Kalender

Programmieren II – Go

von

9495107, 4706893, 9608900

Inhalt

1. Architekturdokumentation

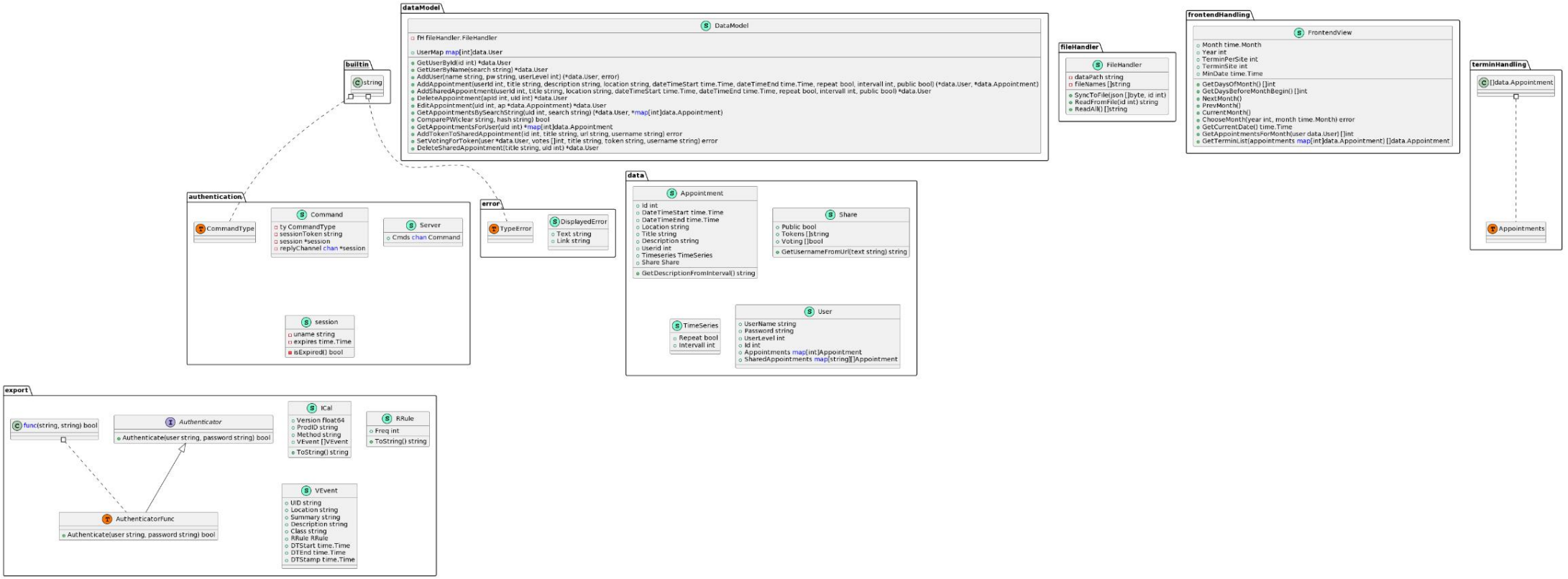

Abbildung 1: UML

Bei der Architektur der Anwendung wurde bereits während der Planung auf eine hohe Modularität und Kapselung geachtet, um so durch Anwendung von TDD einzelne Packages einfach und komfortabel als Blackbox testen zu können und somit parallel an verschiedenen Bereichen der Anwendung gleichzeitig zu arbeiten.

Packages

data

Im Package data sind die Types für Datenstrukturen für die Termine (Appointment) und Benutzer (User) definiert, die genutzt werden, um Daten aus JSON-basierten String direkt in structs mappen zu können.

Die beiden Unterstrukturen Share und TimeSeries sind in die Struktur Appointment integriert, alle weiteren Eigenschaften, inkl. Ihres Datentyps sind in Abbildung 1: UML zu finden. Es besteht die Beziehung User 1 – n Appointment.

Die Unterstruktur Share enthält die Methode GetUsernameFromUrl, die anhand einer Share-URL den Usernamen ermitteln kann.

Ein Appointment enthält weiterhin die Methode GetDescriptionFromIntervall, die die Integer-Eigenschaft Intervall einer TimeSeries in den zugehörigen Text mappt.

Alle in diesem Package spezifizierten Typen werden durch einen Constructor instanziiert.

fileHandler

Das Package fileHandler ist für alle Dateibasierten Operationen (CRUD) zuständig und verfügt über den Type FileHandler.

Ein FileHandler schreibt nur in einen Ordner, der als fester Pfad bei der Instanziierung übergeben und in dataPath gespeichert wird. Besteht der Zielpfad nicht bereits, so wird er durch den FileHandler angelegt.

Alle in diesem Pfad erstellten Dateien werden mit Ihren Namen in der slice fileNames gesammelt.

Die Methode SyncToFile schreibt ein übergebenes Byte-Array in die korrekte Datei. Diese wird durch die ebenfalls übergebene User-ID ermittelt. Das Schema der Benennung ist über die UserID geregelt: Die Informationen des Users mit der ID 1 werden in der Datei „1.json“ gespeichert. Pro User wird eine Datei angelegt. Die Methode SyncToFile kann entweder eine neue Datei anlegen oder Informationen in einer bereits bestehenden überschreiben. Nach Anlegen einer neuen Datei wird diese in die slice der fileNames aufgenommen.

Die Methode ReadFromFile ließt anhand einer User-ID eine Datei ein und gibt die darin befindlichen Informationen als string zurück.
Die Methode ReadAll ließt anhand der slice fileNames alle Dateien ein und gibt eine slice mit deren Informationen als string zurück.
Alle in diesem Package spezifizierten Typen werden durch einen Constructor instanziiert.

dataModel

Das Package dataModel dient der gesamtheitlichen Verwaltung der gespeicherten Daten und dem Cachen der Informationen. Dazu verfügt es über den Type DataModel.
Ein DataModel verfügt über einen FileHandler und eine Map, die User und deren ID direkt bereithält, um einen schnellen Zugriff zu gewährleisten.
Die Methode GetUserByID gibt einen Pointer auf einen User anhand der zugehörigen ID zurück.
Die Methode GetUserByName gibt einen Pointer auf einen User anhand des zugehörigen Benutzernamens zurück.
Die Methode AddUser erstellt anhand gegebener Informationen einen neuen User. Das Passwort wird hierbei direkt in einen Salted Hash gewandelt und gibt einen Pointer auf den neuen User zurück.
Die Methode AddAppointment erstellt anhand gegebener Informationen ein neues Appointment, das einem User unter Appointments zugeordnet wird und gibt einen Pointer auf den User und einen Pointer auf das erstellte Appointment zurück.
Die Methode AddSharedAppointment erstellt anhand gegebener Informationen ein neues Appointment, das einem User unter Shared zugeordnet wird und gibt einen Pointer auf den User zurück.
Die Methode DeleteAppointment löscht anhand gegebener Informationen ein Appointment und gibt einen Pointer auf den User zurück.
Die Methode EditAppointment überschreibt ein bereits vorhandenes Appointment und gibt einen Pointer auf den User zurück.
Die Methode GetAppointmentBySearchString durchsucht die Felder Title und Description der Appointments eines Users und gibt einen Pointer Map mit den zugehörigen Treffern zurück.
Die Methode ComparePW vergleicht einen Klartext String mit einem gehashten String und gibt einen booleschen Wert als Ergebnis zurück.

Die Methode GetAppointmentsForUser gibt anhand einer User-ID alle zugehörigen Appointments zurück.
Die Methode AddTokenToSharedAppointment erstellt anhand der gegebenen Informationen einen Token und fügt diesen einem SharedAppointment hinzu. Im Fehlerfall wird ein error zurück gegeben.
Die Methode SetVotingForToken setzt ein neues Voting für einen Token eines SharedAppointments und gibt im Fehlerfall einen error zurück.
Die Methode DeleteSharedAppointment löscht ein SharedAppointment und gibt einen Pointer auf den zugehörigen User zurück.
Alle in diesem Package spezifizierten Typen werden durch einen Constructor instanziiert.

export

Das Package export beinhaltet alles, was für den Export in das ICal-Format benötigt wird. Dazu verfügt das Package über die Types ICal, VEvent, RRule, AuthenticatorFunc und das Interface Authenticator.
Bei VEvent und RRule handelt es sich um Unterstrukturen, die in ein ICal eingegliedert werden. RRule verfügt über die Methode ToString, die die Informationen in einer RRule in einen String wandelt, wobei das richtige Format nach RFC5545 eingehalten werden muss.
Ein ICal repräsentiert alle Informationen, die aus einem Appointment benötigt werden, um einen ICalendar nach RFC5545 zu erstellen.
Die Methode ToString wandelt ein gesamtes ICal in einen String nach RFC5545.
Das Interface Authenticator stellt eine generalisierte Methode dar, die von der AuthenticatorFunc aufgegriffen wird. Diese wird für Basic Auth genutzt, um ein Authentifizierung zu ermöglichen und gibt dazu für einen user und ein Passwort einen booleschen Wert als Ergebnis zurück.

error

Das Package error ist für eigene Fehler zuständig, die im Frontend angezeigt werden sollen und enthält den Type Displayed Error.
Dieser enthält jeweils einen String für die Fehlermeldung und für den Link zur vorherigen Seite, um die Navigation zurück für den User jederzeit zu ermöglichen.

authentication

Das Package authentication enthält den LoginHandler und den LogoutHandler, die für die Authentication eines Users über das Frontend zuständig sind.

Das Package enthält die Types Command, Server und session. Ein Command enthält spezifische Informationen, welche Session wie verarbeitet werden soll und in welchen Channel die Antwort gesendet werden soll. Ein Server enthält einen Channel aus Commands. Eine session enthält ihren Namen und ihren Endzeitpunk (Expiry).
Die zugehörige Methode isExpired gibt dazu einen booleschen Wert über die Gültigkeit der Session zurück.

frontendHandling

Das Package frontendHandling enthält und verwaltet aktuelle Daten und Informationen des Frontends. Dazu enthält es den Type FrontendView.
Die Methode GetDaysOfMonth gibt eine slice aus Integerwerten zurück, die die Monatstage enthält.
Die Methode GetDaysBeforeMonthBegin errechnet die restlichen Tage einer Woche bi zum Beginn eines neuen Monats und gibt diese als slice aus Intergerwerten zurück.
Die Methode NextMethode addiert zur FrontendView einen Monat hinzu.
Die Methode PrevMethode subtrahiert einen Monat von der FrontendView.
Die Methode CurrentMonth setzt den Monat und das Jahr der FrontendView auf die aktuellen Werte.
Die Methode ChooseMonth setzt den Monat und das Jahr der FrontendView auf einen übergebenen Wunschwert, bei fehlerhaften Parametern wird ein error zurückgegeben.
Die Methode GetCurrentDate gibt das aktuelle Datum zurück.
Die Methode GetAppointmentsForMonth gibt anhand eines Users die zugehörigen Terminzahl pro Tag zurück.
Die Methode GetTerminList ermittelt anhand der Map aus Appointments eines Users die Termine, die noch stattfinden werden und gibt diese als slice zurück.

terminHandling

Das Package terminHandling enthält die einzelnen Handler, die die http-Requests verarbeiten, diese sind:

- Create
- Edit
- Export
- Share
- ShareList

- View
- Voting

templates

Im Package templates befinden sich die HTML-Templates, CSS-Dateien und ein templateHelper, der die Templates initialisiert.

calendar

Im Package calendar befindet sich die tableView, die inputs aus der calendarView des Frontends bearbeitet und ggf. Cookies und die View anpasst.

configuration

Im Package configuration befindet sich die Konfiguration der Flags, die geparsed und gegebenenfalls mit Standardwerten belegt werden.

main

Von der main aus wird die Applikation gestartet. Hier werden alle Initialisierungen vorgenommen und zugehörige Wrapper und Handler für die htpt-Requests anhand der URL's registriert. Am Ende der main wird der Server gestartet.

2. Anwenderdokumentation

Anmeldung:

Als erstes muss sich ein Nutzer anmelden, um Zugriff auf seinem Kalender zu erhalten. Deshalb wird er beim Aufruf des Webservers zunächst auf die Anmeldeseite geleitet. Hier kann sich ein Nutzer entweder anmelden oder sich registrieren.

Kalenderansicht:

Bei erfolgreicher Anmeldung oder Registrierung wird der Nutzer auf seinen Kalender weitergeleitet. Dieser zeigt den aktuellen Monat an. Der Kalender enthält außerdem die Anzahl an Terminen pro Tag. Der Nutzer kann den angezeigten Monat ändern, indem er mit den Pfeilen einen Monat vor -oder zurückspringt. Außerdem kann er durch Eingabe eines Jahres und Monats einen beliebigen Monat auswählen. Durch Anklicken des aktuellen Datums kann er wieder zum aktuellen Monat zurückspringen.
In der Kalenderansicht kann der Nutzer auch seinen Kalender exportieren. Dafür muss er zunächst seine Anmeldedaten erneut eingeben und erhält dann eine ICal-Datei, die alle Termine des Nutzers beinhaltet.

Terminliste:

Außerdem kann der Nutzer zur Terminansicht navigieren, in der er Termine einsehen und bearbeiten kann. Hier werden dem Nutzer seine Termine in einer Liste nach Datum sortiert angezeigt. Die Liste beginnt mit dem aktuellen Datum und zeigt initial 7 Termine an. Angezeigt werden Titel, Beschreibung, Zeitraum und Wiederholungsintervall des Termins. Der Nutzer kann auf dieser Seite die Terminliste auch konfigurieren. Er kann das Datum einstellen, ab dem Termine angezeigt werden sollen. Außerdem kann er die Anzahl an Terminen pro Seite und die angezeigte Seite angeben. Durch Klicken des Submit-Buttons wird die Terminliste aktualisiert.
Der Nutzer kann über das entsprechende Suchfeld auch nach Terminen suchen. Hierbei werden Titel und Beschreibung der Termine durchsucht und die Ergebnisse werden als Liste angezeigt.

Termin bearbeiten/löschen:

In der Terminliste kann der Nutzer auch die angezeigten Termine bearbeiten oder löschen. Dabei wird er auf eine Detailansicht des ausgewählten Termins weitergeleitet. Hier kann er die Eigenschaften des Termins über die Eingabefelder ändern und die Änderungen speichern. Außerdem kann hier auch der Termin gelöscht werden. Danach wird der Nutzer wieder zurück auf die aktualisierte Terminliste geleitet.

Termin anlegen:

Die Terminliste enthält ebenfalls einen Button, um einen Termin anzulegen. Diese Ansicht ähnelt der Ansicht zum Bearbeiten/Löschen. Der Nutzer kann Titel, Zeitraum, Beschreibung und Wiederholungsintervall eintragen und einen Termin anlegen. Danach wird der Nutzer zurück auf die Terminliste geleitet.

Terminfindung:

Der letzte verfügbare Button der Terminliste ist der Button „Terminfindung“. Diese zeigt die offenen Terminfindungen des Nutzers an. Außerdem kann er über den entsprechenden Button eine Terminfindung erstellen. Hierbei muss der Nutzer einen Termintitel und einen Zeitraum eingeben und kann zusätzlich ein Wiederholungsintervall angeben.
Nach Anlegen der Terminfindung wird der Nutzer auf die Terminfindungsliste zurückgeleitet. Hier kann sich der Nutzer dann eine Terminfindung anzeigen lassen, um die zugehörigen Details einzusehen und die Terminfindung zu bearbeiten.

Terminfindung bearbeiten:

Die Detailansicht zeigt die bisher angelegten Zeiträume für den Termin sowie die eingeladenen Personen, die für diesen Termin abgestimmt haben. Außerdem kann der Nutzer des Kalenders Zeitvorschläge zur Terminfindung hinzufügen, für die die eingeladenen Personen abstimmen können. Dazu kann der Nutzer ebenfalls in dieser Ansicht Personen zur Terminfindung einladen. Die eingeladenen Personen werden ebenfalls in der Detailansicht der Terminfindung angezeigt.

Für Termine abstimmen:

Für die eingeladenen Personen einer Terminfindung wird ein eindeutiger Token in Form einer URL generiert. Bei Aufruf des Webservers mit dem erhaltenen Token wird die Person auf eine Seite geleitet, auf der sie für die angelegten Zeitvorschläge der Terminfindung abstimmen kann. Dem Nutzer des Kalenders kann anschließend einsehen, für welche Zeiträume die Person abgestimmt hat.

Terminfindung auswählen:

Der Nutzer kann aus seinen vorgegebenen Terminvorschlägen unabhängig vom Abstimmungsergebnis einen Vorschlag auswählen. Dieser wird dann in seinen Kalender übertragen. Das Abstimmungsergebnis wird als Beschreibung übernommen und die zugehörige Terminfindung wird gelöscht.

3. Dokumentation des Betriebs

Konfiguration und Starten der Anwendung

Die Konfiguration der Anwendung erfolgt über Startparameter beziehungsweise Flags. Um die Anwendung zu starten, muss man folgende Schritte befolgen:

1) Anwendung builden: go build <project> (in unserem Fall **go build go_cal**)
   Es wird eine ausführbare Datei mit dem Namen <project>.exe (hier: go_cal.exe) erstellt, welche dann ausgeführt werden kann.
2) Die zuvor erstellte .exe-Datei ausführen: **./go_cal**
   → wenn man keine Parameter übergibt, startet die Anwendung mit festgelegten default-Werten

<u>Folgenden Flags können gesetzt werden (mit jeweils einem Beispiel):</u>

- port (Port auf dem die Anwendung läuft)
  → **-port=443**
- timeout (Zeit in Minuten nachdem ein User wegen Inaktivität abgemeldet wird)
  → **-timeout=15**

- folder (relativer Pfad zu einem Ordner, in dem die Dateien gespeichert werden) → **-folder=.\files**
- certPath (Pfad zum SSL cert.pem) → **-certPath=.\cert.pem**
- keyPath (Pfad zum SSL key.pem) → **-keyPath=.\key.pem**

Beispiel: **go_cal.exe -port=443 folder=“.\files“ timeout:300 -certPath=“.\ssl\server.crt“ -keyPath=“.\ssl\server.key“**

Sessionhandling

Nutzer haben die Möglichkeit sich mit einem bestehenden Nutzernamen und Passwort anzumelden. Wenn man noch keinen Account besitzt, muss man sich mit Nutzername und Passwort registrieren.

Sobald sich ein User erfolgreich eigeloggt (oder registriert) hat, bekommt er einen Cookie („session_token“) im Browser zugewiesen. Dieser Cookie enthält einen serverseitig generierten Sessiontoken, welcher den Nutzer im Folgenden authentifiziert. Der zufällig generierte Sessiontoken wird serverseitig zusammen mit dem Nutzernamen und dem Expires-Datum gespeichert, um bei folgenden Requests einen Abgleich von dem gesendeten Cookie und dem gespeicherten Sessiontoken zu machen. Der Nutzer bleibt so über längere Zeit authentifiziert, ohne jedes Mal seine Anmeldedaten eingeben zu müssen.

Wenn nicht anders durch Flags konfiguriert, ist der Cookie für 10 Minuten gültig. Wenn der Nutzer aktiv auf der Seite herumklickt, verlängert sich der Cookie immer auf die 10 Minuten. Ist er allerdings für mindestens 10 Minuten inaktiv, läuft die Session ab und der Nutzer muss sich erneut über das Page-Login anmelden.

Die serverseitige Speicherung der Sesssion wird von einem sogenannten Server über Channels koordiniert, sodass es keine Probleme beim parallelen Zugriff (zwei Nutzer melden sich gleichzeitig an) auf die Datenstruktur gibt. Der Channel des Servers stellt sicher, dass immer nur eine go-Routine gleichzeitig auf die gespeicherten Sessions zugreifen kann (vor allem schreibend). Man erzeugt ein sogenanntes Command, welches man dann über einen Channel an den Server übergibt. Der Command beinhaltet neben alle nötigen Informationen auch die Art des angefragten Zugriffs (read, write, update oder remove). Die im Server laufende go-Routine „lauscht“ auf den Channel. Sobald ein Command aus einem Handler abgeschickt wurde, wird dieser im Server verarbeitet, beispielsweise wird eine neue Session zu einem Nutzer gespeichert. Währenddessen wartet der aufrufende Handler solange, bis der Server eine Antwort über den Antwortchannel schickt. Anschließend kann der

Programmfluss im Handler weitergehen. Da es sich um einen ungebufferten Channel an den Server handelt, kann der Server immer nur eine Anfrage gleichzeitig bearbeiten, wodurch es keine Race-Conditions auf die gespiecherten Sessions gibt.
Neben dem Sessiontoken gibt es einen weiteren Cookie („fe_parameter“), welcher im Browser des Nutzers gespeichert wird. Er enthält Parameter, die für die Anzeige im Frontend wichtig sind, beispielsweise den aktuell ausgewählten Monat in der Kalenderansicht. Dieser wird im Cookie zwischengespeichert, sodass der Nutzer nach dem Verlassen und anschließenden Zurückkehren zur Kalenderansicht die vorherige Ansicht wiederfindet. Zusätzlich werden die Einstellungen für die Seitennavigation in der Terminliste in dem Cookie gespeichert (Anzahl Seiten, Anzahl Einträge pro Seite). Hier wird serverseitig nichts dergleichen gespeichert, da es sich nur um Aspekte der Visualisierung handlet, welche nicht validiert werden müssen. Sobald sich Werte, die den Cookie betreffen ändern, wird der Cookie im Browser des Nutzers aktualisiert.

Basic Auth

Für den Export der Termine als ical wird Basic Auth verwendet. Wenn ein Nutzer Termine exportieren möchte, wird er aufgefordert, seinen Nutzernamen und sein Passwort über eine standardisiertes Browserpopup einzugeben. Anschließend werden seine Credentials (über https) im Header der Anfrage übermittelt und serverseitig validiert. Wenn die Authentifizierung erfolgreich war, wird der Download automatisch gestartet.

4. Einzelbeiträge zur Projektumsetzung

9495107:

- Alle Daten- und Substrukturen
- FileHandler, inkl. Strategie zur Aufteilung und Speicherung von Daten
- DataModel, inkl. Caching und API-Bereitstellung für alle benötigten Funktionen
- ICalExport, inkl. Aufbereitung in String nach RFC5545
- TLS Aktivierung, inkl. Benötigter lokaler Testumgebung und zugehöriger Flags für Zertifikate

Durch Anwendung von TDD jeweils inkl. Tests.

4706893:

- Frontend für Login und Register (html-Template)
- LoginHandler
- RegisterHandler
- LogoutHandler
- Sessionhandling, session_token-Cookie, Session läuft nach einer einstellbaren Zeit ab
- standardisiertes Error-Handling, um dem User Fehlermeldungen anzuzeigen
- Templatehelper, welcher die html-Templates zur Verfügung stellt
- Basic Auth für ical-Export
- Zum Teil: Funktion zur Terminfindung + Frontend-Einbindung dafür (Vorschlag hinzufügen, User hinzufügen, Abstimmung der User) + Handler und Templates

→ jeweils inkl. Tests

9608900:

- Kalenderansicht mit zugehörigem Handler, Template und Gestaltung + Frontend-Logik für Buttons und korrekte Anzeige der Daten
- Termin-Listenansicht mit zugehörigem Handler, Template und Gestaltung + Frontend-Logik für Buttons und korrekte Anzeige der Daten, Seitenzahl, Wahl der Seitenzahl, Wahl des Datums (über Cookie)
- Frontend-Einbindung für Terminsuche, Termin bearbeiten, Termin anlegen + zugehörige Handler und Templates, Anzeige der richtigen Daten und korrekte Veränderung der Daten
- Zum Teil: Funktion zur Terminfindung + Frontend-Einbindung dafür (Vorschlag hinzufügen, auswählen, User hinzufügen, Abstimmung der User) + Handler und Templates
- Tests für die oben genannten Komponenten