Endabgabe Feuerwerk

Hausarbeit

Entwicklung interaktiver Anwendungen II

WS 2020-2021

Erstellt von:

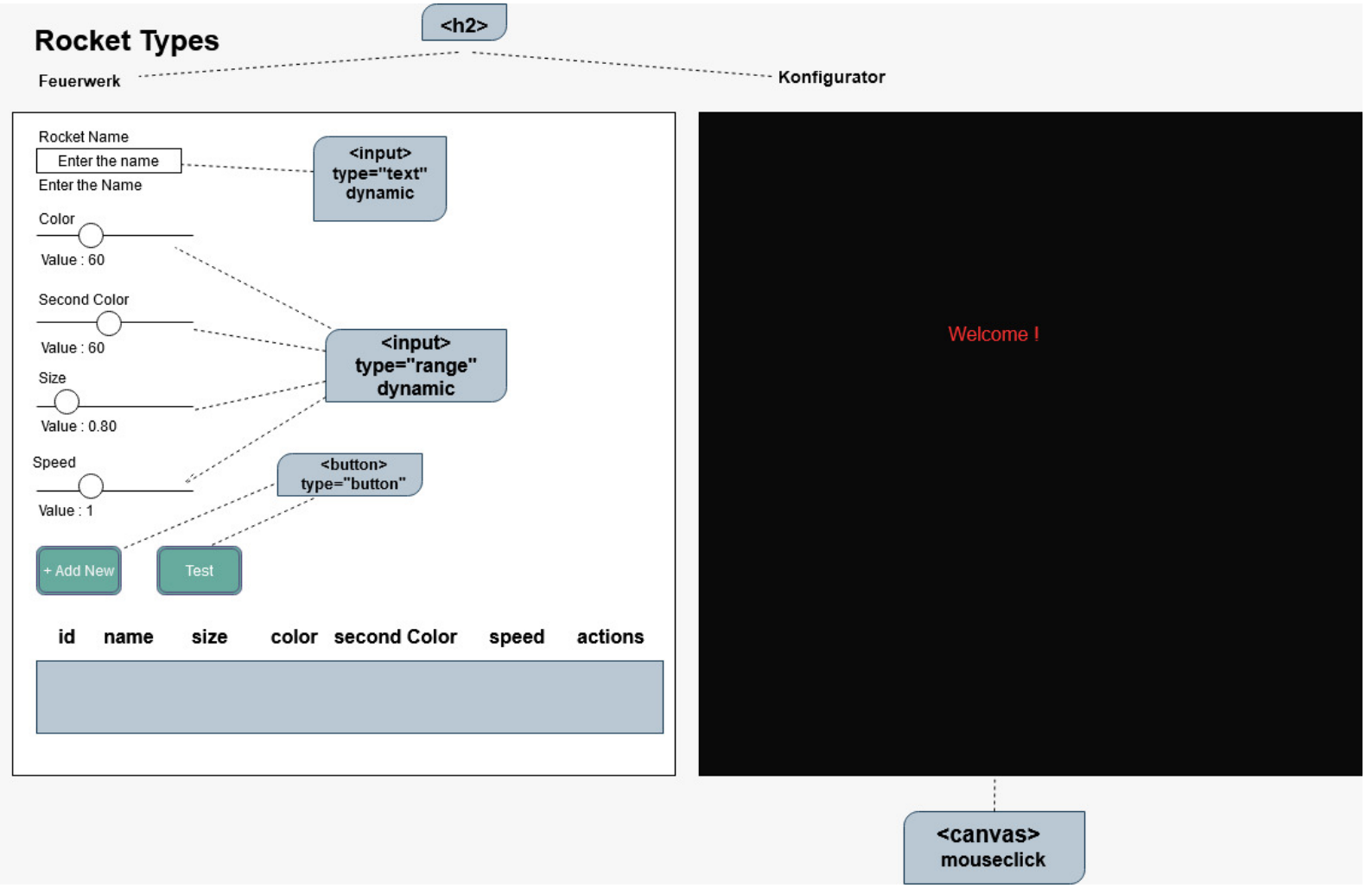
Oliver Fabio	Gern	265098
Emre	Aylan	265382
Amra	Dropic	265606
Selina	Wasinger	265186
Simon E. J.	Daiber	265607
Ebru	Yeniay	259405

Prof. Jirka Dell'Oro-Friedl

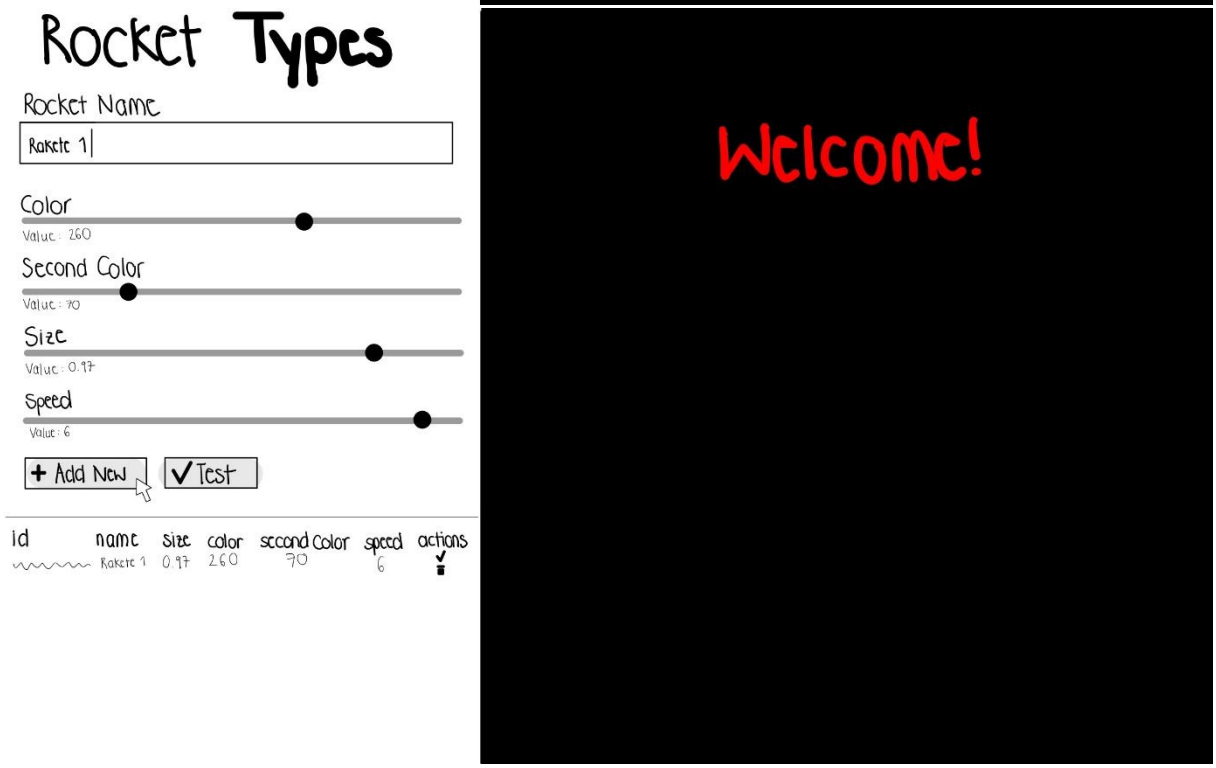
Anleitung zur Interaktion in der Anwendung „Feuerwerk“

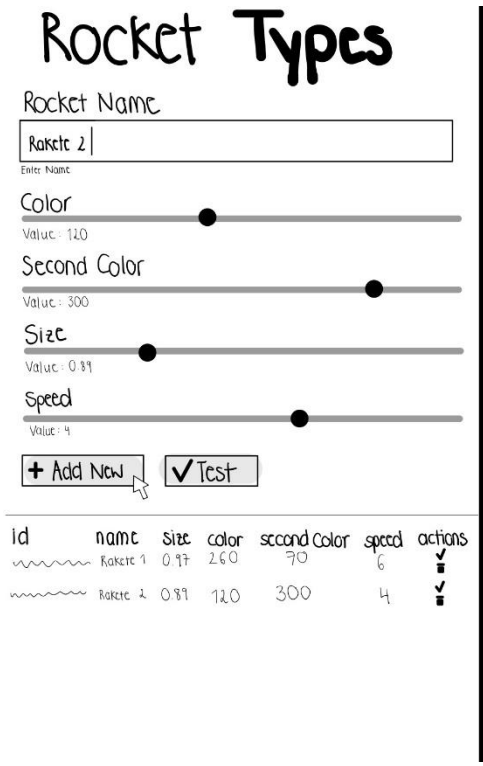
1. Es können unterschiedliche Raketen Typen erstellt werden.
2. Über das Eingabefeld „Rocket Name“ kann die zu erstellende Rakete benannt werden.
3. Über den Schieberegler „Color“ kann die erste Farbe mithilfe der HSV-Farbton Skala von 0 – 360 erstellt werden.
4. Über den Schieberegler „Second Color“ kann die zweite Farbe mithilfe der HSV-Farbton Skala von 0 – 360 erstellt werden. Die Farben werden dann später im Canvas als Explosionsmischung ausgegeben.
5. Über den Schieberegler „Size“ kann die Raketengröße bestimmt werden.
6. Über den Schieberegler „Speed“ kann die Geschwindigkeit der Rakete geändert werden.
7. Durch den Button „Test“ kann die Rakete im Canvas getestet werden, um zu entscheiden, ob sie der Datenbank hinzugefügt werden soll.
8. Wenn der Nutzer zufrieden ist, kann er über den Button „Add New“ die erstellte Rakete der Datenbank hinzufügen.
9. Es können beliebig viele verschiedene Raketen erstellt und getestet werden.
10. In der Datenbank selbst können die erstellten Raketen nochmal einzeln durch das „Harken Symbol“ im Canvas ausgegeben werden.
11. Um eine Rakete wieder aus der Datenbank zu löschen ist ein einfacher Klick auf den Mülleimer notwendig.
12. Sie haben sich erfolgreich Ihre Raketen zusammengestellt? Jetzt können Sie durch einen Klick auf das Canvas alle Raketen, die in der Datenbank hinterlegt wurden, auf einmal ausgeben lassen. Genießen Sie das Feuerwerk und viel Spaß beim Erstellen der verschiedenen Raketen.

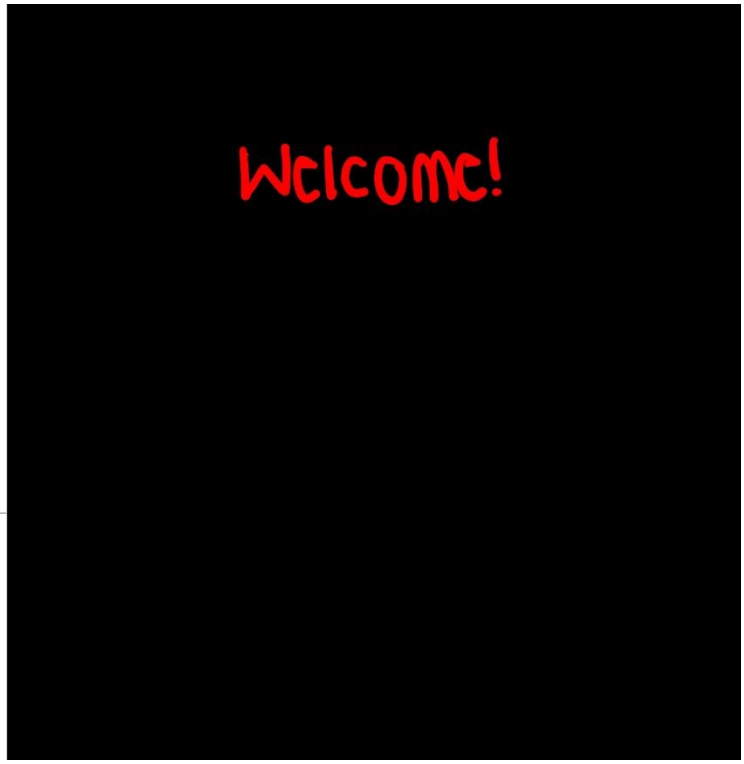
Scribble

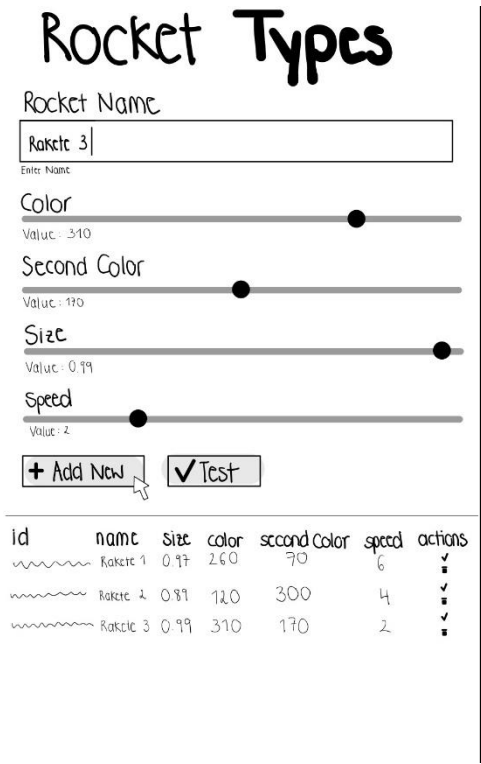

Rocket Types
<h2>
Feuerwerk
Konfigurator
Rocket Name
Enter the name
Enter the Name
<input> type="text" dynamic
Color
Value : 60
Second Color
Value : 60
Size
Value : 0.80
<input> type="range" dynamic
Speed
Value : 1
<button> type="button"
+ Add New
Test
id
name
size
color
second Color
speed
actions
Welcome !
<canvas> mouseclick

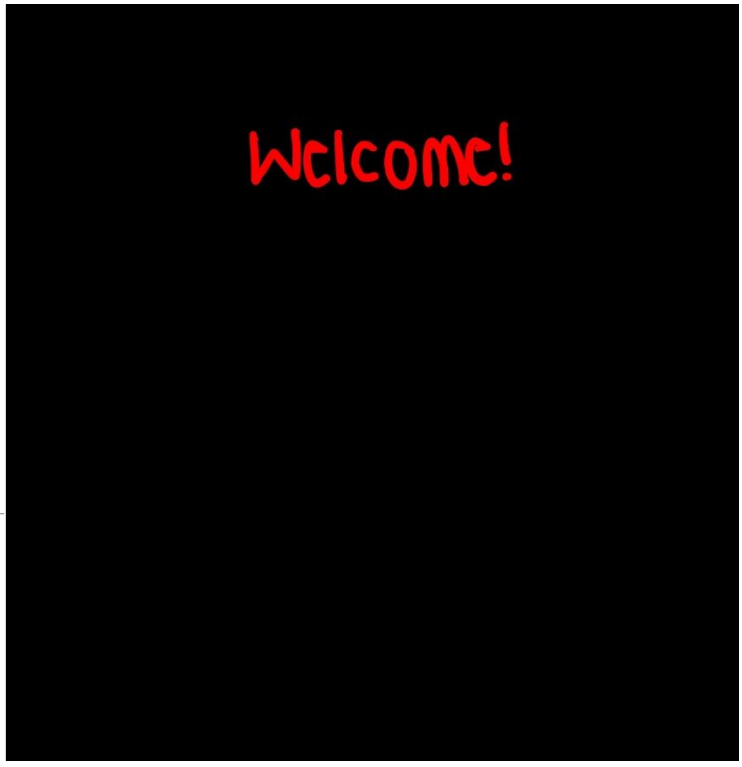

Rocket Types
Rocket Name
Value:
Color
Value:
Second Color
Value:
Size
Value:
Speed
+ Add New
✓Test
id
name
size
color
second Color
speed
actions
Welcome!

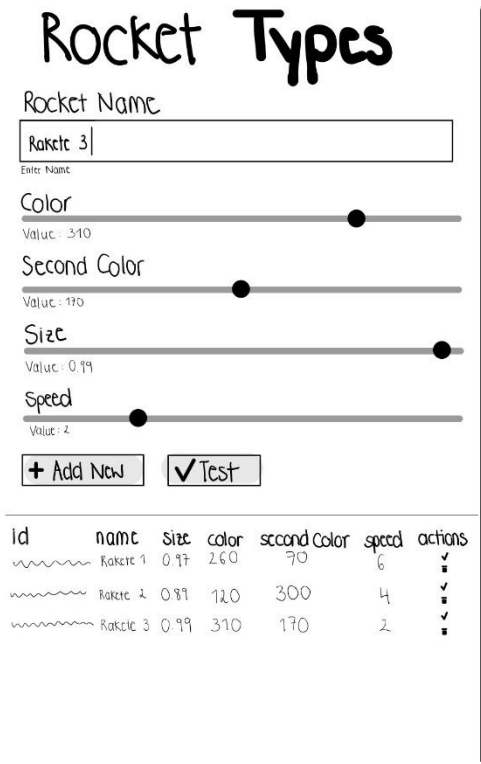

Rocket Types
Rocket Name
Rakete 1
Color
Value: 260
Second Color
Value: 70
Size
Value: 0.97
Speed
Value: 6
+ Add New
✓Test
id
name
size
color
second Color
speed
actions
Rakete 1
0.97
260
70
6
Welcome!

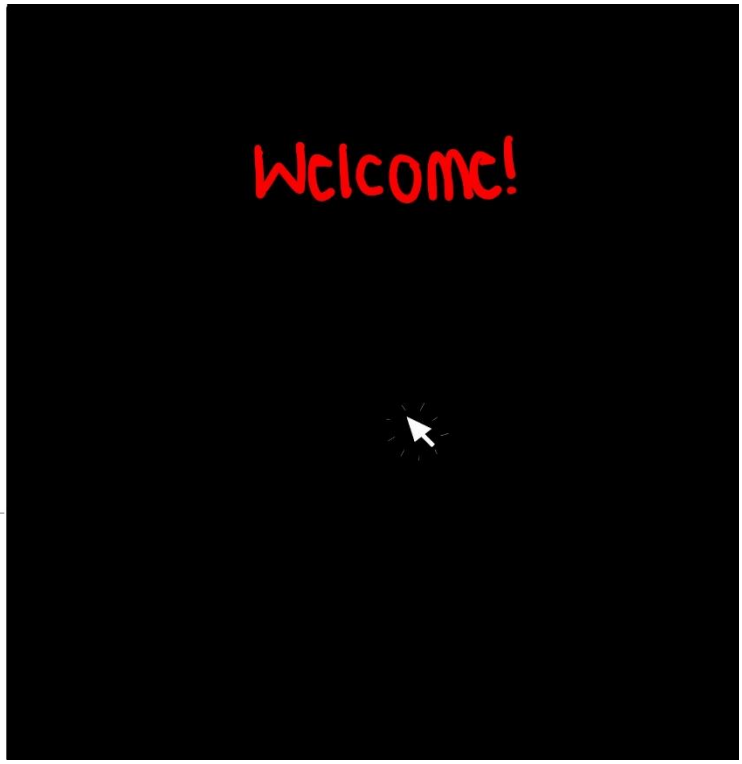

Rocket Types
Rocket Name
Rakete 2
Enter Name
Color
Value: 120
Second Color
Value: 300
Size
Value: 0.89
Speed
Value: 4
+ Add New
✓ Test
id name size color second Color speed actions
Rakete 1 0.97 260 70 6
Rakete 2 0.89 120 300 4

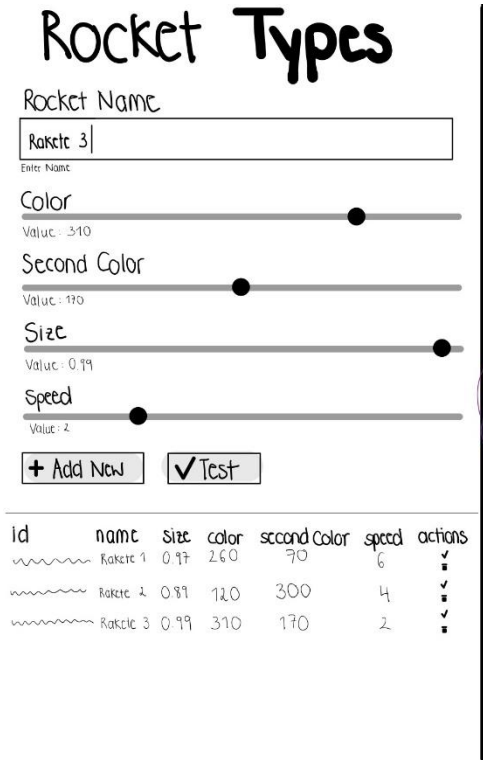

Welcome!


Rocket Types
Rocket Name
Rakete 3
Enter Name
Color
Value: 310
Second Color
Value: 170
Size
Value: 0.99
Speed
Value: 2
+ Add New
✓ Test
id name size color second Color speed actions
Rakete 1 0.97 260 70 6
Rakete 2 0.89 120 300 4
Rakete 3 0.99 310 170 2


Welcome!


Rocket Types
Rocket Name
Rakete 3
Enter Name
Color
Value: 310
Second Color
Value: 190
Size
Value: 0.99
Speed
Value: 2
+ Add New
✓ Test
id name size color second Color speed actions
Rakete 1 0.97 260 90 6
Rakete 2 0.89 120 300 4
Rakete 3 0.99 310 170 2


Welcome!


Rocket Types
Rocket Name
Rakete 3
Enter Name
Color
Value: 310
Second Color
Value: 190
Size
Value: 0.99
Speed
Value: 2
+ Add New
✓ Test
id name size color second Color speed actions
Rakete 1 0.97 260 90 6
Rakete 2 0.89 120 300 4
Rakete 3 0.99 310 170 2


Welcome!

Anleitung zur Installation der Anwendung „Feuerwerk“

Die Feuerwerk Anwendung kann unter Berücksichtigung einiger elementaren Prozesse innerhalb des Codes bzw. mithilfe der Datenbank Mongo und dem Server Heroku gestartet werden. Nur so kann die Anwendung in ihrer Fülle an Funktionen funktionieren und verwendet werden.

1. Der Client – Um die Anwendung auf einem eigenen Server zu starten muss die bestehende Serveradresse durch eine eigene Heroku Adresse getauscht werden. Der Client kann dann im Folgeprozess seine Anfragen an die gewechselte Adresse senden. (Voraussetzung: Registrierung bei Heroku und eine „App“ erstellen)

```
FrontendEIA2 > TS configuration.ts > ...
1   namespace firework {
2   // server url local and heroku
3   export const localhostUrl: string = 'http://127.0.0.1:8081/api/';
4   export const herokuUrl: string = "https://fireworkoli.herokuapp.com/api/";
5   }
6
```

2. Server – Die package.json muss erneuert werden. (die Codezeile in der auf die Server.js Datei verwiesen wird)→ Ausführung über Node

```
"start": "node ENDABGABE/Server/server.js"
```

Entsprechende Erweiterungen müssen importiert werden, diese dienen zur Vereinfachung für die Kommunikation mit dem Server. (Mit „express“ funktionierte es erst richtig) Die Idee dazu kam durch die Recherche in „Stackoverflow“.

```
EndAbagbeEIA2 > TS app.ts > ...
1   import { Routes } from './routes';
2   import * as express from 'express';
3   import * as bodyParser from "body-parser";
4   import * as mongoose from "mongoose";
5
```

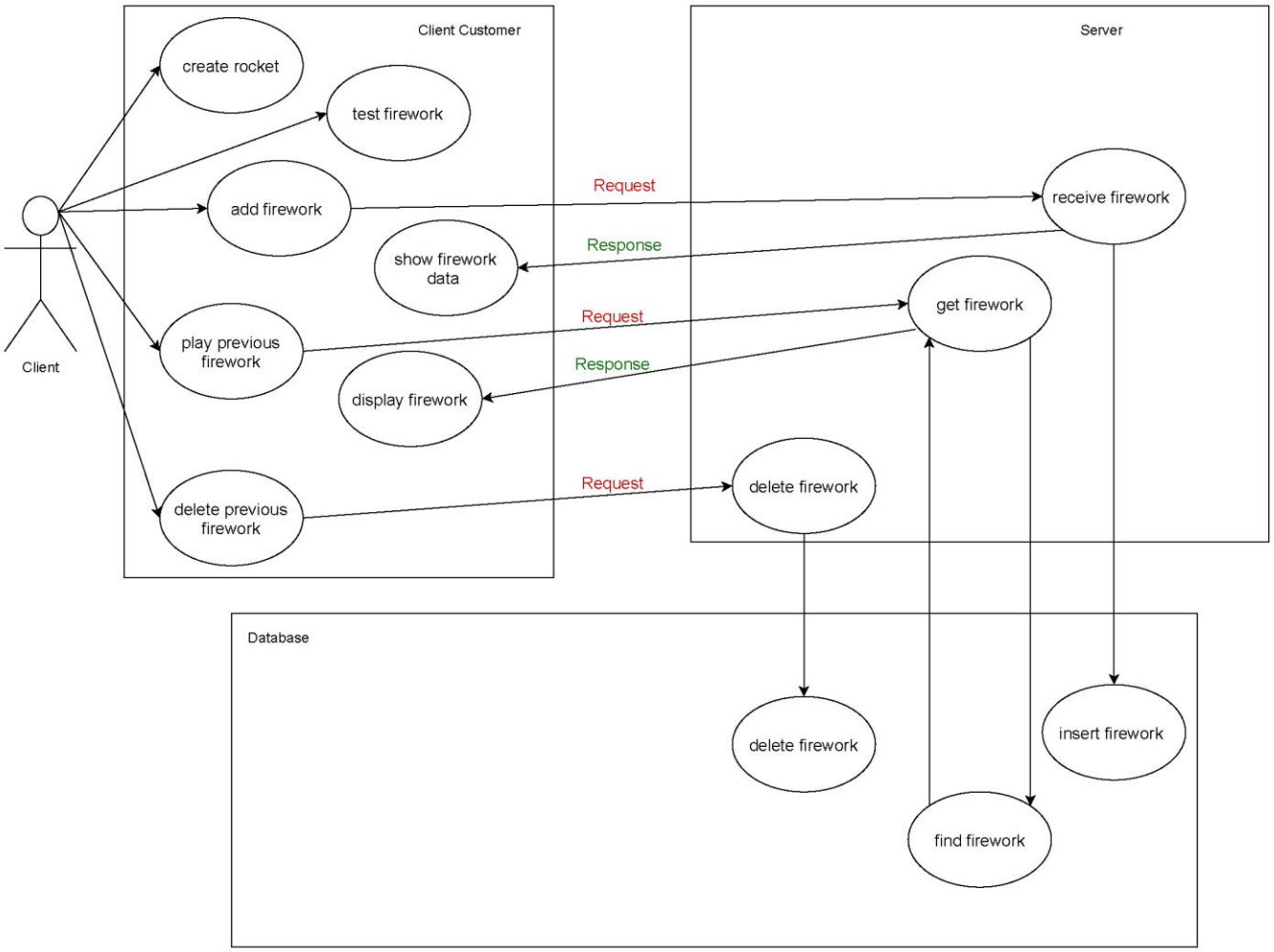
Ähnlich wie bei der Verbindung mit Heroku, benötigt man auch für die Verbindung mit dem Mongo Atlas einen Connection String zur Datenbank.

(IMPORTANT: <passwort> muss mit dem festgelegten Passwort ersetzt werden (hier: BeispielPasswort), <dbname> ebenso (hier: firework))

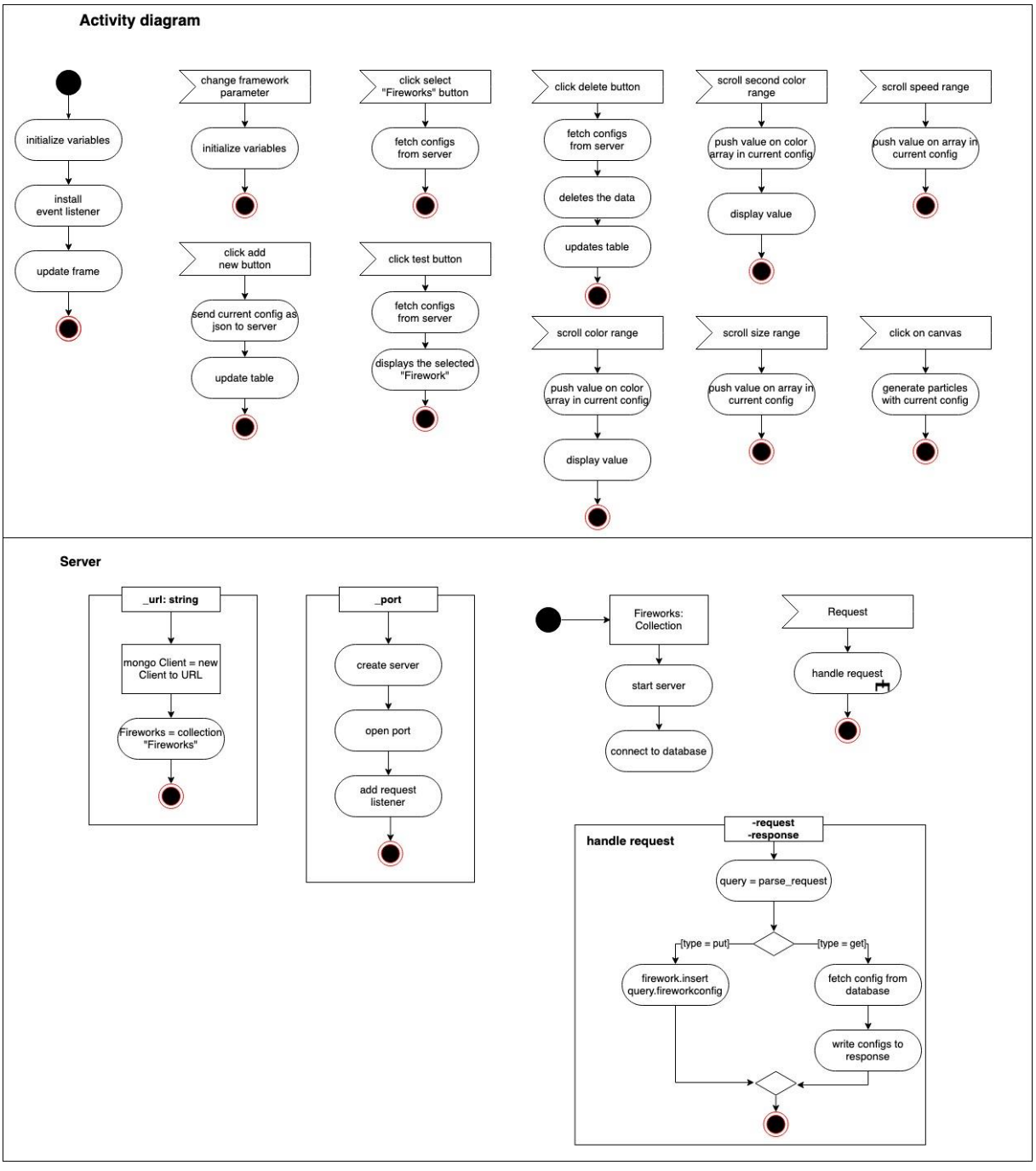
Durch die Kommentare im Code (grün) kann der weitere notwendige Prozess ausgelesen werden. Wie bereits beschrieben, hat bei uns durch Konfigurierung der „Express App“ den kompletten Backend Verlauf um einiges erleichtert und automatisiert.

```
class App {
  // create express application which is a nodejs internal
  public app: express.Application;
  //create the routes like /api/rockets
  public routes: Routes = new Routes();
  //local mongodb url
  public localMongoUrl: string = 'mongodb://127.0.0.1:27017/firework';
  //remote mongodb url
  public remoteMongoUrl: string = "mongodb+srv://gernoliv:BeispielPassword@cluster0.dlgwu.mongodb.net/firework?retryWrites=true&w=majority";
  // variable holding the parameter from  npm start if the database url should be remote or local
  public isRemote: boolean;


  constructor() {
    // create express app
    this.app = express();
    //load the confog
    this.config();
    //initialize thea app with the routes
    this.routes.routes(this.app);
    //check if the passed parameter is remote
    //process.argv = "remote"
    this.isRemote = process.argv[2] == "remote";
    //setup the mongo configuration
    this.mongoSetup();

  }
```

Use Case Diagramm



Aktivitätsdiagramm



Klassen Diagramm