Contact List Application

Design Document

Varsha Samararatne

March 20th, 2020

Table of Contents

1. Introduction 2
1.1 Purpose 2
1.2 Scope 2
1.3 Assumptions 2
1.3 Glossary 3
2. System Architecture 3
2.1 Database Design 3
2.2 Backend and Graphical-User-Interface 5

1. Introduction

1.1 Purpose

The purpose of this document is to comprehensively outline the design of the application and the user interface. It describes the system architecture including design decisions and assumptions.

1.2 Scope

This design document involves the creation of a database host application that interfaces with a backend database implementing a Contact List application. The host application contains a native graphical user interface (GUI). The GUI application interfaces with the Contact List MySQL database via JDBC API. All interface with the database is done via the GUI which is programmed with Java and JavaFX. Provided baseline data to initialize the database are populated to database tables via a python script. Database tables store first, middle, and last names, address, phone number and date details.

The Contact List application GUI allows users to search contacts given any combination of Name components, address component (s) and phone number components. All searches are done via a single search field. Search results include a list of hits that display full name. Upon selecting a full name from the search results, detailed information about the contact is displayed along with an option to modify or delete the entry.

The application GUI also allows users to add new contacts via a button that brings up a new contact entry form. This form allows entry of all data including first, middle and last names, a variable number of address, phone numbers and dates.

1.3 Assumptions

The following is a list of identified assumptions for the Contact List application project:

Assumptions:

- Application will be used in the Windows operating system.
- Active database connectivity is required to use the application.
- JavaFX SDK version 11 or above and JDK version 11 or above must be installed to use the application.
- Application security module will be built in a future phase

1.3 Glossary

Term	Definition
JDBC	Java Database Connectivity is an application programming interface for the Java programming language which defines how a client may access a database.
JavaFX	Java library for creating and delivering desktop applications and rich Internet applications.
IntelliJ IDEA	Open source integrated development environment
Maven	Build automation tool used primarily for Java projects
SDK	Software Development Kit. Collection of software development tools in one installable package.
JDK	Java Development Kit. A development environment for building applications and applets using the Java programming language.
Scene Builder	Visual, drag-and-drop layout tool for designing JavaFX application user interfaces
FXML	XML-based language designed to build the user interface for JavaFX applications.

2. System Architecture

2.1 Database Design

The baseline data provided was normalized onto a schema and tables using a python script. The script extracts information from the csv file and creates SQL INSERT statements that can be run to populate tables. As per the first normal form, every attribute in a relation is a single valued attribute. To satisfy second normal form, partial dependencies were removed and placed into separate relations.

The data model involves four entities (Contact, Address, Phone, Dates) in one to many relationships. Based on this ER model, a relational database was constructed utilizing MySQL as the management system. Initial database creation and population was done using the admin tool MySQL Workbench. The attributes for each entity and their respective data types are presented in the EER diagram in Figure 1. Every record has a primary key which is an integer that is auto generated using the SQL AUTO_INCREMENT option. The primary keys for the tables CONTACT, ADDRESS, PHONE and DATES are Contact_id, Address_id, Phone_id and Date_id respectively. Contact_id in ADDRESS, PHONE and DATES is a foreign key that references Contact_id in CONTACT.

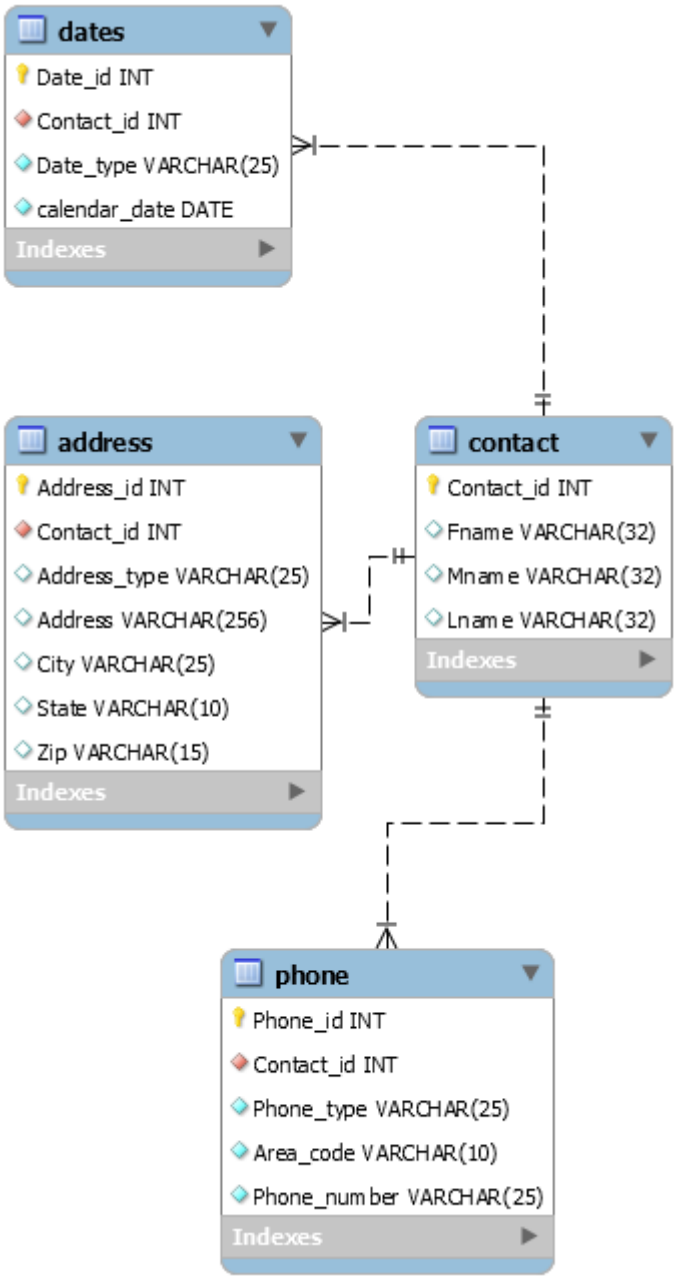

Figure 1. EER diagram for the Contact schema

2.2 Backend and Graphical-User-Interface

The GUI application interacts with the backend MySQL database through a JDBC. This was developed with Java and the user interface was programmed with JavaFX. Separate Java classes were created for Contact, Address, Phone and Date entities which supports data encapsulation. JavaFX Scene Builder which is a visual layout tool was used to help design the user interfaces.

Scene Builder was configured within the IntelliJ IDEA which was used for all the development. The resulting FXML files were then combined with the Java project by binding the user interface to the application's logic. For this purpose, controller java classes were defined. These classes initialize the user interface elements of the application main window, add new contact window and edit contact window. Exception handling is included in all Java classes.

Maven was used as the build automation tool. JavaFX dependencies were added to the project using Maven. These dependencies were defined in the pom.xml file and they were downloaded into the local repository from the central or remote repository via Maven. Additionally, with Maven a Jar file for the project was built. Running this Jar file outside of the integrated development environment requires specifying the path to where the JavaFX SDK is installed. Therefore, a Windows executable file was generated which specifies the path and runs the Jar file. The Contact List application can be started by clicking this executable file.

The details of the Java classes created are included below.

contact.java

Include instance variables for first name, middle name, last name and contact ID. Provides getter and setter methods for these instance variables

Address.java

Includes instance variables for address type, street address, city, state and zip. Provides getter and setter methods for these instance variables.

Phone.java

Includes instance variables for phone type, area code and phone number. Provides getter and setter methods for these instance variables

Date.java

Include instances variables for date type and date. Provides getter and setter methods for these instance variables.

Datasource.java

Contains methods to open and close database connection. Includes methods to insert, delete, modify, query and retrieve data from the MySQL database.

Controller.java

Contains methods to initialize UI elements in the main Contact List application window and in the add new contact application window.

EditContactController.java

Contains methods to initialize UI elements in the edit contact window of the application.

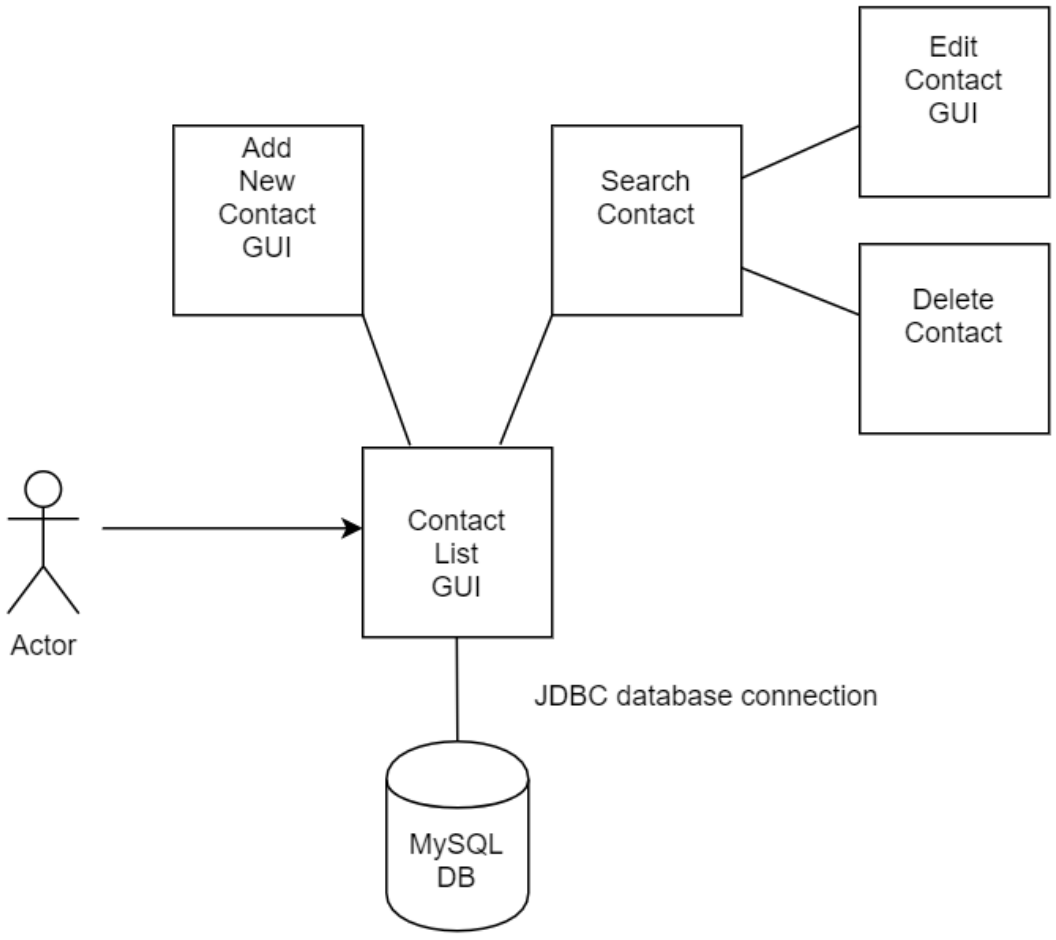

Figure 2 System Architecture of the Contact List application.