**Technische Dokumentation Red:Wire**

Über Red:Wire

Red:Wire ist eine Requirements Engineering Webapplikation. Mit ihr ist es möglich, Anforderungen nach der Rupp-Schablone strukturiert aufzunehmen und mit ihnen zu arbeiten. Ein User- sowie Teamsystem erlaubt kollaboratives Arbeiten in Gruppen. Weiterhin existiert ein Administrator, welcher User und Teams aus der Software entfernen kann.

Verwendete Technologien

* Apache 2 Webserver
* MySQL Datenbank
* Socket.io Server
* Node.js

* HTML
* CSS
* SQL
* Javascript / jQuery

Entwickelt auf:

* Windows 7 64 Bit
* XAMPP for Windows

Entwickelt mit:

* Notepad++
* Mozilla Firefox / Google Chrome (Desktop View)
* Samsung Galaxy S3 / Google Nexus 5 (Mobile View)

Verzeichnisstruktur:

* Red:Wire Client
  + bootstrap
    - [Bootstrap Stuff]
  + js
    - adminpageScript.js
    - dashboardScript.js
    - jquery-1.11.3.min.js
    - table2excel.js
    - translate.js
  + css
    - dashboard.css
    - login.css
  + img
    - de
    - golf
    - header
    - logo
    - uk
  + adminpage.html
  + index.html
  + dashboard\_de.html
  + install
    - index.html
    - email.html
    - database.html
    - admin.html
    - summary.html
* Red:Wire Server
  + conf
    - config.json
  + log
    - track/date/
    - errorlog/date/
  + node\_modules
    - express
    - js-md5
    - fs
    - mysql
    - nconf
    - nodemailer
    - socket.io
  + app.js
  + package.json

Ungefährer Zeitaufwand Konzept + Coding + Dokumentation:

* ca. 330 Stunden

Features von Red:Wire:

* Anforderungen mittels Rupp-Schablone aufnehmen
* User registrieren
* User einloggen
* Team erstellen
* User zu Team hinzufügen
* User aus Team entfernen
* Userinformationen der Team-User anzeigen
* Profildaten editieren
* Newsfeed
* Kategorien erstellen
* Anforderungen bearbeiten
* Anforderungen löschen
* Kategorien bearbeiten
* Kategorien löschen
* Anforderungen sortieren
* Anforderungen herunterladen (.xlsx)
* Anforderungen herunterladen (.csv)
* Logout
* Anforderungen durchsuchen
* Tooltips einschalten
* Sprachführung ändern

Funktionsweise der Software:



Der Browser lädt die Red:Wire Applikation in seinen Speicher. Die Page wird im Browserfenster dargestellt und alle nötigen Javascript sowie css Dateien werden ausgeführt.

Der Browser verbindet sich mit dem Socketserver auf <IP>:3000. Nachdem die Seite vom Browser gespeichert wurde, kann sie nahezu ohne Webserverkommunikation auskommen (es wird lediglich die Einbindung der jQuery-UI Bibliothek durchgeführt).

Jede Kommunikation geschieht ab jetzt über den Socketserver. Dieser fungiert abstrahiert als Schnittstelle zur MySQL Datenbank. Der Socketserver besitzt für alle möglichen Anfragen korrespondierende Listener, welche einkommende Daten verarbeiten und eine Antwort zurücksenden. Beispielsweise wird ein Datensatz aus der Datenbank abgefragt.

Socketserver

Client

Datenbank

Allgemeine Informationen

Generell gilt, dass Funktionen nur zur Verarbeitung von vorhandenen Informationen genutzt werden (Auswertung Return Values, Berechnung von etwas,…) und zum Absetzen von Requests (socket.emit) an den Socketserver. Alle hier gelisteten Funktionen haben nativ keinen Zugriff auf die Datenbank/-basis und müssen alle nötigen Daten per Emit vom Socketserver anfordern, bzw. sind teilweise nur dazu gebraucht, Emits abzusetzen. Weiterhin werden die Funktionen genutzt, um Daten vom Nutzer einzulesen und an den Socketserver zwecks Speicherung in der Datenbank zu senden. Listener in den Clientseitigen .js-Dateien sind für den sofortigen Empfang von Serverdaten genutzt, welche dann meistens formatiert und dem User auf der grafischen Oberfläche präsentiert werden. Als einzige Datei, hat „app.js“ (Socketserver) Zugriff auf die Datenbank mittels eingebundenem MySQL Modul.

dashboardScript.js

Zuständig für: Login-, Dashboard-Funktionen  
Zeilen: 1747

Die dashboardScript.js beinhaltet alle Funktionen und Socket-Emits bzw. –Listener, welche die index.html und dashboard\_de.html benötigen.

Erwähnenswerte Variablen:

feedEmpty – true; sobald Newsfeed befüllt wird 🡪 startet clearFeed()

toggle – toggelt die Anforderungssortierfunktionen

sessionID – sessionID des verbundenen Users. Wird zur Unterscheidung gespeichert, falls sich User mehrmals mit gleichem Account einloggt. Ist ein Timestamp.

oldActive – wird genutzt, um ausgewählte Menüpunkte hervorzuheben.

Funktionen:

* setBlock() / isBlocked() – setzt / liefert „Blocked“-Status der Session. Falls true, wird bei Notifications eine DOM-Manipulation übersprungen (z.B. Aktualisierung der Anforderungen, während User Anforderung erstellt)
* switchToDE() / switchToEN() – setzt die Stringvariablen aus translate.js neu. Bei Laden von neuem Content werden die Variablen für die Beschriftung von Objekten / Feldern genutzt. Außerdem wird per Javascript DOM-Manipulation der Menüeinträge die Sprache sofort angepasst.
* checkCredentials() – liest Logindaten der index.php Eingabefelder aus und sendet sie an den Socketserver.
* registerUser() – liest eingegebene Daten aus dem Registrierungs-Modal und sendet sie an den Socketserver. Leert danach die Eingabefelder.
* insertIntoFeed(mess) – fügt mess an den Feed an. Startet das Intervall von clearFeed();
* changeData() – liest Eingaben aus Profil-Modal aus und schickt sie an den Server.
* checkRequirement() – validiert Eingaben, welche zum Erstellen einer Anforderung eingegeben wurden. Gibt true zurück, wenn Eingaben ok.
* fieldError() – gibt Fehlermeldung aus
* insertReq(origin ) –liest Eingabefelder aus, je nach Ort des Aufrufes (origin), wird eine bestimmte Aktion vom Server ausgeführt. Bsp.: Erstellen wird anders behandelt, als Editieren (Löschen + Einfügen)
* function deleteReq(id, doAfterThis, param) – Löscht eine Anforderung (gibt Befehl an Server). Beim Editieren wird hier als doAfterThis insertReq() übergeben.
* function getRequirements(query) –Fragt bei Server nach Anforderungen an
* function sortById(arr) – Sortiert Array nach ID. Je nach reversedID toggelt die Funktion aufwärts/abwärts.
* function sortByReq(arr) – Sortiert Array nach Anforderungsstring
* function sortByPrio(arr) – Sortiert Array nach Priorität
* function sortByStatus(arr) – Sortiert Array nach Statusstring
* function sortByTime(arr) – Sortiert Array nach Erstellungsdatum
* function timeConverter(UNIX\_timestamp) – Konvertiert Timestamp in TT – MM- YYYY Format. Liefert String zurück
* function setTable(requirements) – Füllt Contentbereich mit Anforderungstabelle
* function edit(id) – löscht Anforderung und fügt eine neue Anforderung ein (als bearbeitet)
* function confirmRemoval(reqID) – öffnet Dialogfenster, ob wirklich gelöscht werden soll
* function refreshExport(arr) – erneuert Downloaddatei mit aktuell dargestellten Anforderungen
* function chooseDownload(reqID) – öffnet Dialogfenster, als welches Format heruntergeladen werden soll
* function enableTooltips() – setzt Titelattribute für Hover-Tooltips
* function helpEnabled() – prüft, ob Tooltip Checkbox markiert ist
* function checkHelpEnabled() – wenn helpEnable(), dann aktiviere Tooltips
* function sizeAccordion() – passt Größe der Teammenü-Bereiche an
* function createReqForm() – erstellt Anforderungsformular im Contentbereich
* function createEditForm(id) – erstellt Bearbeitungsformular im Contenbereich
* function createCategory() – erstellt Kategorieformular im Contentbereich
* function submitCategory() – liest Kategorieeingabe aus und sendet sie an den Server
* function getCategories(user,act) – sendet Kategorieanfrage an den Server
* function deleteCat(id, username) – öffnet Dialogfenster, ob Kategorie wirklich gelöscht werden soll
* function editCat(id) – fragt Kategorieinformationen zur Bearbeitung ab
* function submitEdit(id) – liest neuen Kategorienamen aus Eingabefeld aus und sendet ihn an den Server
* function loadTeamOptions() – liest Teamdaten aus und passt Teammenügröße an.
* function createTeam() –liest Teamnamen aus Eingabefeld aus und sendet ihn an den Server
* function getMyGroups() – sendet Anfrage an den Server, dass Teams des Users zurückgesendet werden sollen
* function leaveTeam() – sendet Anfrage an Server, dass User Team verlässt
* function intoTeam(team\_id) – Sendet Anfrage an Server, dass User Team mit der ID team\_id beitreten will
* confirmTeamRemoval(team\_id) – öffnet Dialogfenster, ob Team wirklich gelöscht werden soll
* function deleteTeam(team\_id) – sendet Anfrage an Server, dass Team mit der ID team\_id gelöscht werden soll.
* function refreshTeamDropdown() – erneuert auswählbare Teams auf Dropdown im „Mitglieder hinzufügen“ Reiter des Teammenüs
* function refreshTeamData(opt) – ruft getMyGroups und refreshTeamDropdown() auf. Wenn opt == 1, werden danach die Anforderungen angefordert.
* function addTeamMember() – liest Membername aus Eingabefeld aus und sendet Anfrage an Server, dass <Membername> zu ausgewähltem Team aus Team-Dropdown hinzugefügt werden soll.
* function deleteUserFromTeam(userID,teamID) – sendet Anfrage an Server, dass User (userID) aus Team (teamID) gelöscht werden soll.
* function editTeam(teamID) – sendet Anfrage an Seerver, dass Teammitglieder zurückgesendet werden sollen
* function showUserData(userID) – sendet Anfrage an Server, dass Userinformationen von User mit userID angezeigt werden sollen.
* function getParameter(param) – extrahiert Parameter (param) aus der URL.
* function createUser(name) – bei Login wird User in Local Storage gespeichert und eine SessionID aus dem aktuellen Timestamp generiert.
* function getUserName() – liest und returnt den Namen des eingeloggten Users aus dem Localstorage
* function redirectToDashboard() – Leitet zum Dashboard weiter. Wenn username == admin, leitet zur Adminpage weiter
* function logOut() – löscht User aus dem Localstorage und leitet auf index.html zurück
* function install() – schreibt eingetragene Werte für Mailkonfiguration in Array und übergibt das Array an den Socketserver
* function skipInstall() – überspringt Mailkonfiguration
* function sendTestEmail() – sendet Signal an Socketserver, um Email mit temporärer Konfiguration aus Eingabefeldern zu senden (zum Testen der Konfig vor dem Speichern)
* function testDatabaseConnection() – testet Datenbankverbindung mit temporären Werten aus Eingabefeldern
* function writeDatabaseConfiguration() – übergibt Datenbankkonfiguration an Socketserver
* function checkAdminConfiguration() – überprüft Validität des Administratorpassworts (Install Assistant)
* function showSummary() – fragt Server nach Ausgabe aller gespeicherten Konfigurationswerte von Datenbank und Mail
* function configureRedWire() – meldet Server, dass alle Konfigurationsdaten korrekt sind und abgespeichert werden können

Listener:

* socket.on('connect' – setzt Client-Socket-ID
* socket.on('requestUsernameLogin' – sendet Username an Server
* socket.on('disconnected' – zeigt Meldung und leitet zur index.html zurück
* socket.on('requestUsernameLogOut' - /
* socket.on('newReq' – Bestätigung, dass Anforderung eingetragen wurde. Schreibt Meldung in News-Feed
* socket.on('reqFail' – Fehler beim Eintragen der Anforderung
* socket.on('requestUsers' – für Debugzwecke. Sendet Aufforderung an Server, das User-Array anzuzeigen (im Serverfenster)
* socket.on('editReq' - Bestätigung, dass Anforderung edititert wurde. Schreibt Meldung in News-Feed
* socket.on('deleteReq' - Bestätigung, dass Anforderung gelöscht wurde. Schreibt Meldung in News-Feed
* socket.on('getRequirements' – Annahme der angefragten Anforderungen
* socket.on('fillCategorySelector' – Füllt Kategorie-Dropdown auf dem Dashboard
* socket.on('youBeenAdded' – Schreibt Meldung in Newsfeed
* socket.on('changeData' – wertet Server Returncode der Profildatenänderung aus und generiert Meldung
* socket.on('registerUser' – wertet Returncode der Userregistration des Servers aus und zeigt Meldung an
* socket.on('checkLogin' – prüft Returncode des Userlogins. Wenn Daten korrekt, Weiterleitung an Dashboard / Adminpage
* socket.on('getCatID' – Nimmt eingehende Anforderung des Servers und leitet diese an Server zurück. (Im Schritt davor wurde die Kategorie-ID angefordert. Sie ist nun dem data-Array hinzugefügt worden. Mit dem neu entstandenen Array wird nun die Anforderung erstellt)
* socket.on('FillCategoryDropdown' – Füllt Kategorie-Dropdown bei der Anforderungserstellung
* socket.on('doEditReq' – Erstellt Editierformular im Content-Bereich
* socket.on('submitCategory' – wertet Returncode des Kategorie-Inserts aus und schreibt Meldung in News-Feed
* socket.on('getCategories' – listet Kategorien in der Kategorie-Ansicht (Menüpunkt „Kategorie“) auf
* socket.on('deleteCat' – wertet Returncode des Kategorie-Deletes aus und schreibt Meldung in News-FEED
* socket.on('getEditData' – Setze erhaltene Werte in Kategorieformular ein, damit diese bearbeitet werden können
* socket.on('submitEdit' – nach dem erfolgreichen Bearbeiten der Kategorie wird wieder das Kategorieerstellformular angezeigt.
* socket.on('createTeam' - Wertet Returncode der Teamerstellung aus und schreibt Meldung in News-Feed
* socket.on('createDefaultCategory' – schreibt Meldung in News-Feed
* socket.on('insertTeamOwner' - Wertet Returncode des Teambesitzer-Inserts aus und schreibt Meldung in News-Feed
* socket.on('addTeamMember' – Wertet Returncode des Mitglied-Adds aus und schreibt Meldung in News-Feed
* socket.on('teamChanged' – nach Teamwechsel wird eine Meldung in den News-Feed geschrieben.
* socket.on('getMyGroups' – nimmt erhaltene Teams (Groups) und erneuert die Ansicht des Team-Modals
* socket.on('leaveTeam' - Wertet Returncode des Teamverlassens aus und schreibt Meldung in News-Feed
* socket.on('intoTeam' – Wertet Returncode des Teambeitretens aus und schreibt Meldung in News-Feed
* socket.on('deleteTeam'
* socket.on('getTeamDropdown' – nimmt erhaltene Teams und erstellt Team-Dropdown im
* socket.on('deleteUserFromTeam' - Wertet Returncode der Mitgliedsentfernung aus und schreibt Meldung in News-Feed
* socket.on('getMembers' – nimmt erhaltene Mitgliederdaten und erstellt Mitglieder-Ansicht im Team-Modal
* socket.on('getUserInfos' – nimmt erhaltene Userinfos und erstellt Userinfo Ansicht im Userinfo-Team-Modal
* socket.on('testMail' – meldet Sendeversuchsergebnis über GUI-Meldung an User
* socket.on("redirectYes" – not working!!
* socket.on("redirectNo" – not working!!
* socket.on('testDatabaseConnection' – meldet Rückgabewert der Datenbankverbindung an User zurück
* socket.on('writeDatabaseConfiguration' – meldet, dass Übergabe der Datenbankkonfiguration erfolgreich war
* socket.on('returnToIndex' – leitet vom Install Assistant zur Loginpage zurück

adminpageScript.js

Zuständig für: Administrator Backend Funktionen  
Zeilen: 295

Funktionen:

* function changeData() – liest Daten aus Eingabefeldern ein und sendet Request an Server, diese Profildaten in die Datenbank zu schreiben
* function logOut() – leitet zur index.html zurück
* function confirmUserRemoval(username,userID) – öffnet Dialogfenster, zum Bestätigen oder Abbrechen des Löschvorganges eines Users
* function deleteTeam(user,team\_id,userID) – sendet Request an Server, um ein Team zu löschen
* function confirmTeamRemoval(user,team\_id,userID) – öffnet Dialogfenster, zum Bestätigen oder Abbrechen des Löschvorganges eines Teams
* function displayKickContent(username,userid) – listet alle Teams, falls vorhanden, eines zu löschenden Users auf.
* function forceDeleteUser(userID) – lösche User, lose alle Abhängigkeiten in der Datenbank auf
* function getUsers(query) – Sende Request nach User an Server (mit Query)
* function confirmNewOwner(teamID,oldOwner,oldOwnerID) – setze neuen Teambesitzer (optional, anstatt das Team eines Users zu löschen) | Request an Server
* function searchUsers() – sendet Request mit Searchquery für Usernames an Server

Listener:

* socket.on('changeData' – wertet Returncode des Servers zur Profildatenänderung aus
* socket.on('deleteAllUsersFromTeam' – alle User aus Team gelöscht, sende nun Request zum Löschen des Teams
* socket.on('getMyGroups' – bekommt und verarbeitet angeforderte Teams eines zu löschenden Users.
* socket.on('forceDeleteUser' – aktualisiere Userliste nach dem Löschen eines Users
* socket.on('getAllUsers' – listet alle angeforderten User im GUI auf
* socket.on('newOwner' – aktualisiert die Teams des zu löschenden Users nach dem Übergeben eines Teams

translate.js

Zuständig für: Variablen für die Übersetzung Deutsch/Englisch  
Zeilen: 787

app.js

Zuständig für: Node.js Socketserver  
Zeilen: 1624

Funktionen:

* function getCurTime() – Returnwert ist aktuelle Zeit
* function logerr(mess) – schreibt mess in Errorlog
* function sendMail(error) – schickt Mail mit Errormeldung
* function traceLog(mess) – schreibt mess in Tracelog
* function isInArray(sock,arr) - prüft, ob Socket sock in Array arr ist, gibt true/false zurück
* function addUserToArray(username,sock) – schreibt Username und Usersocket in Userarray (derzeit eingeloggte User)
* function deleteUserFromArray(socketid,arr) – löscht bei Logout User aus dem Userarray
* function openConfig() – öffnet Configfile (schreibt newline)
* function writeConfig(attr,val) – schreibt Attribut und Wert in Configfile
* function closeConfig() – schließt Configfile und schreibt random Value hinein, da durch asynchrone Zugriffe das „}“ Zeichen möglicherweise an falscher Stelle erscheint und die .json fehlerhaft ist.
* function notifyNewMember(member) – schickt emit an bestimmten User, dass er einem Team zugefügt wurde
* function initInput() – überprüft, ob eingetragene Werte != NULL
* function checkEmailValid(email) – validiert Email
* function checkPassLength(pw,pw2) – validiert Passwortlänge
* function checkPasswordsEqual(password,password2) – überprüft, ob Passwörter gleich sind
* function changeData(data) – schreibt neue Profildaten in Datenbank
* function registerUser(data) – schreibe neuen User in Datenbank

Listener:

* socket.on('sendEmail', - sende Testemail
* socket.on('configureRedWire' – schreibe temporäre Install Assistant Arraydaten in ConfigFile
* socket.on("writeAdminConfiguration" – schreibe Adminpasswort in temporäres Array
* socket.on('showSummary' – zeige Zusammenfassung der Konfiguration an
* socket.on('testDatabaseConnection' – teste Datenbankconnection mit temporären Arraydaten
* socket.on('writeDatabaseConfiguration' – schreibe temporäre Feldeingaben für Datenbank in Array
* socket.on('reloadConf' – lade ConfigFiledaten in den Server
* socket.on('testMail' – teste Mailversand mittels Daten aus Eingabefeld
* socket.on('enableReports' – schreibe Mailkonfiguration in temporäres Array
* socket.on('disableReports' – überspringe Mailkonfiguration
* socket.on('receiveUsernameLogin' – führt addUserToArray() aus
* socket.on('disconnect', - löscht User aus Userarray
* socket.on('showUsers' – Debugfunktion, schreibt alle eingeloggten User aus Userarray in Konsolenfenster des Servers
* socket.on('newReq', - trägt neue Anforderung ein
* socket.on('doEditReq' – hole Anforderung aus Datenbank für editForm
* socket.on('editReq' – Benachrichtige alle Teammitglieder, dass Anforderung bearbeitet wurde
* socket.on('getRequirements' – hole alle Anforderungen aus der Datenbank
* socket.on('getCatID' – hole Kategorie ID
* socket.on('insertReq' – trage Anforderung in Datenbank ein
* socket.on('createTeam' – schreibe neues Team in Datenbank (wenn noch nicht existent)
* socket.on('createDefaultCategory' – erstelle „uncategorized“-Kategorie (nach Teamerstellung)
* socket.on('insertTeamOwner' – schreibe User als Teambesiter in Datenbank
* socket.on('addTeamMember' – füge Team einen User hinzu (ändere dessen Teamid in der Datenbank)
* socket.on('registerUser' – schreibe neuen User in Datenbank
* socket.on('changeData' – ändere Profildaten in Datenbank
* socket.on('checkLogin'- überprüfe bei Login, ob Nutzer existiert und Passwort ok
* socket.on('teamChanged' – ändere Teamid von User in Userarray
* socket.on('leaveTeam' – lösche Teamid von User aus Userarray
* socket.on('getTeamDropdown' – hole alle Userteams aus Datenbank für Teamdropdown (Team Menü)
* socket.on('intoTeam' – ändere Teamid des Users in Datenbank
* socket.on('getUserInfos' – hole Userinformationen für Informations-Modal im Team Bearbeiten Menü
* socket.on('deleteUserFromTeam' – lösche User aus Team (Datenbank und setze Teamid in Userarray = null)
* socket.on('deleteTeam' – lösche Team aus Datenbank
* socket.on('getMembers' – hole alle Teammitglieder eines Teams aus der Datenbank
* socket.on('getAllUsers' – hole alle registrierten User aus der Datenbank
* socket.on('forceDeleteUser' – lösche User, ignoriere (löse) eventuelle Abhängigkeiten
* socket.on('getMyGroups' – hole alle Teams eines Users (für eventuelles Übertragen oder Löschen)
* socket.on('getGroupsToDelete' - hole alle Teams eines Users (für eventuelles Übertragen oder Löschen)
* socket.on('deleteAllUsersFromTeam' – lösche alle User aus dem Team (bei Löschen eines Teams von einem zu löschenden User)
* socket.on('submitCategory' – schreibe neue Kategorie in Datenbank
* socket.on('getCategories' – hole alle Kategorien aus der Datenbank
* socket.on('deleteCat' – lösche Kategorie aus Datenbank
* socket.on('getEditData' – hole Namen einer Kategorie aus der Datenbank (für edit)
* socket.on('submitEdit' – benenne Kategorie um