

Digger

Database Schema Documentation Tool

Hildeberto Mendonca

Version 1.2.0

Table of Contents

1. Introduction	2
2. Installation	3
2.1. Installing the Released File	3
2.1.1. Using the Embedded Storage	3
2.1.2. Using PostgreSQL Storage	3
2.2. Installing From Source	4
3. Security	6
3.1. Signing Up	6
3.2. Login	6
4. Features	8
4.1. Administration	8
4.1.1. Users	8
Enabling and Disabling a User	8
Changing the Role of a User	8
4.2. Datasources	9
4.2.1. Creating and Editing a Datasource	9
4.2.2. The Datasource	10
4.3. Tables	10
4.3.1. Documenting a Table	11
4.3.2. A Table	12
4.4. Ignored Tables	12
4.4.1. Ignoring Tables	13
4.5. Columns	13
4.5.1. Documenting a Column	13
4.5.2. A Column	14
5. Contributing to the Project	15
5.1. Assumptions	15
5.2. Local Environment Setup	15
5.3. Data Model	15
5.4. Deployment	16
5.5. Test Automation	16
5.6. Technologies in Use	17
5.7. Using Git	17
5.7.1. Changing The Author To The One Recognizable by GitHub	17
5.7.2. Changing Several Commits in Bulk	17
5.7.3. Adding a File to the Most Recent Commit	17

Digger is a light-weight web application to centralize and share the accumulated collective knowledge about all the relational databases of the organization.

Chapter 1. Introduction

After decades of software development, we realized that data is more valuable than software. An application is more likely to be rewritten in a modern technology and continue pointing to an existing database than an existing application be modified to access data from a different database. Of course both cases exist, but applications tend to become more chaotic than databases.

Even with a longer time span, databases are rarely documented. Often, developers have to read the code to understand the meaning of tables, views, columns, and how to use them. It is not rare to find columns and tables that are not referenced at all, but we never know whether they are still in use by an obscure trigger, stored procedure or third-part application. If at least they had a defragmented and up-to-date documentation they could rely on.

Chapter 2. Installation

Digger is easy to install but it requires Java 8+ installed and configured in the system. The application comes with an embedded database for simple use cases, but it can also be configured to store data in a PostgreSQL database server, which also has to be installed and configured separately.

2.1. Installing the Released File

Download Digger from the [release page](#) and save it where you want to install it. That is a `jar` file with the naming convention: `digger-<version>.jar`. For example: `digger-1.2.0.jar`.

2.1.1. Using the Embedded Storage

To run Digger with its default configuration, go to the terminal and execute:

```
$ cd <path-to-digger-folder>
$ java -jar digger-1.2.0.jar
```

A few moments later, open your browser and visit the address <http://localhost:8080> to use Digger with its embedded database. The folder `data` is automatically created during the initialization and the sign up page below is presented by default.

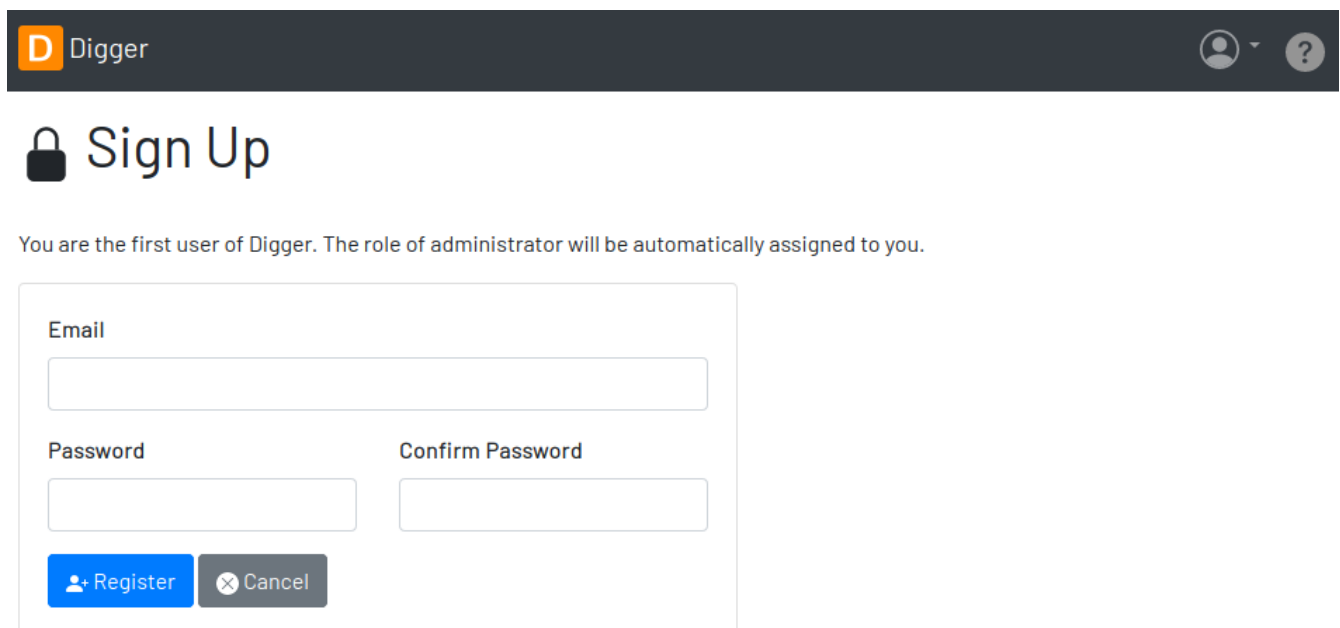
The screenshot shows the Digger application's initial setup page. At the top, there is a dark header bar with the Digger logo (an orange square with a white 'D') and the word 'Digger' in white. On the right side of the header, there are two circular icons: a user profile icon and a help/question mark icon. Below the header, the main content area has a light gray background. It starts with a large black padlock icon followed by the text 'Sign Up' in a large, bold, black font. Below this, a smaller line of text states: 'You are the first user of Digger. The role of administrator will be automatically assigned to you.' Underneath this text is a white rectangular form with a thin gray border. Inside the form, there are three input fields: one for 'Email' at the top, and two for 'Password' and 'Confirm Password' side-by-side below it. At the bottom of the form, there are two buttons: a blue button with a white user icon and the text 'Register', and a gray button with a white 'X' icon and the text 'Cancel'.

Figure 1. Initial Setup

2.1.2. Using PostgreSQL Storage

The embedded database is robust enough to support a reasonable volume of data, but it won't scale to support multiple concurrent users. For that, you can use PostgreSQL to handle a larger demand for information. To switch to PostgreSQL:

1. if the application is already running, stop it using `[Ctrl+C]` in the terminal
2. create a sub-directory named `config` in the same directory of the application
3. download the files `application.properties` and `application-server.properties` and save them in the `config` folder
4. open the file `application.properties` and change the following entry from `embedded` to `server`:

```
spring.profiles.active=server
```

5. Then open the file `application-server.properties` and change the following connection parameters to your PostgreSQL server:

```
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/digger
spring.datasource.username=digger_usr
spring.datasource.password=secret
```

6. Restart the application to take the new configuration into account:

```
$ java -jar digger-1.2.0.jar
```

7. Finally, refresh the page <http://localhost:8080>

Make sure the database user has full rights over Digger's database, so it can generate the schema and perform all operations.

2.2. Installing From Source

A new version of Digger is released from time to time, but if you can't wait for a feature that was just finished, then you may need to build Digger from source. To do it, you need:

- [JDK](#), a Java Development Kit to compile and run the code,
- [Maven](#), a traditional software life-cycle management tool for Java, and
- [Git](#), a distributed version control system. Please, visit their respective documentation and get them installed and configured in your system.

To start, fetch the code from GitHub:

```
$ git clone https://github.com/htmfilho/digger.git
```

Then build the project:

```
$ cd digger
$ mvn package
```

All the artifacts you need are ready! The jar file is now available at **target/** and the configuration files at **config/**. You can run it using the java command:

```
$ java -jar target/digger-1.2.0.jar
```

or Maven:

```
$ mvn spring-boot:run
```

If you already have Digger installed, just put the generated jar file in the same folder of the existing installation and remove the old jar. Execute the new jar from that point on.

You can also get all subsequent changes whenever they are available by fetching updates:

```
$ git pull origin master
```

Then you can package and run it:

```
$ mvn clean package
```

Chapter 3. Security

Digger ensures that only authorized people in the organization are allowed to document and to access the documentation of the schemes. Users are managed by the application and their passwords are strongly encrypted in the database, to the point they cannot be recovered, only reset.

3.1. Signing Up

When Digger starts for the first time, it forces the creation of the first user account by automatically redirecting the user to the Sign Up page. The role of administrator (ROLE_ADMIN) is automatically assigned to the first user, who is empowered to manage the application including other users.

Digger

Sign Up

Register as a reader of database documentations. Your registration is subject of approval.

Email
me@hildeberto.com

Password
.....

Confirm Password
.....

[Register](#) [Cancel](#)

Copyright © 2019-2020 Hildeberto Mendonca

Figure 2. User Sign Up

All people signing up after the first user are disabled and assigned to the role of Reader by default. That's why the user cannot login after the sign up. The administrator must enable the user and assign him or her to the appropriate role or leave the user as reader.

3.2. Login

The login tries to match the user's credentials. If the matching is successful, the user is allowed into the application to access confidential information, otherwise the user is informed that the matching was unsuccessful.

Login

Email

me@hildeberto.com

Password

 Login

 Cancel

Copyright © 2019-2020 Hildeberto Mendonca

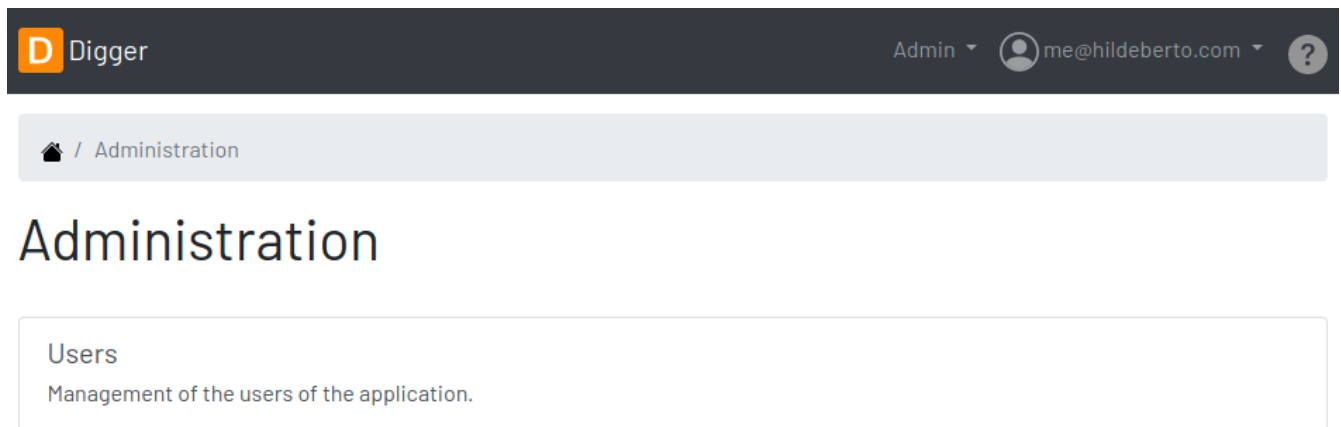
Figure 3. User authentication

Chapter 4. Features

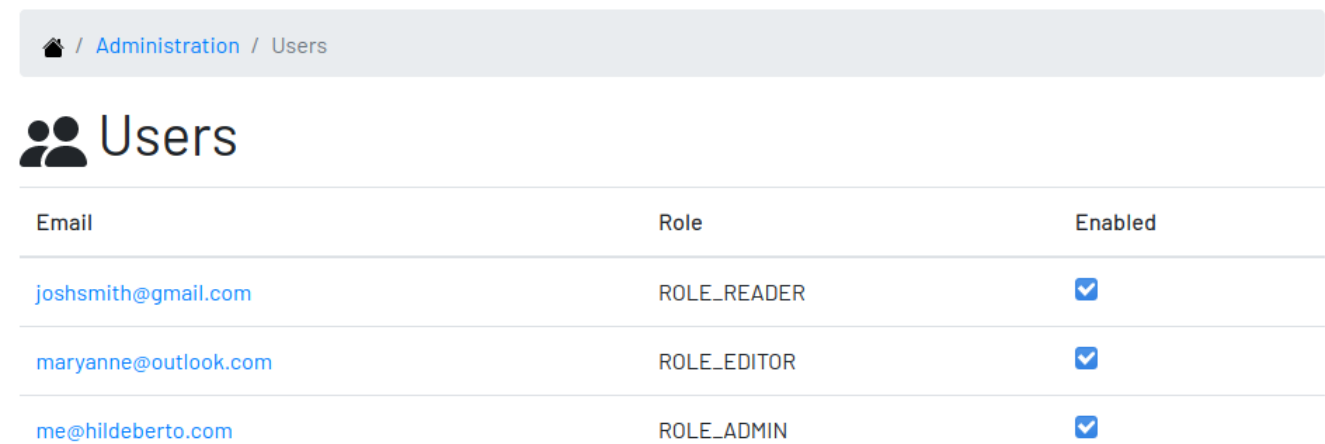
Digger gives you a good set of features to help you document the database schemes of your organization.

4.1. Administration

The administration is accessible via the top menu, in the "Admin" option. It allows the administrator to manage user accounts.



4.1.1. Users



Enabling and Disabling a User

After signing up, a user doesn't have instant access to Digger. All users are disabled by default and the administrator has to enable them. To enable a user:

1. click on the "Admin" option on the top menu and select "Users" in the list
2. check the users you want to enable and uncheck the ones you want to disable


Changing the Role of a User

Digger defines 3 levels of authority represented by roles. They are:

- **Administrator:** has access to all functionalities of the system.
- **Editor:** has rights limited to document and visualize the documentation of the schemas.
- **Reader:** has rights limited to visualizing the documentation of the schema.

The first user of Digger is assigned to the role of Administrator and all subsequent users are assigned to the role of Reader. Only the administrator has the right to change the role of a user. To do this:

1. Click on the "Admin" option on the top menu and select "Users" in the list
2. click on the user you want to change
3. click on the button "Options" and select "Edit" in the list
4. select the role you want for that user and save

 / [Administration](#) / [Users](#) / [maryanne@outlook.com](#)

User


Username
maryanne@outlook.com

Role
☐ Admin
 ☒ Editor
 ☐ Reader


☒ Enabled

4.2. Datasources

Datasource is a reference to an existing database that we intend to document. A datasource has enough information to connect to the database and extract metadata from it.



Datasources



Bookstore Database Documentation of the legacy bookstore database	org.postgresql.Driver
Digger's Database Documentation of Digger's Database	org.h2.Driver

4.2.1. Creating and Editing a Datasource

To create a new datasource, click on the **New** button on the top right of the list of datasources. It opens the datasource form, where you can give it a **Name**, give more details about it in the **Description**, and inform the connection attributes. The **Driver Class** drop-down field offers a list of the currently supported database engines. Each driver requires a different URL format, so when a driver is selected, its corresponding URL template appears below the **URL** field for reference.

Finally, inform a valid **Username** and **Password** with at least *Read* privileges to the database. Click on **Save** to register the information or **Cancel** to go back to the datasource list.

Datasource

Name *

Bookstore Database

Description

Documentation of the legacy bookstore database

Driver Class *

org.postgresql.Driver

URL *


jdbc:postgresql://localhost:5432/bookstore

Username

bookstore_usr

Password


 Save

 Cancel

To edit a datasource, click on it in the list. In the datasource page, click on the button **Options** on the top right, then select the option **Edit**. The same form appears, but this time completely filled. Make the intended changes and **Save**.


4.2.2. The Datasource

The datasource page shows all information related to the datasource, as well as all possible operations such as **Edit**, **Remove**, **Add Table**, **Ignore Tables**, etc.

 / Bookstore Database

Datasource

Options ▾

Name	Description	Status
Bookstore Database	Documentation of the legacy bookstore database	
Driver Class	URL	Username
org.postgresql.Driver	jdbc:postgresql://localhost:5432/bookstore	bookstore_usr

100%

4.3. Tables

A datasource's Table is a tabular structure used to store, organize and retrieve data. It can be a database table, a temporary table, a view, and other vendor specific alternatives. They are listed in the datasource page, from where they can be reached and documented.

Datasource

Options ▾

Name	Description	Status
Bookstore Database	Documentation of the legacy bookstore database	🟢
Driver Class	URL	Username
org.postgresql.Driver	jdbc:postgresql://localhost:5432/bookstore	bookstore_usr

50%

Tables Ignored

📊 Tables

+ New

Name	Friendly Name	Type
ATBK	Author Of The Book	TABLE
ATHR	Author	TABLE
BOOK	Book	TABLE

4.3.1. Documenting a Table

To document a table, go to the datasource that the table belongs to, then to the "Tables" section, and click on the **New** button on the right. Filling the form by selecting the physical name of the table in the dropdown, confirming the type that is automatically detected, a friendly name that is more readable than the physical name, and write down everything you know about that table.

Click on the **Save** button to complete or **Cancel** to go back to the datasource page.

📊 Table

Name *

ATBK

▾

Type

TABLE

Friendly Name

Author Of The Book

Documentation (Asciidoc)

Edit

Preview

The table that links a book to an author or several authors, which is also useful to know all books written by an author.


Save


Cancel

The **Documentation** field uses [Asciidoc](#) as markup language. It has a human friendly syntax to allow anybody write rich content without touching any HTML or CSS code.

4.3.2. A Table

The table's page shows all information related to the table, including its columns. To edit a table, click on the **Options** button on the top right then select the option **Edit**. The form appears filled with the table's attributes and documentation. Make the intended changes and save, or cancel to return to the table's page.

 / [Bookstore Database](#) / Author Of The Book

 **Table**


Options ▾

Datasource	Friendly Name	Physical Name	Type
Bookstore Database	Author Of The Book	ATBK	TABLE



Documentation

The table that links a book to an author or several authors, which is also useful to know all books written by an author.

100%

 **Columns**


+ New

Name	Friendly	Type	Nullable	
ATHR	Author	int8 (19)	false	
BOOK	Book	int8 (19)	false	

4.4. Ignored Tables

Datasource

Options ▾

Name	Description	Status
Bookstore Database	Documentation of the legacy bookstore database	
Driver Class	URL	Username
org.postgresql.Driver	jdbc:postgresql://localhost:5432/bookstore	bookstore_usr




100%

Tables

Ignored

Ignored Tables

+ New

Name	
ATHR_IDAT_seq	
ATHR_pkey	
BOOK_IDBK_seq	

4.4.1. Ignoring Tables

Ignoring Tables

☐ Check All

☐ ATHR_IDAT_seq

☐ _pg_foreign_table_columns

☐ applicable_roles

☐ pg_aggregate

 Save

 Cancel

4.5. Columns

4.5.1. Documenting a Column

Column

Table

Book (BOOK)

Name *

ISBN

Friendly Name

Isbn

Type

varchar (20)

Nullable

true

Default Value

Foreign Table

Select...

Foreign Column

Select...

Documentation ([AsciiDoc](#))

Edit

[Preview](#)

****International Standard Book Number (ISBN)****

The International Standard Book Number (ISBN) is a numeric commercial book identifier that is intended to be unique. Publishers purchase ISBNs from an affiliate of the International ISBN Agency. An ISBN is assigned to each separate edition and variation (except reprintings) of a publication. For example, an e-book, a paperback and a hardcover edition of the same book will each have a different ISBN. The ISBN is ten digits long if assigned before 2007, and thirteen digits long if assigned on or after 1 January 2007. The method of assigning an ISBN is nation-specific and varies between countries, often depending on how large the publishing industry is within a country. ([https://en.wikipedia.org/wiki/International_Standard_Book_Number\[Wikipedia\]](https://en.wikipedia.org/wiki/International_Standard_Book_Number[Wikipedia]))

[Save](#)

[Cancel](#)

4.5.2. A Column

[Home](#) / [Bookstore Database](#) / [Book](#) / International Standard Book Number

Column

Options ▾

Table

[Book](#)

Friendly Name

International Standard Book Number

Physical Name

ISBN

Type

varchar (20)

Nullable

true

Default Value

Foreign Table

[Table](#)

Foreign Column

[Column](#)

Documentation

International Standard Book Number (ISBN)

The International Standard Book Number (ISBN) is a numeric commercial book identifier that is intended to be unique. Publishers purchase ISBNs from an affiliate of the International ISBN Agency. An ISBN is assigned to each separate edition and variation (except reprintings) of a publication. For example, an e-book, a paperback and a hardcover edition of the same book will each have a different ISBN. The ISBN is ten digits long if assigned before 2007, and thirteen digits long if assigned on or after 1 January 2007. The method of assigning an ISBN is nation-specific and varies between countries, often depending on how large the publishing industry is within a country. ([Wikipedia](#))

Chapter 5. Contributing to the Project

Follow these instructions if you want to contribute to Digger.

5.1. Assumptions

We assume your development environment is configured with:

- **Java 8+:** you can perform the commands `java` and `javac` in your terminal
- **Maven 3:** you can perform the command `mvn` in your terminal
- **Git:** you can perform the command `git` in your terminal

5.2. Local Environment Setup

We favour the use of the command line to set up the local environment, so we do not depend on any other tool for this basic step. Open the Windows/Linux terminal and start by cloning the repository in your local machine:

```
$ cd [your-java-projects-folder]
$ git clone https://github.com/htmfilho/digger.git
```

It creates the folder `digger` that contains the entire source code of the application. Execute the following Maven command to build, test, and run the application:

```
$ cd digger
$ mvn spring-boot:run
```

Visit the local address <http://localhost:8080/> to use the application. To stop it, type `Ctrl+C` on the terminal.

5.3. Data Model

The data managed by Digger is persisted in a relational database. If you launched Digger as is, without changing the configuration, you are using the embedded database [H2](#). If you are using the server configuration then you are using [PostgreSQL](#). The data is organized according to the following diagram.

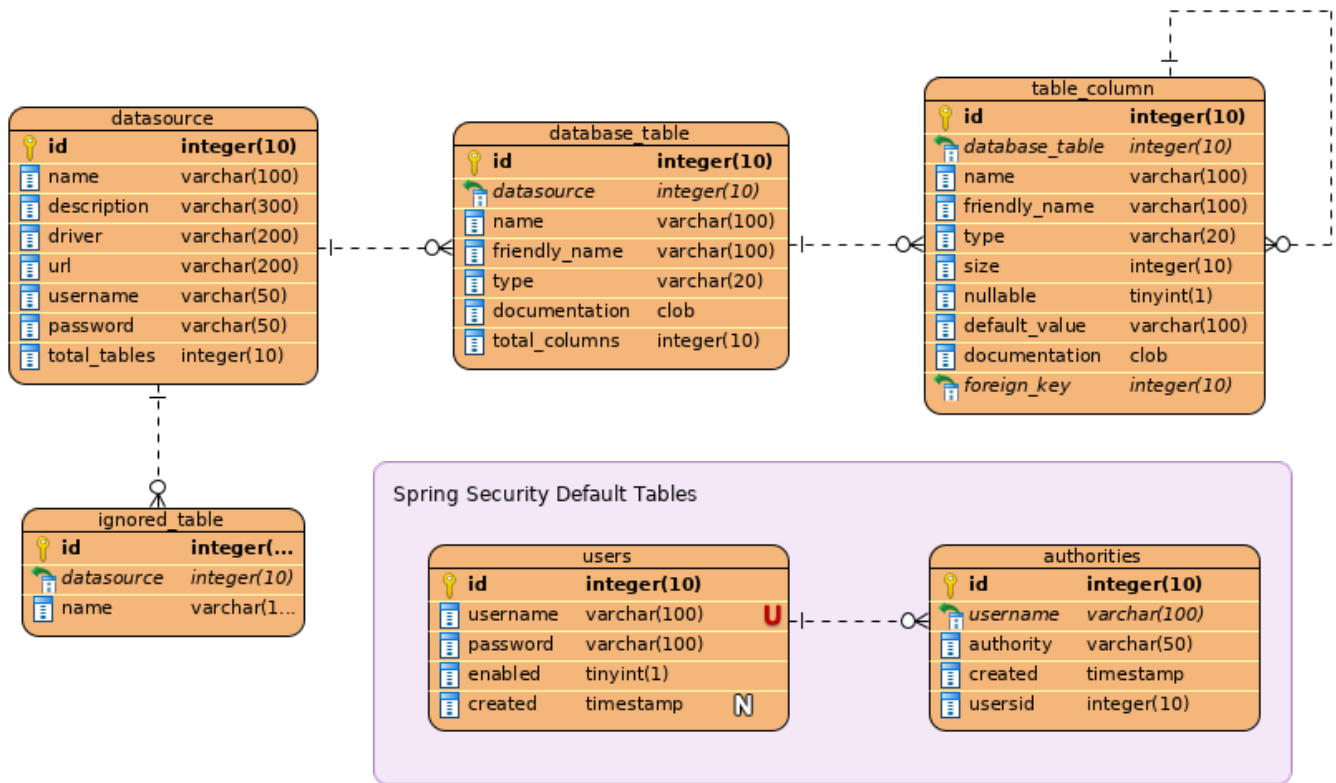


Figure 4. Digger's Entity Relational Model

5.4. Deployment

Create a deployment package using Maven:

```
$ mvn clean package
```

It creates a Java standalone application package in the folder **target**.

If the default port **8080** is already in use, set the environment variable **SERVER_PORT** to **8081**.

Run the package to check if everything works:

```
$ cd [your-java-projects-folder]/digger
$ java -jar target/digger-<version>-SNAPSHOT.jar
```

5.5. Test Automation

Digger was initially released with very few automated tests. This is not good, but we wanted to give some use to the book [Refactoring](#), by [Martin Fowler](#). This book explains how to refactor the code by first writing tests to ensure the refactoring won't break existing functionalities. So, our approach for testing is basically ensuring regression, increasing test coverage as the application is modified.

To execute the test suite, run:

```
$ mvn test
```

Only submit your pull request if these tests pass. To see the test coverage report, open the page generated at [target/site/jacoco](#).

5.6. Technologies in Use

- [Spring MVC](#)
- [Spring Security](#)
- [Thymeleaf](#)

5.7. Using Git

5.7.1. Changing The Author To The One Recognizable by GitHub

In case your default Git author is not the same as GitHub, configure the author of the repository:

```
$ git config user.name "John Doe"
$ git config user.email "john@doe.org"
```

It can also be done to a specific commit:

```
$ git commit --author="John Doe <john@doe.org>"
```

5.7.2. Changing Several Commits in Bulk

If commits were done with a wrong author, use Git Rebase to fix the authors of the commits:

```
$ git rebase -i -p <commit-id>
$ git commit --amend --author="John Doe <john@doe.org>"
$ git rebase --continue
$ git push -f origin master
```

5.7.3. Adding a File to the Most Recent Commit

```
$ git add missed-file.txt
$ git commit --amend
```