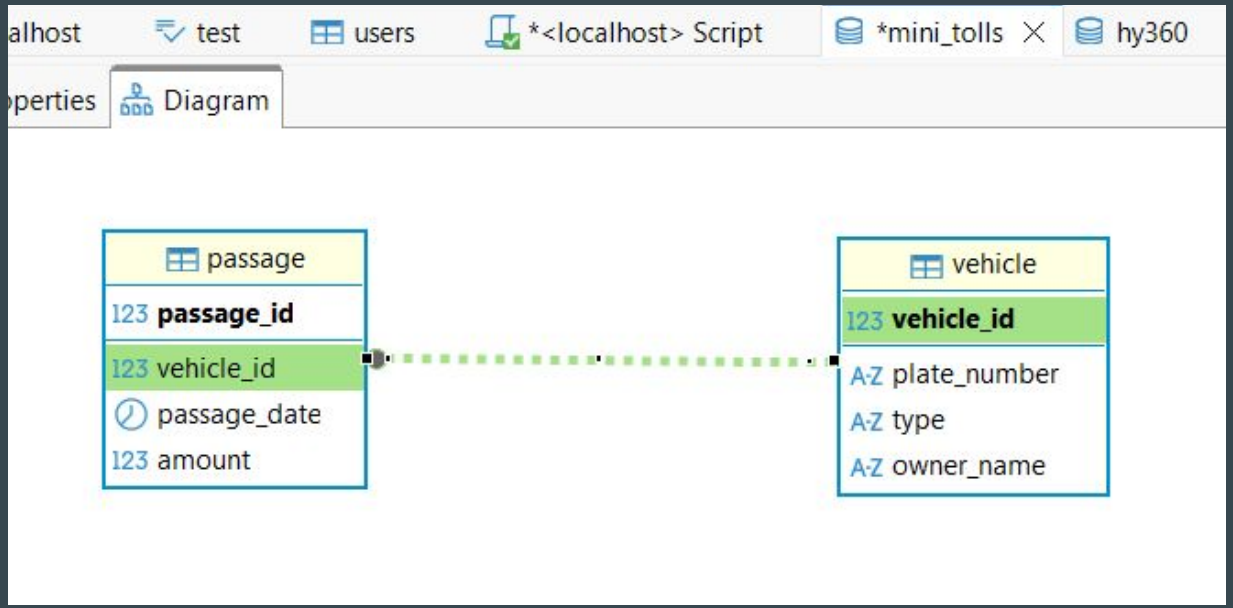
- Το Σενάριο:

  Η εταιρεία λειτουργίας αυτοκινητοδρόμου θέλει να καταγράφει αυτόματα τις διελεύσεις οχημάτων.
- Τα Δεδομένα:
  - Οχήματα (Vehicles): Έχουν Πινακίδα και Τύπο.
  - Διελεύσεις (Passages): Καταγράφουμε ποιο όχημα πέρασε, πότε και το κόστος χρέωσης.
- Οι Κανόνες (Business Logic):
  - Τα Επιβατικά (CAR) χρεώνονται 2.80€.
  - Τα Φορτηγά (TRUCK) χρεώνονται 6.00€.
- Ζητούμενο: Μια λειτουργία όπου ο υπάλληλος εισάγει την πινακίδα και το σύστημα καταχωρεί τη διέλευση με το σωστό ποσό.

Σχεδιασμός Λύσης

- Μοντελοποίηση:
  Δεν φτιάχνουμε πίνακα CARS και πίνακα TRUCKS. Φτιάχνουμε πίνακα VEHICLE με πεδία-σημαίες (ENUM).
- Ιστορικότητα:
  Κάθε διέλευση είναι ένα νέο γεγονός. Χρειαζόμαστε πίνακα PASSAGE συνδεδεμένο με το όχημα (Foreign Key).
- Λογική Εφαρμογής:
  Η Java θα ρωτήσει "Τι όχημα είναι;" και θα αποφασίσει το ποσό πριν κάνει το Insert.

SQL Schema (DBeaver)



Σημείωση: Εδώ κάνουμε μία απλή εγγραφή. Τεχνικά, η βάση δεν χρειάζεται transaction [conn.setAutoCommit(false)] για ένα μόνο Insert. Θα δούλευε και χωρίς αυτό.

Υλοποίηση σε Java

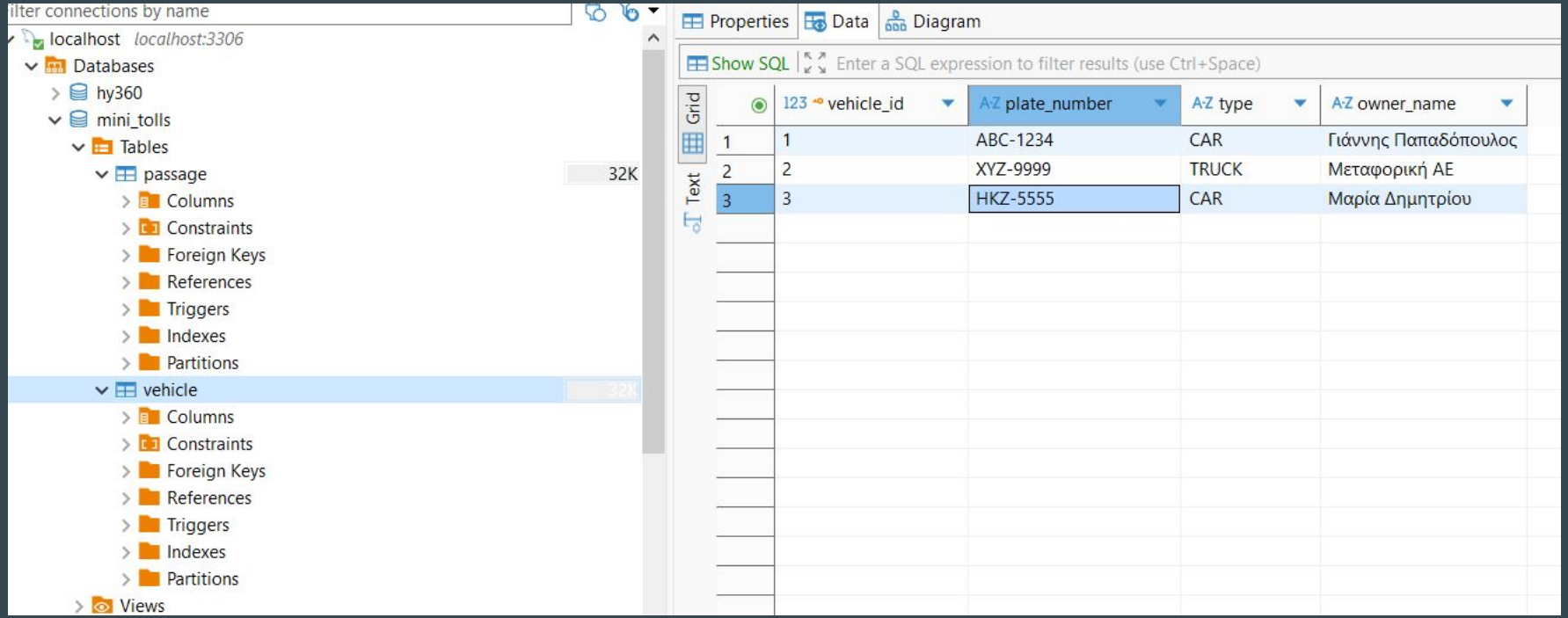
```
private static final String URL = "jdbc:mysql://localhost:3306/mini_tolls?serverTimezone=UTC";
private static final String USER = "root";
private static final String PASS = "";

private static Connection getConnection() throws SQLException {
    return DriverManager.getConnection(URL, USER, PASS);
}
```

```
try (Connection conn = getConnection()) {
    conn.setAutoCommit(false);

    try {
        // Βήμα 1: Βρες το όχημα και τον τύπο του
        String sqlCheck = "SELECT vehicle_id, type, owner_name FROM Vehicle WHERE plate_number = ?";
        PreparedStatement pstmtCheck = conn.prepareStatement(sqlCheck);
        pstmtCheck.setString(1, plateNumber);

        ResultSet rs = pstmtCheck.executeQuery();

        if (rs.next()) {
            int vId = rs.getInt("vehicle_id");
            String type = rs.getString("type");
            String owner = rs.getString("owner_name");
            double cost = 0.0;

            // Βήμα 2: Business Logic (Java Logic)
            if ("CAR".equalsIgnoreCase(type)) {
                cost = 2.80;
            } else if ("TRUCK".equalsIgnoreCase(type)) {
                cost = 6.00;
            }

            // Βήμα 3: Καταχώρηση στη βάση
```

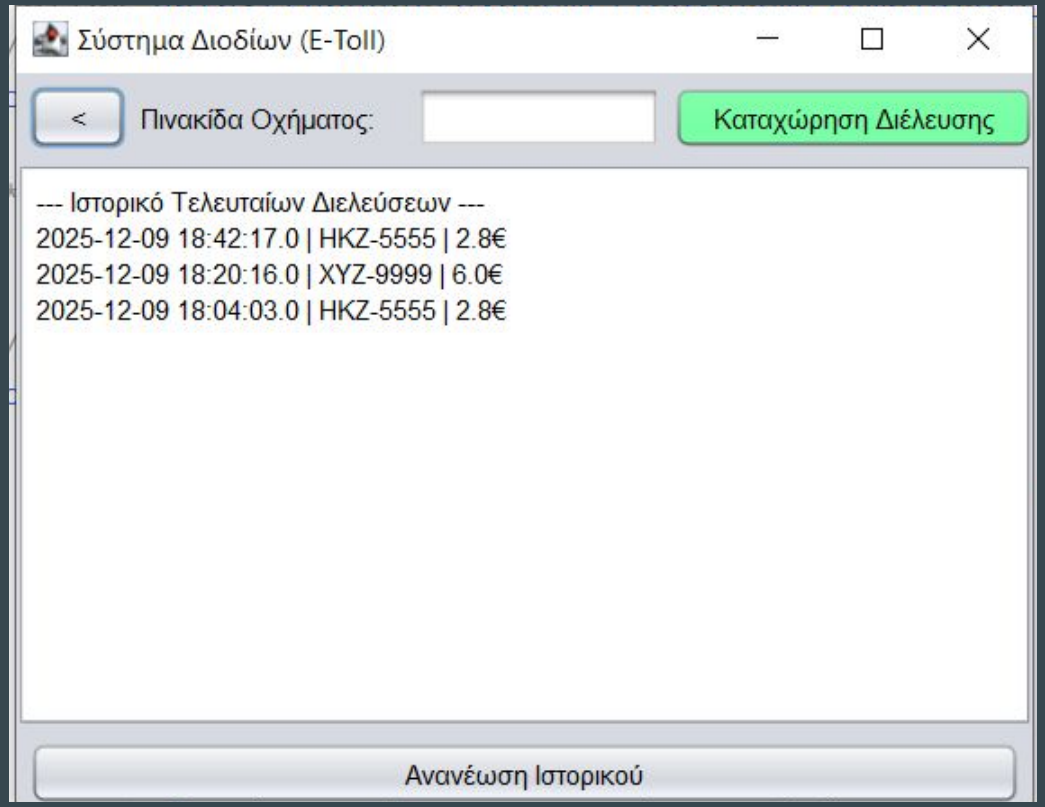
Υλοποίηση σε Java

```
                cost = 6.00;
            }

            // Βήμα 3: Καταχώρηση στη βάση
            String sqlInsert = "INSERT INTO Passage (vehicle_id, amount, passage_date) VALUES (?, ?, NOW())";
            PreparedStatement pstmtInsert = conn.prepareStatement(sqlInsert);
            pstmtInsert.setInt(1, vId);
            pstmtInsert.setDouble(2, cost);
            pstmtInsert.executeUpdate();

            // Βήμα 4: COMMIT - Οριστικοποίηση της συναλλαγής
            // Μόνο τώρα τα δεδομένα γράφονται πραγματικά στη βάση
            conn.commit();

            resultMessage = "Επιτυχία. Όχημα: " + type + " (" + owner + ")\n" +
                            "Χρέωση: " + cost + "€";
        } else {
            // Αν δεν βρεθεί το όχημα, πετάμε Exception (θα πάει στο catch για rollback)
            throw new Exception("Το όχημα με πινακίδα " + plateNumber + " δεν βρέθηκε.");
        }
    } catch (Exception e) {
        // Βήμα 5: ROLLBACK - Ακύρωση σε περίπτωση σφάλματος (είτε SQL error, είτε Logic error)
        // Γυρνάμε τη βάση στην κατάσταση που ήταν πριν ξεκινήσουμε.
        if (conn != null) {
            System.out.println("Προέκυψε σφάλμα, γίνεται Rollback...");
            conn.rollback();
        }
        throw e;
    }
}
```

Περιεχόμενα πίνακα vehicle



Mini Java εφαρμογή



Ας δούμε live ένα demo της υλοποίησης

Χρήσιμες Συμβουλές για την Υλοποίηση

- Διαχείριση Ημερομηνιών (Dates): Η Java χρησιμοποιεί συνήθως LocalDate ενώ η Βάση DATE. Δεν είναι το ίδιο. Για να τα αποθηκεύσετε σωστά, χρησιμοποιήστε τη μετατροπή: java.sql.Date.valueOf(myLocalDate).
- Transactions (Για τη Μισθοδοσία): Όταν κάνετε μαζικές εγγραφές (π.χ. υπολογισμός μισθοδοσίας για όλους), προτείνεται να χρησιμοποιήστε Transactions (conn.setAutoCommit(false)). Έτσι εξασφαλίζετε ότι ή θα πληρωθούν όλοι ή κανένας (αν συμβεί σφάλμα), προστατεύοντας τη βάση από "μισά" δεδομένα.
- Οργάνωση Κώδικα: Μην γράφετε SQL queries μέσα στον κώδικα των κουμπιών (UI). Φτιάξτε μια ξεχωριστή κλάση (π.χ. DatabaseHelper) που θα διαχειρίζεται τη βάση. Το κουμπί πρέπει απλά να καλεί έτοιμες μεθόδους.
- Προσοχή στο SQL Injection: Μην ενώνετε Strings με το σύμβολο + μέσα στα queries σας (π.χ. "SELECT... WHERE name=" + name). Είναι κενό ασφαλείας. Χρησιμοποιείτε PreparedStatement με τα ερωτηματικά (?) ως placeholders.

Τέλος τρίτου μέρους (Φροντιστηρίου)

Ερωτήσεις;

ΗΥ - 360 Αρχεία και Βάσεις Δεδομένων
Project Εξαμήνου
Σύστημα Μισθοδοσίας Πανεπιστημίου Κρήτης
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης
(10/12/25)