Java Persistence API

Kenneth Rodriguez

Java Persistence API

This project is a demonstration of the Java Persistence API using Spring Boot. Using the JPA and MySQL, we will create a movie database with information about movies' titles, release date, budget, revenue, runtime and public rating, as well as a Java application that will interact with the database. Using the JPA will allow us to simplify the process of managing a database with Java, removing the need to write manual SQL queries and preventing potential SQL Injection attacks.

Database

For this project, we will create a database with some of the most popular movies in TMDB. We can use this publicly available dataset to get the data we need to construct our database.

Creating the database

With the following SQL script, we can create a MySQL Database and Table to store our data.

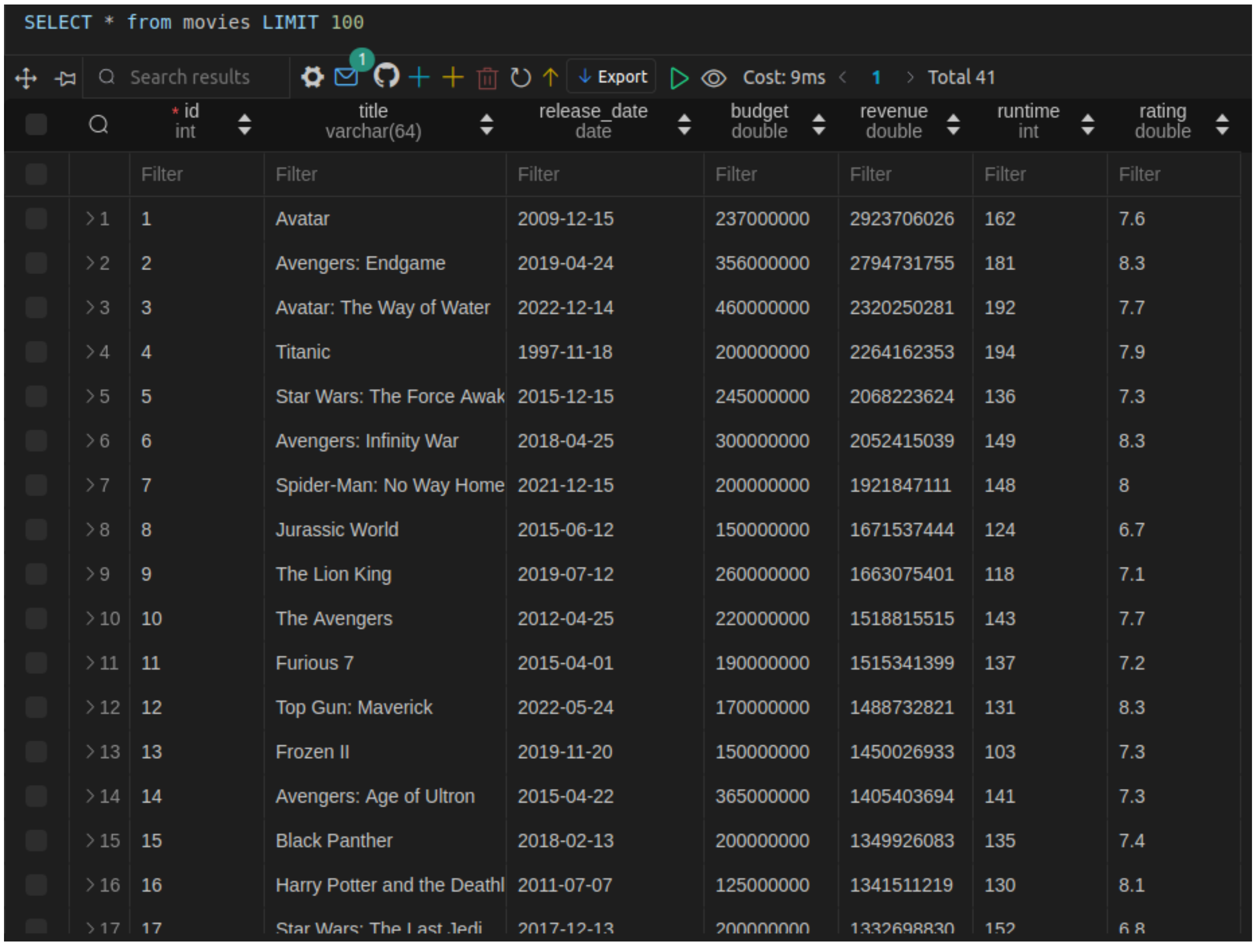
```
CREATE DATABASE IF NOT EXISTS MovieDB;

USE MovieDB;

DROP TABLE IF EXISTS movies;

CREATE TABLE movies (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(64) DEFAULT NULL,
    releaseDate DATE DEFAULT NULL,
    originalLanguage VARCHAR(64) DEFAULT NULL,
    budget DOUBLE DEFAULT 0,
    revenue DOUBLE DEFAULT 0,
    runtime INT DEFAULT 0,
    rating DOUBLE
) AUTO_INCREMENT=1;
```

Now we can insert values into the table and populate the database.

Finally, we can perform a simple query to see that our database has been set up correctly.

```
SELECT * FROM movies;
```

If everything is fine, we should get a result like this:



```
SELECT * from movies LIMIT 100
```

	* id int	title varchar(64)	release_date date	budget double	revenue double	runtime int	rating double
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Avatar	2009-12-15	237000000	2923706026	162	7.6
2	2	Avengers: Endgame	2019-04-24	356000000	2794731755	181	8.3
3	3	Avatar: The Way of Water	2022-12-14	460000000	2320250281	192	7.7
4	4	Titanic	1997-11-18	200000000	2264162353	194	7.9
5	5	Star Wars: The Force Awak	2015-12-15	245000000	2068223624	136	7.3
6	6	Avengers: Infinity War	2018-04-25	300000000	2052415039	149	8.3
7	7	Spider-Man: No Way Home	2021-12-15	200000000	1921847111	148	8
8	8	Jurassic World	2015-06-12	150000000	1671537444	124	6.7
9	9	The Lion King	2019-07-12	260000000	1663075401	118	7.1
10	10	The Avengers	2012-04-25	220000000	1518815515	143	7.7
11	11	Furious 7	2015-04-01	190000000	1515341399	137	7.2
12	12	Top Gun: Maverick	2022-05-24	170000000	1488732821	131	8.3
13	13	Frozen II	2019-11-20	150000000	1450026933	103	7.3
14	14	Avengers: Age of Ultron	2015-04-22	365000000	1405403694	141	7.3
15	15	Black Panther	2018-02-13	200000000	1349926083	135	7.4
16	16	Harry Potter and the Deathl	2011-07-07	125000000	1341511219	130	8.1
17	17	Star Wars: The Last Jedi	2017-12-13	200000000	1332698830	152	6.8

Maven Dependencies

This project requires 2 dependencies to work properly: **Spring Data JPA** and **MySQL Connector/J**.

Spring Data JPA

This dependency helps implement a JPA application. It will essentially be the backbone of the project, as it will allow integration with our database.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
    <version>3.2.9</version>
</dependency>
```

MySQL Connector/J

This dependency provides the driver necessary for connecting our application with a MySQL database. This driver may be exchanged for another to allow connections with other databases.

```
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>8.4.0</version>
</dependency>
```

Project Resources

For the Spring application to work correctly, we must provide it with login access to the database. Within the `resources` directory in the project source files, we must include a file named `application.properties`, which should contain the necessary credentials, such as the database URL, username, and password.

The properties file should resemble something like this:

```
spring.datasource.url=jdbc:mysql://localhost:3306/MovieDB
# spring.datasource.username=USERNAME
# spring.datasource.password=PASSWORD
```

Note: For the security reasons I have not included my database login information, and an example file has been added instead.

Classes

The application consists of 4 classes and interfaces: `MovieApp`, `Movie`, `MovieDAO`, and `MovieDAOImpl`.

MovieApp

`MovieApp` is our Spring application, as it will execute the desired queries. In particular, the `commandLineRunner` method with the `@Bean` annotation is what will determine what method will be executed when Spring Boot is initialized.

```java
package com.kennethrdzg.jpa;

// Imports ...

@SpringBootApplication
public class MovieApp {
    public static void main(String[] args) {
        SpringApplication.run(MovieApp.class, args);
    }

    @Bean
    public CommandLineRunner commandLineRunner(MovieDAO movieDAO){
        return runner -> {
            createMovie(movieDAO);
            readMovie(movieDAO);
            queryAllMovies(movieDAO);
            queryMoviesByTitle(movieDAO);
            updateMovie(movieDAO);
            deleteMovie(movieDAO);
            deleteAllMovies(movieDAO);
        };
    }
    // Methods...
}
```

Movie

This class represents an entity of Movie, and is associated with the database table `"movie"`. Each of this class's attributes, like `id` or `title` is associated with one of the tables columns, which will help map each value in the table to an instance of this class.

The parameters of this class are:

- **id**: An `Integer`, and the primary key of the table.
- **title**: a `String`
- **releaseDate**: a `LocalDate` object.
- **budget**: a `Double`
- **revenue**: a `Double`
- **runtime**: an `Integer` representing the movie's duration.
- **rating**: a `Double`, the average rating of the film by audiences.

```
@Entity
@Table(name = "movies")
public class Movie {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    // Other Attributes

    // Constructors
    public Movie(){
    }

    public Movie(String title, LocalDate releaseDate, double budget, double
revenue, int runtime, double rating){
        this.title = title;
        this.releaseDate = releaseDate;
        this.budget = budget;
        this.revenue = revenue;
        this.runtime = runtime;
        this.rating = rating;
    }

    // Getters and Setters...
}
```

MovieDAO

This interface represents a `Data Access Object`, which allows it to interact with the database to execute queries. The interface does not contain method implementations, only definitions. Instead, the class `MovieDAOImpl` will contain the implementations and define the behavior of each method.

```
public interface MovieDAO {
    void save(Movie movie);
    Movie findById(Integer id);
    List<Movie> findAll();
    List<Movie> findByTitle(String title);
    void update(Movie movie);
    void delete(Integer id);
    int deleteAll();
}
```

MovieDAOImpl

This class implements the `MovieDAO` interface, and defines the behavior for each of its methods.

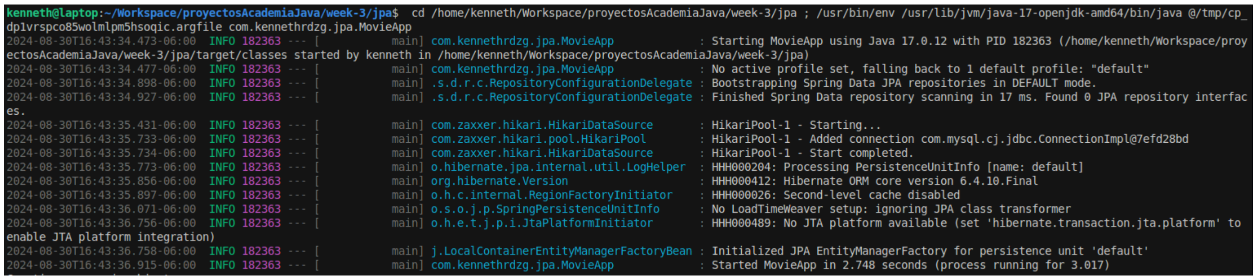
Some of this class's methods are transactional, which means that changes to the database are executed and saved only if everything was executed correctly. This prevents the unwanted loss of data (like updating an entry incorrectly).

The only attribute of this class is `entityManager`, which is used to provide persistence to entity instances in the program.

```
@Repository
public class MovieDAOImpl implements MovieDAO {
    private EntityManager entityManager;

    @Autowired
    public MovieDAOImpl(EntityManager entityManager){
        this.entityManager = entityManager;
    }

    // Other method implementations
}
```

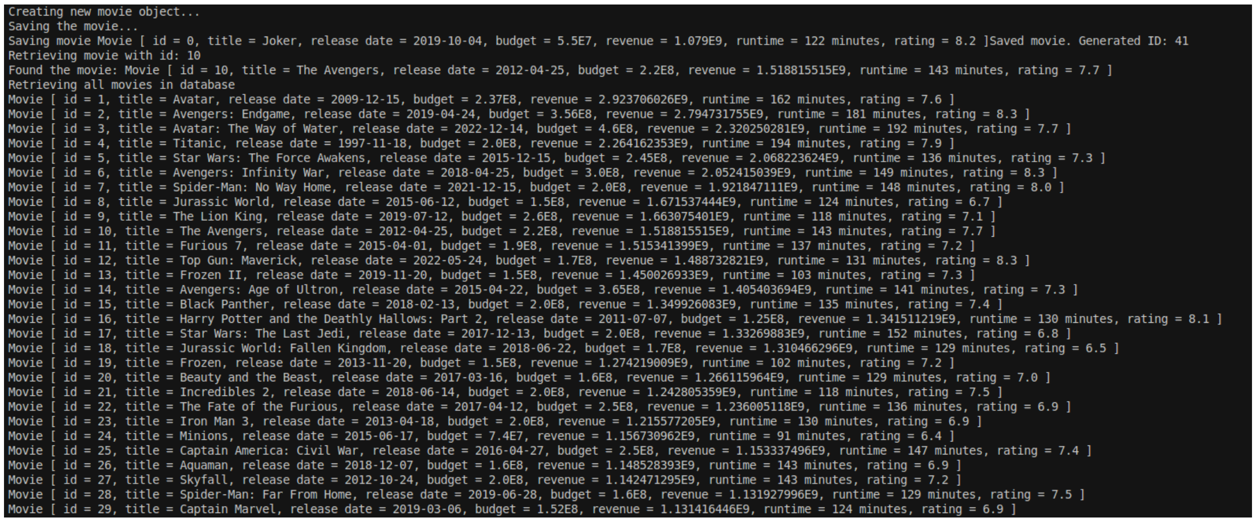
Result

Once we execute the application, we can observe how Spring Boot is initialized.



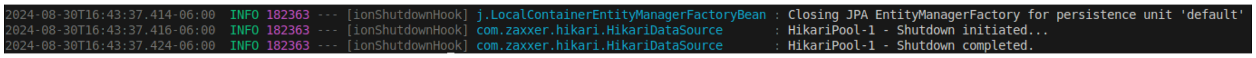
```
kenneth@laptop:~/Workspace/proyectosAcademiaJava/week-3/jpa$  cd /home/kenneth/Workspace/proyectosAcademiaJava/week-3/jpa ; /usr/bin/env /usr/lib/jvm/java-17-openjdk-amd64/bin/java @/tmp/cp_
dp1vrspco85wolmlpm5hsoqic.argfile com.kennethrdzg.jpa.MovieApp
2024-08-30T16:43:34.473-06:00  INFO 182363 --- [           main] com.kennethrdzg.jpa.MovieApp             : Starting MovieApp using Java 17.0.12 with PID 182363 (/home/kenneth/Workspace/proy
ectosAcademiaJava/week-3/jpa/target/classes started by kenneth in /home/kenneth/Workspace/proyectosAcademiaJava/week-3/jpa)
2024-08-30T16:43:34.477-06:00  INFO 182363 --- [           main] com.kennethrdzg.jpa.MovieApp             : No active profile set, falling back to 1 default profile: "default"
2024-08-30T16:43:34.898-06:00  INFO 182363 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-08-30T16:43:34.927-06:00  INFO 182363 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 17 ms. Found 0 JPA repository interfac
es.
2024-08-30T16:43:35.431-06:00  INFO 182363 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-08-30T16:43:35.733-06:00  INFO 182363 --- [           main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@7efd28bd
2024-08-30T16:43:35.734-06:00  INFO 182363 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2024-08-30T16:43:35.773-06:00  INFO 182363 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-08-30T16:43:35.856-06:00  INFO 182363 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.4.10.Final
2024-08-30T16:43:35.897-06:00  INFO 182363 --- [           main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2024-08-30T16:43:36.071-06:00  INFO 182363 --- [           main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class transformer
2024-08-30T16:43:36.756-06:00  INFO 182363 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to
enable JTA platform integration)
2024-08-30T16:43:36.758-06:00  INFO 182363 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-08-30T16:43:36.915-06:00  INFO 182363 --- [           main] com.kennethrdzg.jpa.MovieApp             : Started MovieApp in 2.748 seconds (process running for 3.017)
```

After initialization is finished, all methods called by the `commandLineRunner` method in `MovieApp` will be executed, resulting in an output similar to this:



```
Creating new movie object...
Saving the movie...
Saving movie Movie [ id = 0, title = Joker, release date = 2019-10-04, budget = 5.5E7, revenue = 1.079E9, runtime = 122 minutes, rating = 8.2 ]Saved movie. Generated ID: 41
Retrieving movie with id: 10
Found the movie: Movie [ id = 10, title = The Avengers, release date = 2012-04-25, budget = 2.2E8, revenue = 1.518815515E9, runtime = 143 minutes, rating = 7.7 ]
Retrieving all movies in database
Movie [ id = 1, title = Avatar, release date = 2009-12-15, budget = 2.37E8, revenue = 2.923706026E9, runtime = 162 minutes, rating = 7.6 ]
Movie [ id = 2, title = Avengers: Endgame, release date = 2019-04-24, budget = 3.56E8, revenue = 2.794731755E9, runtime = 181 minutes, rating = 8.3 ]
Movie [ id = 3, title = Avatar: The Way of Water, release date = 2022-12-14, budget = 4.6E8, revenue = 2.320250281E9, runtime = 192 minutes, rating = 7.7 ]
Movie [ id = 4, title = Titanic, release date = 1997-11-18, budget = 2.0E8, revenue = 2.264162353E9, runtime = 194 minutes, rating = 7.9 ]
Movie [ id = 5, title = Star Wars: The Force Awakens, release date = 2015-12-15, budget = 2.45E8, revenue = 2.068223624E9, runtime = 136 minutes, rating = 7.3 ]
Movie [ id = 6, title = Avengers: Infinity War, release date = 2018-04-25, budget = 3.0E8, revenue = 2.052415039E9, runtime = 149 minutes, rating = 8.3 ]
Movie [ id = 7, title = Spider-Man: No Way Home, release date = 2021-12-15, budget = 2.0E8, revenue = 1.921847111E9, runtime = 148 minutes, rating = 8.0 ]
Movie [ id = 8, title = Jurassic World, release date = 2015-06-12, budget = 1.5E8, revenue = 1.671537444E9, runtime = 124 minutes, rating = 6.7 ]
Movie [ id = 9, title = The Lion King, release date = 2019-07-12, budget = 2.6E8, revenue = 1.663075401E9, runtime = 118 minutes, rating = 7.1 ]
Movie [ id = 10, title = The Avengers, release date = 2012-04-25, budget = 2.2E8, revenue = 1.518815515E9, runtime = 143 minutes, rating = 7.7 ]
Movie [ id = 11, title = Furious 7, release date = 2015-04-01, budget = 1.9E8, revenue = 1.515341399E9, runtime = 137 minutes, rating = 7.2 ]
Movie [ id = 12, title = Top Gun: Maverick, release date = 2022-05-24, budget = 1.7E8, revenue = 1.488732821E9, runtime = 131 minutes, rating = 8.3 ]
Movie [ id = 13, title = Frozen II, release date = 2019-11-20, budget = 1.5E8, revenue = 1.450026933E9, runtime = 103 minutes, rating = 7.3 ]
Movie [ id = 14, title = Avengers: Age of Ultron, release date = 2015-04-22, budget = 3.65E8, revenue = 1.405403694E9, runtime = 141 minutes, rating = 7.3 ]
Movie [ id = 15, title = Black Panther, release date = 2018-02-13, budget = 2.0E8, revenue = 1.349926083E9, runtime = 135 minutes, rating = 7.4 ]
Movie [ id = 16, title = Harry Potter and the Deathly Hallows: Part 2, release date = 2011-07-07, budget = 1.25E8, revenue = 1.341511219E9, runtime = 130 minutes, rating = 8.1 ]
Movie [ id = 17, title = Star Wars: The Last Jedi, release date = 2017-12-13, budget = 2.0E8, revenue = 1.33269883E9, runtime = 152 minutes, rating = 6.8 ]
Movie [ id = 18, title = Jurassic World: Fallen Kingdom, release date = 2018-06-22, budget = 1.7E8, revenue = 1.310466296E9, runtime = 129 minutes, rating = 6.5 ]
Movie [ id = 19, title = Frozen, release date = 2013-11-20, budget = 1.5E8, revenue = 1.274219009E9, runtime = 102 minutes, rating = 7.2 ]
Movie [ id = 20, title = Beauty and the Beast, release date = 2017-03-16, budget = 1.6E8, revenue = 1.266115964E9, runtime = 129 minutes, rating = 7.0 ]
Movie [ id = 21, title = Incredibles 2, release date = 2018-06-14, budget = 2.0E8, revenue = 1.242805359E9, runtime = 118 minutes, rating = 7.5 ]
Movie [ id = 22, title = The Fate of the Furious, release date = 2017-04-12, budget = 2.5E8, revenue = 1.236005118E9, runtime = 136 minutes, rating = 6.9 ]
Movie [ id = 23, title = Iron Man 3, release date = 2013-04-18, budget = 2.0E8, revenue = 1.215577205E9, runtime = 130 minutes, rating = 6.9 ]
Movie [ id = 24, title = Minions, release date = 2015-06-17, budget = 7.4E7, revenue = 1.156730962E9, runtime = 91 minutes, rating = 6.4 ]
Movie [ id = 25, title = Captain America: Civil War, release date = 2016-04-27, budget = 2.5E8, revenue = 1.153337496E9, runtime = 147 minutes, rating = 7.4 ]
Movie [ id = 26, title = Aquaman, release date = 2018-12-07, budget = 1.6E8, revenue = 1.148528393E9, runtime = 143 minutes, rating = 6.9 ]
Movie [ id = 27, title = Skyfall, release date = 2012-10-24, budget = 2.0E8, revenue = 1.142471295E9, runtime = 143 minutes, rating = 7.2 ]
Movie [ id = 28, title = Spider-Man: Far From Home, release date = 2019-06-28, budget = 1.6E8, revenue = 1.131927996E9, runtime = 129 minutes, rating = 7.5 ]
Movie [ id = 29, title = Captain Marvel, release date = 2019-03-06, budget = 1.52E8, revenue = 1.131416446E9, runtime = 124 minutes, rating = 6.9 ]
```

Finally, when all methods have been executed, the Spring application will shutdown.



```
2024-08-30T16:43:37.414-06:00  INFO 182363 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
2024-08-30T16:43:37.416-06:00  INFO 182363 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
2024-08-30T16:43:37.424-06:00  INFO 182363 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Conclusion

I had previously used JDBC to establish a connection to databases, but hadn't really considered the security risks of SQL Injection attacks. This project has helped me understand the importance of working with layers of abstraction, minimizing the potential risks of a project, as well as producing cleaner code.