Technical Solution Description

DESCRIPTION. Application “UberBahn” represents an information system of a company, which provides passenger rail transportation. It allows adding stations, routes and getting information about trains and registered passengers for its employees. Application provides clients with information about trains, station timetables and gives opportunity to buy tickets.

USED TECHNOLOGIES AND FRAMEWORKS.

IDE – IntelliJ Idea 16.1

JDK 1.8

Apache Tomcat 8.5.2

Servlet/JSP

MySQL Server 5.7

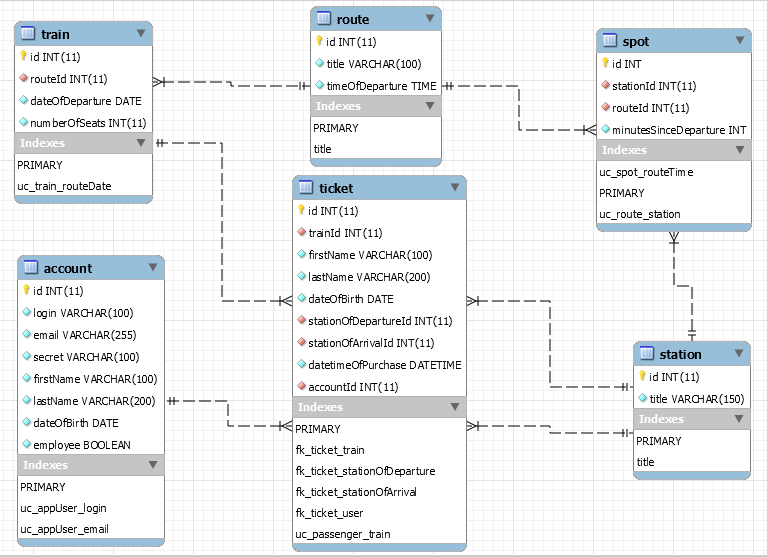
Apache Mavev 3.3.9

JPA 2.0 / Hibernate 5.1

HTML, CSS

JavaScript, jQuery

SCHEME OD DATABASE.



The scheme of database consists of 6 tables: train, route, spot, station, ticket and account.

Each table includes primary key – id.

Table station contains information about station titles (unique), table route contains information about route title (unique) and time of departure, table account includes login (unique), email (unique), secret (or password), first name, last name, date of birth and information whether a user is an employee or not.

Table train contains information about route (relation between table train and table route Many-To-One), date of departure and number of seats. Relation between tables route and station Many-To-Many, to resolve this problem table spot is added, which includes routeId and stationId (relation between tables spot and route or spot and station Many-To-One), minutes since departure. Unique constrains are routeId and stationId, routeId and time since departure. Table ticket provides information about train (relation between tables ticket and train Many-To-One), passenger first name, last name, date of birth (unique constraint), stations of departure and arrival (relation between tables ticket and station Many-To-One), date and time of purchase and account (relation between tables ticket and account Many-To-One).

MODULES OF THE APPLICATION

1. It must be written in English (translation to German is welcomed);

2. Don’t forget to add a section “Content”;

7. Describe your explication and implementation of the model from the task(for example, do you

have routes from point A to point B? What is it? What is a timetable? What is a basket?...)

8. Describe your modules in the application and their interaction.

9. Describe your UI: filters, do you have a template or common parts, give names of mail pages in

the code, css, used library…

10. Describe the level of business logic(services, how are they called, the list of service names with

“short” description);

11. Describe your entities, dao, how do you handle transactions;

12. Attach screenshots of the application(main pages);

13. Junit tests: their names and what they test;

14. How is your application built and deployed?

15. How is the logging configured, give a short example of the log file content;

16. What patterns are used in the application(if they are used), is there any additional mapping,

implemented algorithms? Additional util classes or constant list? Is there any security checks?

17. To make your text more readable provide use case diagrams, class diagrams, sequence

diagrams;

19. If you have any ideas about the improvement of the application or new features to be

implemented in the next releases, please describe them too.