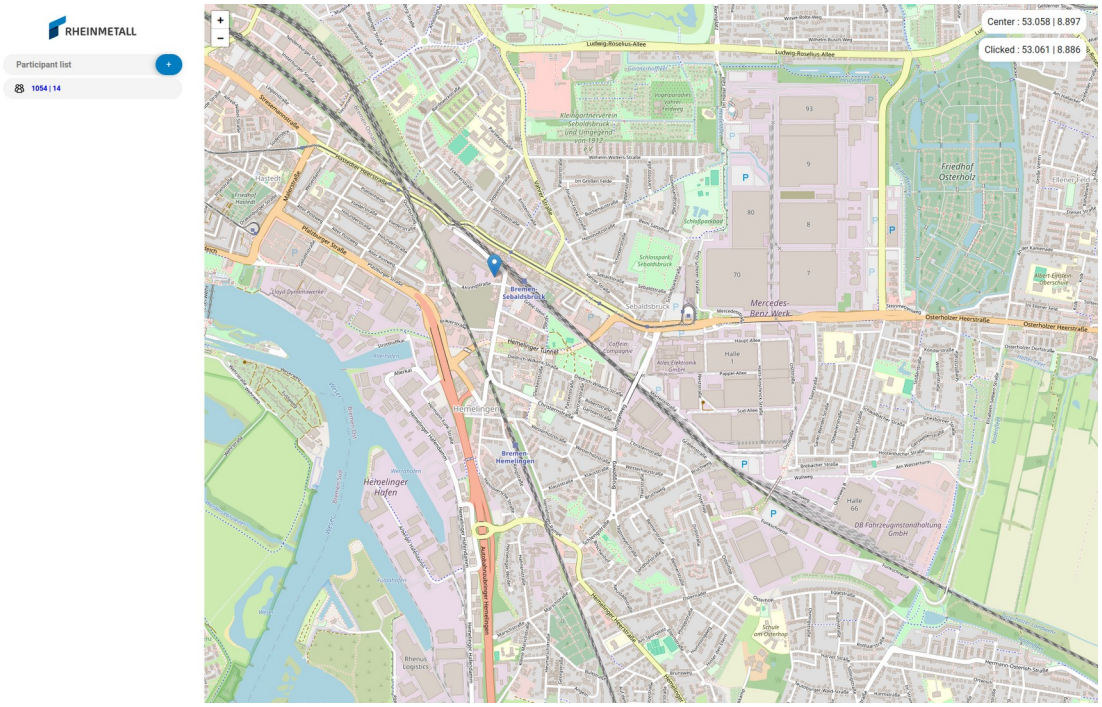
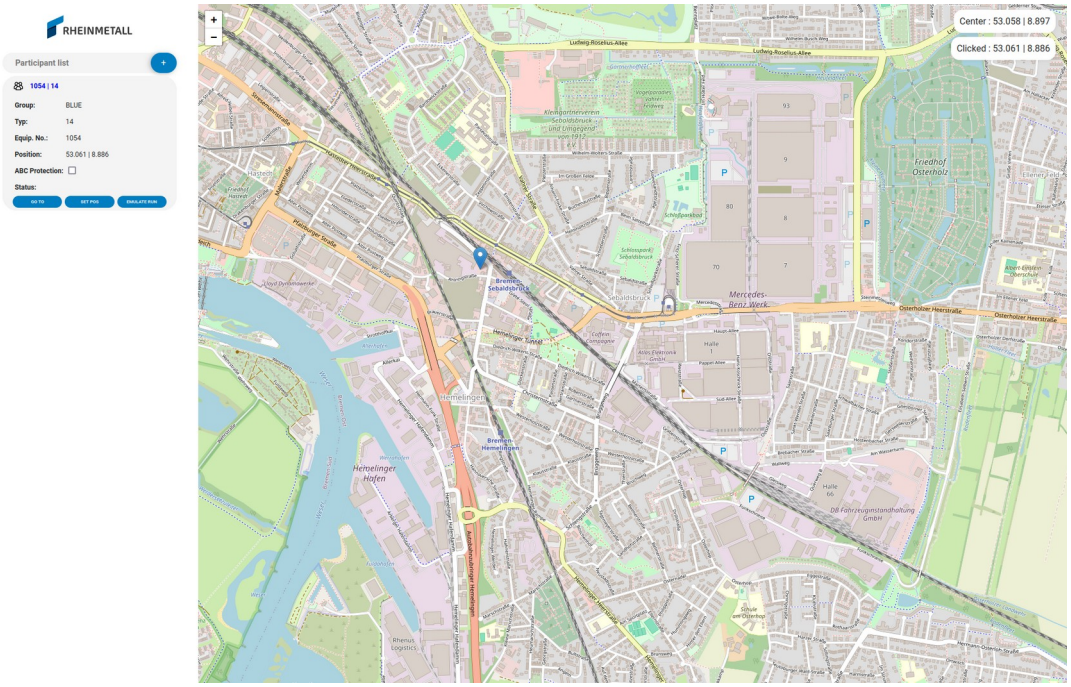
TNE Simulator



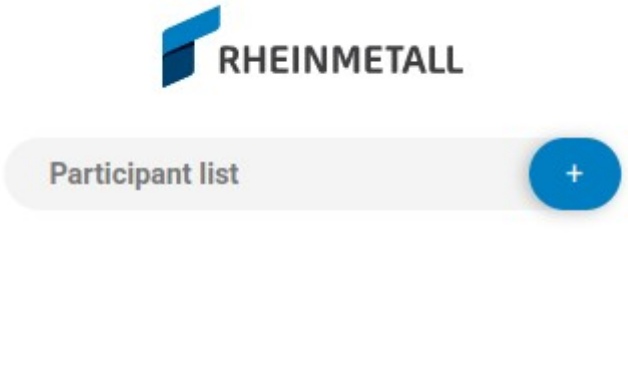
Die Oberfläche des TNE Simulators wurde in Reactjs geschrieben und zeigt visuell das Simulationsgebiet und die Positionen der Participants. Wenn man auf der linken Seite auf den Participant klickt, öffnet sich ein Drop-Down-Menü mit allen Informationen über den Participant sowie Buttons, die Funktionen wie Position setzen, zum Participant springen sowie eine Route setzen zur Verfügung stellen. Außerdem kann der Abc-Schutz des Teilnehmers ein- und ausgeschaltet werden. Das Backend ist in Golang geschrieben und dient als Schnittstelle zwischen dem Frontend und dem Legatus System.



Wie fügt man ein Participant hinzu ?

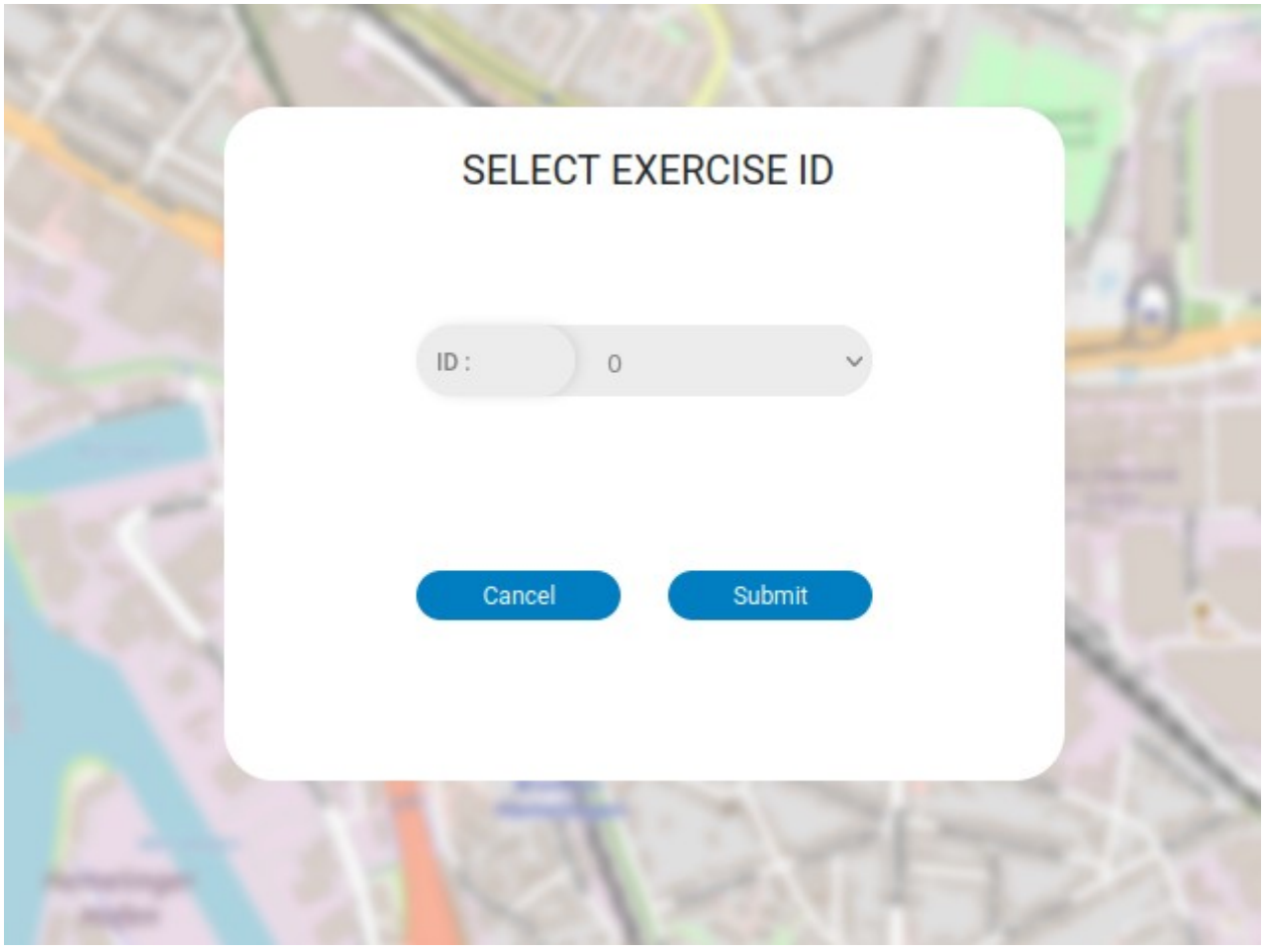
Auf der linken Seite der Anwendung befindet sich eine Sidebar mit einem Button, der ein Plus in sich trägt, wenn man darauf drückt, erscheint ein Popup, in dem man die gewünschte excerciseId auswählen kann, wenn man dann auf Weiter drückt, kommt man zu einem weiteren Popup, in dem man dann den gewünschten Participant hinzufügen kann, indem man die Party Color sowie die dazugehörige Equipmentnummer auswählt.

SideBar

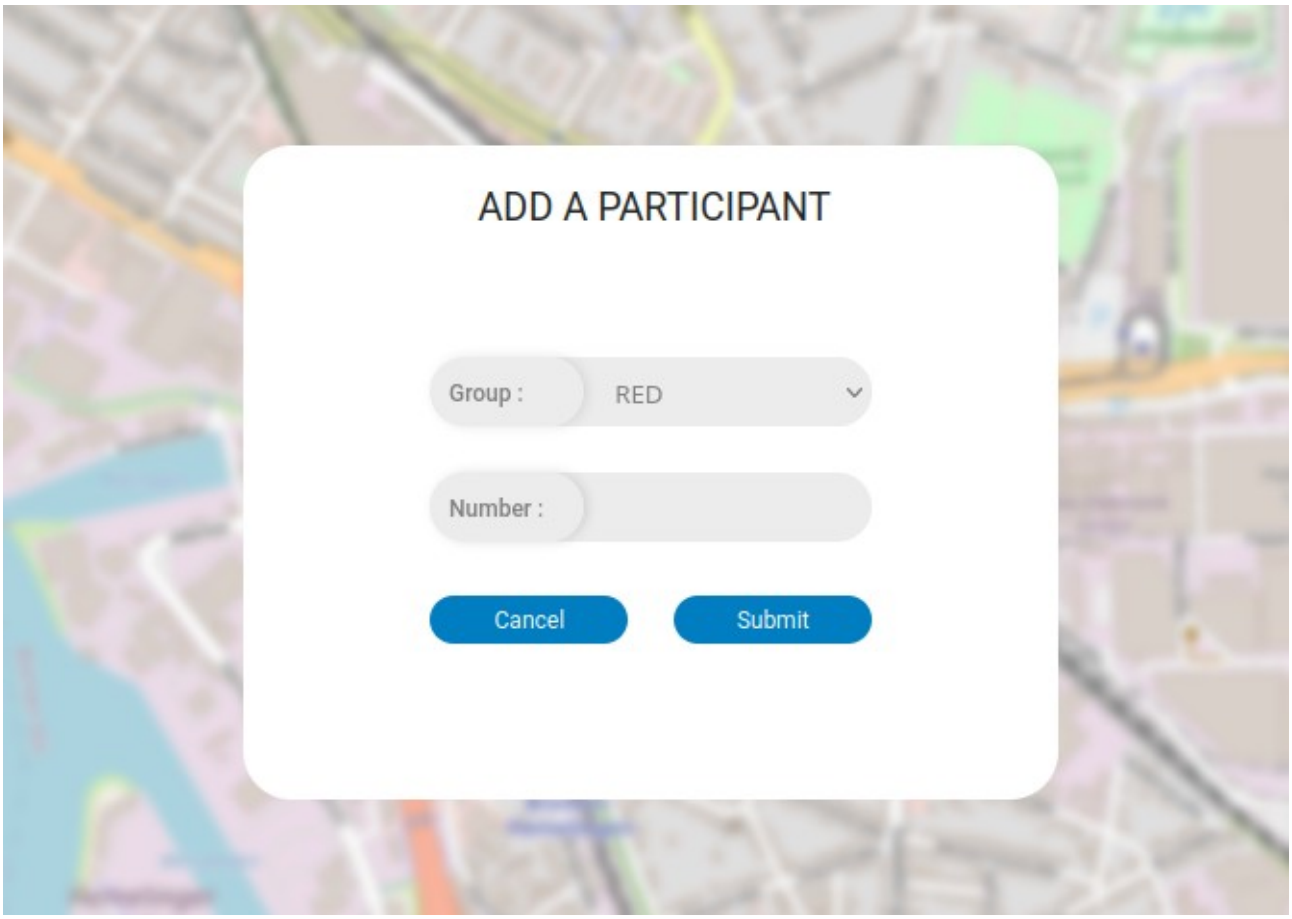




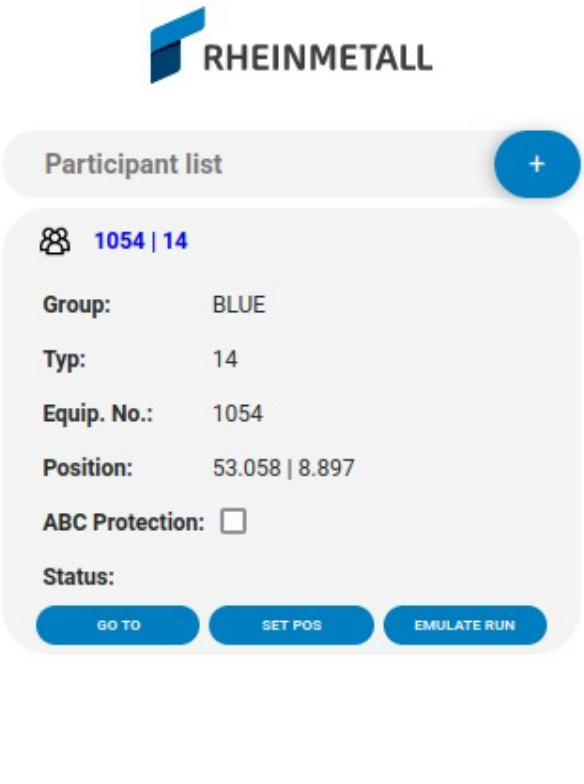
(1) PopUp



(2) Popup



SideBar nachdem ein Participant hinzugefügt wurde





Anforderungen

Das Projekt befindet sich im Svn.
Was muss installiert sein ? :

- Browser (egal welcher) wir haben Firefox 115.6.0esr verwendet
- Nodejs Version v20.13.1
- Golang Version v1.21.5
- Die Worksim und ein Szenario müssen vorhanden sein

Das Projekt kann auf dem eigenen Rechner erweitert, sowie benutzt werden.

Wenn man alles installiert hat und das Projekt erweitern möchte, sollte man bevor man irgendwas macht die Konsole öffnen und dann zum Pfad des Frontends / Backends springen und dort den Befehl Frontend : *npm i* / Backend : go get . ausführen. Nachdem man das gemacht hat kann man anfangen am Projekt zu arbeiten, denn dies installiert alle notwendigen Pakete.

Die Befehle npm i und go get . Benutzt man, um alle dependencies, die im Projekt vorhanden sind zu installieren, um dann am Projekt weiterentwickeln zu können. Wenn man das nicht machen würde, dann würde es beim „kompilieren“ sämtliche Fehler ausgeben, welche die fehlenden dependencies aufweisen.

Um das Projekt zu **debuggen** muss man diese Befehle ausführen.

Frontend :
Konsole öffnen → Pfad zum Frontend Quellcode → npm run start
Backend :
Konsole öffnen → Pfad zum Backend Quellcode → go run .

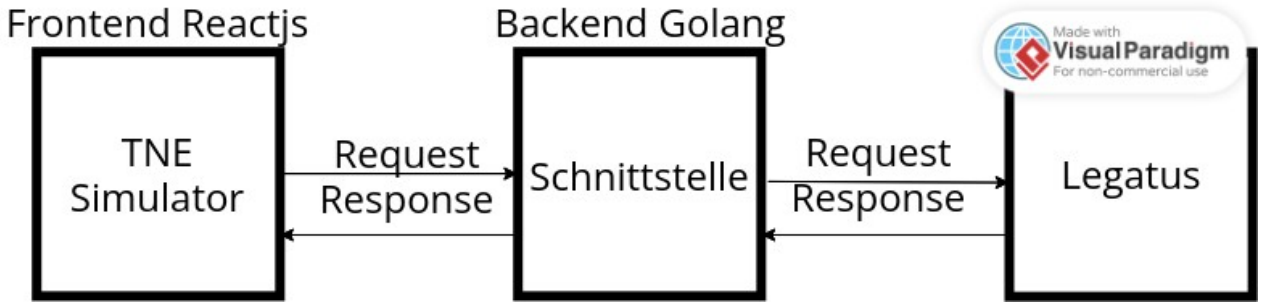
Um ein Build vom Projekt zu machen benutzt man folgende Befehle

Frontend :
(einmalig an jedem Rechner) Konsole öffnen → npm install --global serve

Konsole öffnen → Pfad zum Frontend Quellcode → npm run build → serve -s build
Backend :
Konsole öffnen → Pfad zum Backend Quellcode → go build . → entstehendes executable ausführen

Warum macht man das ?
Npm run start :
Dieser Befehl startet unser Frontend, um es zu Testen
Npm run build :
Wir verwenden im Frontend Javascript und React, damit der Computer das versteht ohne irgendwelche zusätzlichen Programme macht man ein production build, dies übersetzt das Projekt in Javascript und HTML, also Statische Dateien, welche gehasht sind.
Serve -s build :
Dieser Befehl geht im Projektpfad in den Order build und präsentiert die statischen Dateien im localhost auf Port 3000.
go run . :
startet die Schnittstelle zwischen dem Frontend und dem Legatus System

go build . :
kompiliert das Backend Programm zu einem Executable.



Der TNE Simulator funktioniert so, dass das Frontend eine Anfrage an die Schnittstelle stellt, dieser fragt dann das Legatus System, ob die Anfrage passend ist und wenn ja, dann schickt das Legatus System eine Antwort an die Schnittstelle, welche dann diese Antwort weiterleitet.

Websocket Telegramme als JSON :

/participantHandler
- Funktion checkIfParticipantIsValid
  - Request
    - type : `checkIfParticipantIsValid`
    - excerciseId : parseInt(excerciseId)
    - participantData: { “participantId“ : parseInt(equipmentNumber) }
  - Response
    - true || false
- Funktion changeAbcProtection
  - Request
    - type: 'changeAbcProtection',
    - excerciseId: parseInt(exerciseId),
    - participantData: { "participantId": parseInt(equipmentNumber), "abcProtection": abcProtection }
- Funktion sendPositionChange
  - Request
    - type: 'changePosition',
    - excerciseId: parseInt(exerciseId),
    - participantData: { "participantId": parseInt(equipmentNumber), "positions": [{"lat" : geoPosition[0], "lng": geoPosition[1]}] }
- Funktion getAllParticipants
  - Request
    - type: "getAllParticipants",
    - excerciseId: parseInt(excerciseID)
  - Response
    - Liste von allen Participants unter der ExcerciseID

/partyData
- Funktion getPartyData
  - Request
    - excerciseId: parseInt(exerciseId)
  - Response
    - Liste von allen Partys

/excerciseIds
    Funktion getExcerciseIds
        Response
            exerciseIds: []

Diese Telegramme sind alle aus der Sicht des Frontends, heißt also wenn Daten abgerufen werden wollen, wird das Telegramm erst zur Schnittstelle gesendet, wo dieser in einem Struct definiert wird, dieser Struct wird dann an das Legatus System weitergeschickt und verarbeitet zu einer Response.

Welches Telegramm wird wann verwendet ?

CheckIfParticipantIsValid wird beim Hinzufügen eines Participants verwendet, es wird geprüft ob die Party und die Equipmentnummer übereinstimmen, wenn dies der Fall ist wird ein Boolean zurückgegeben und jenachdem wird der Participant hinzugefügt oder nicht.

ChangeAbcProtection wird beim Ändern der Abc Protection ausgeführt, im Dropdown Menü haben wir ein Informationsfeld zur Abc Protection, dieses Feld enthält eine Checkbox, wenn diese angeklickt wird, dann wird das Telegramm gesendet und ändert den aktuellen Status zur Abc Protection.

SendPositionChange wird bei jeder Positionsänderung eines Participants ausgeführt.

GetAllParticipants liefert eine Liste aller Participants unter der entsprechenden ExcerciseId, diese werden dann zur Liste der Participants im Frontend hinzugefügt und erscheinen dann in der Sidebar.

Das GetPartyData Telegramm wird gesendet, wenn ein Participant hinzugefügt wird, dies geschieht bevor das (2) Popup erscheint.

Das getExcerciseIds Telegramm wird direkt beim Betreten des Frontends gesendet, da die Liste der ExcerciseIds sehr wichtig ist, um alle anderen Telegramme senden zu können.