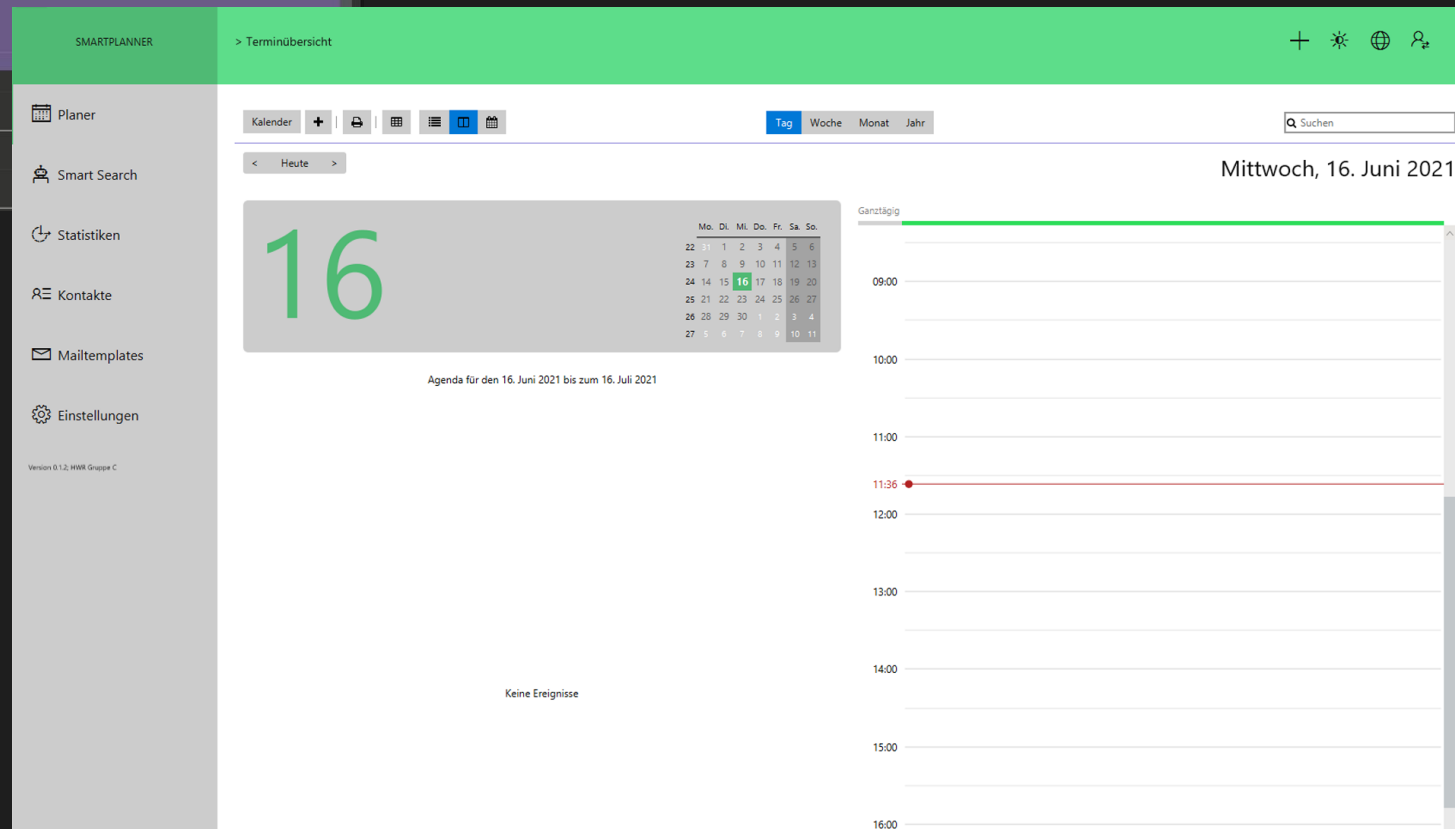


DevTeam:

Kamilla  
Danny  
Corentin  
Nico R.  
Robert  
Nico S.  
Hannes



# SMART PLANNER (Arbeitstitel)



# Gliederung

<> **Analysephase** (Robert): Gesamtübersicht der Usecases als Usecasediagramm, „Hauptusecase“ Smarte Suche als Aktivitätsdiagramm

<> **Entwurfsphase** (Nico): UML-Klassendiagramm und Sequenzdiagramm der Smarten Suche

<> **Architektur** (Danny): Nutzung des MVC Modells, abstrahierte Darstellung der Zusammenhänge der Funktionen

<> **Architektur** (Hannes): genutzte Patterns und Programmfluss, Module und externe Abhängigkeiten

The screenshot shows a web application interface for booking an appointment. At the top, there are three circular buttons: 'Daten' (highlighted in blue), 'Optionen', and 'Auswahl'. Below these is a section titled 'Basisinformationen' (Basic Information) with several input fields and controls:

- Terminname:** A text input field.
- Nächstmöglicher Termin:** A toggle switch is turned on. Next to it is a 'Suche ab' (Search from) field with a calendar icon.
- Termindauer:** A horizontal range slider from 0 to 240 minutes. A 'manuelle Eingabe' (manual input) field is to the right.
- Wochentag egal:** A toggle switch is turned on. To its right are checkboxes for days of the week: Mo, Di, Mi, Do, Fr, Sa, So.
- Termin liegt zwischen:** Two time selection fields (each with hour and minute dropdowns) separated by the word 'und'.

At the bottom of the 'Basisinformationen' section are two buttons: 'ZURÜCK' (Back) and 'WEITER' (Next).

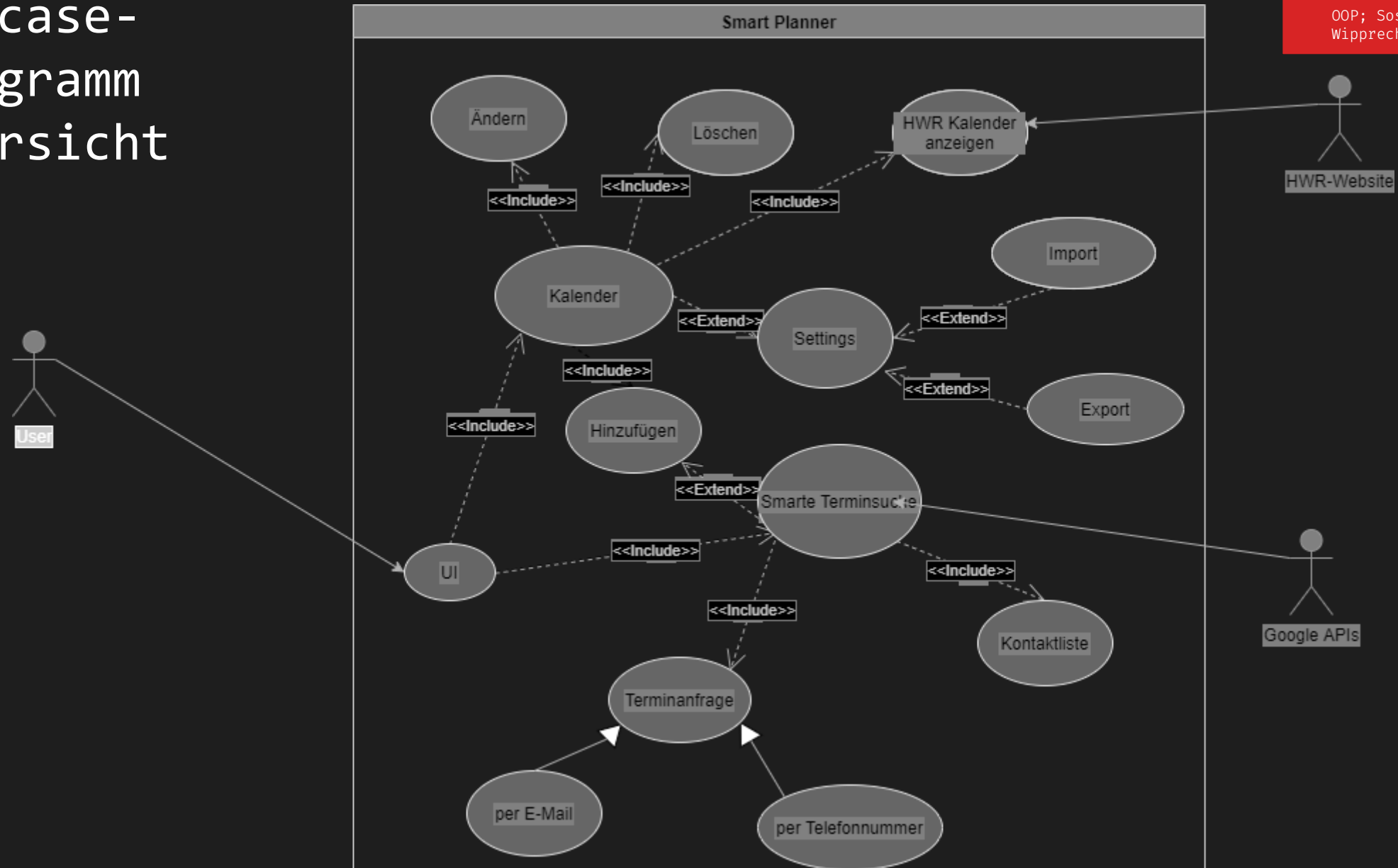
On the right side of the interface, there is a box titled 'Übersicht' (Overview) containing the following text:

Suche einen Termin mit einer Dauer von 60 Minuten, der zwischen dem 1.3.2021 und den 4.6.2021 liegt, dienstags oder donnerstags, in der Zeit von 12.00 bis 15.00 Uhr.

# Motivation und Zielstellung

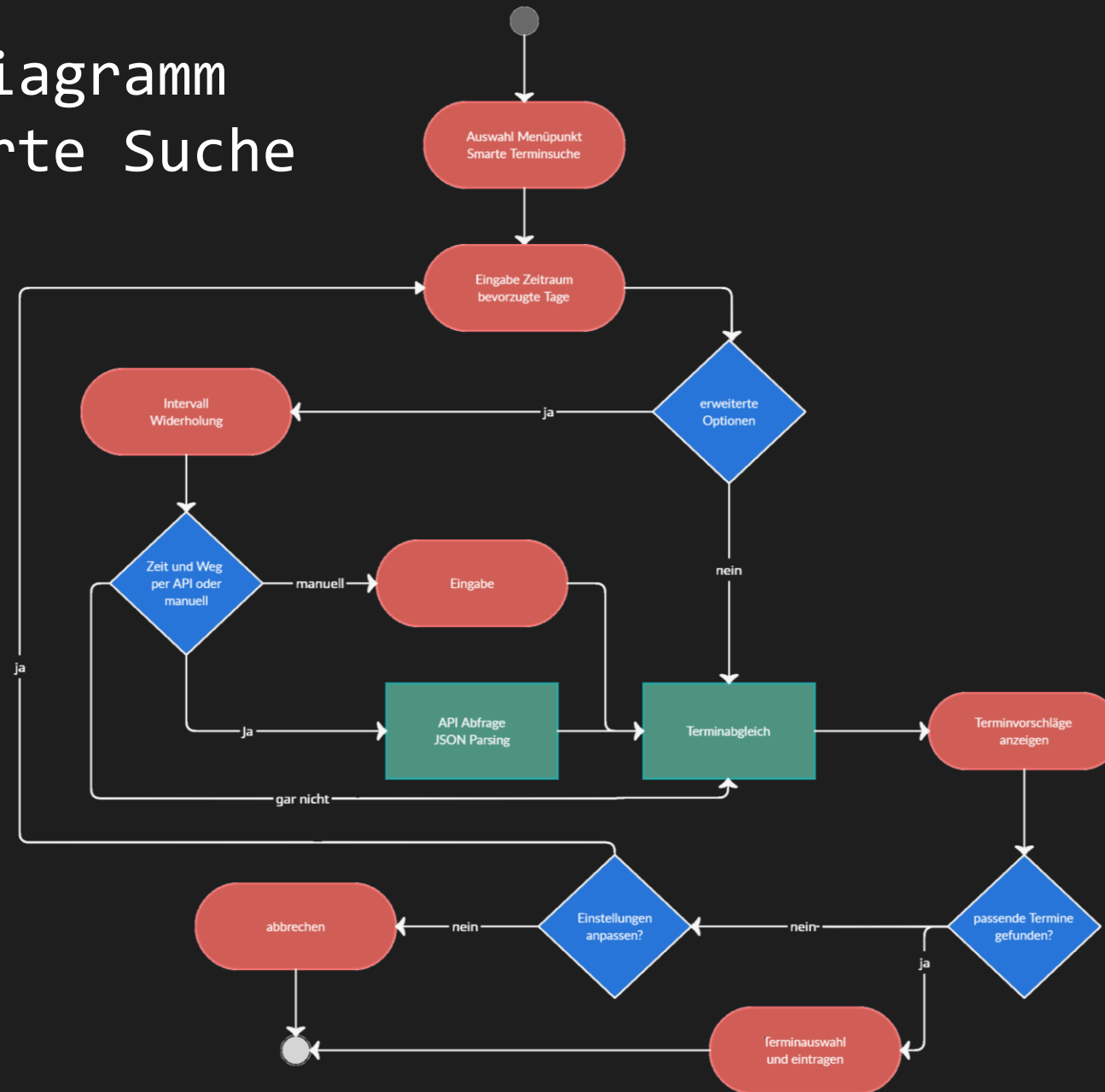
Motivation	Zielstellungen/Hauptanforderungen
vereinfachter Abruf des HWR Kalenders, der sich ständig ändert	eigene Kalenderapp, die regelmäßig die Änderungen crawlt (Intervall oder bei Änderung) und darstellt
Erinnerung und Buchung regelmäßiger Termine (Zahnarzt), die oft vergessen werden	dynamische Suche nach freien Terminen anhand vorhandener Termine und Vereinbarung des Termins per Mailanfrage (automatische Generierung der Mail)
Terminsuche oft schwierig bzw. manuell langwierig	smarte Vorschlagsfunktion die z.B. auch Fahrtzeiten mit einbezieht und automatisch Termine an freien Slots vorschlägt bzw. direkt einträgt
verteilte Kalender nerven und sind in der UX nicht konsistent (Google, HWR, private, Arbeit-Outlook)	durch Import eigener und des HWR Kalenders entsteht eine zentrale Verwaltung der Termine. Fremde Kalender können auch in der smarten Suche genutzt werden und anschließend exportiert werden.

# Usecase- Diagramm Übersicht

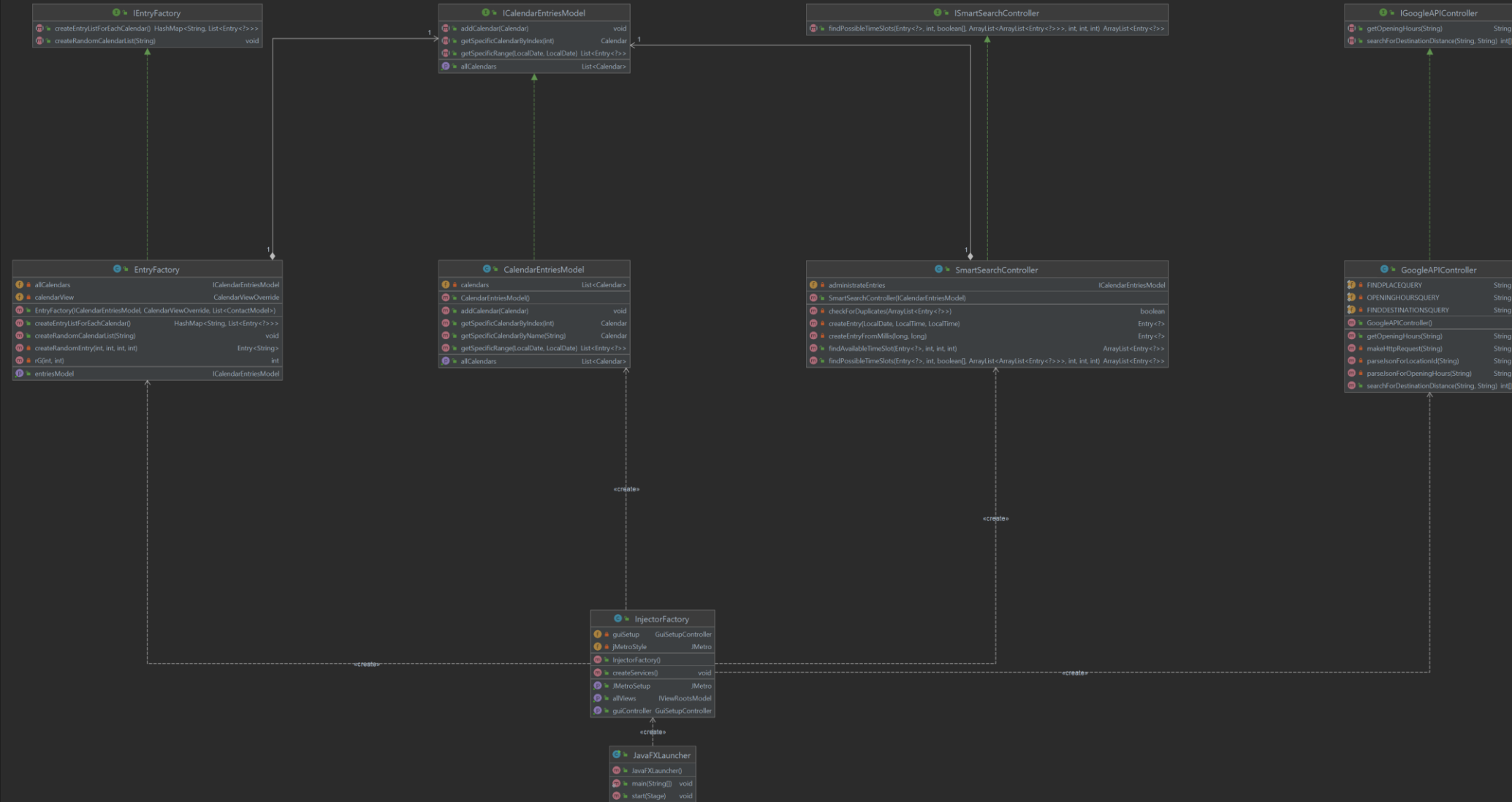


# Aktivitätsdiagramm

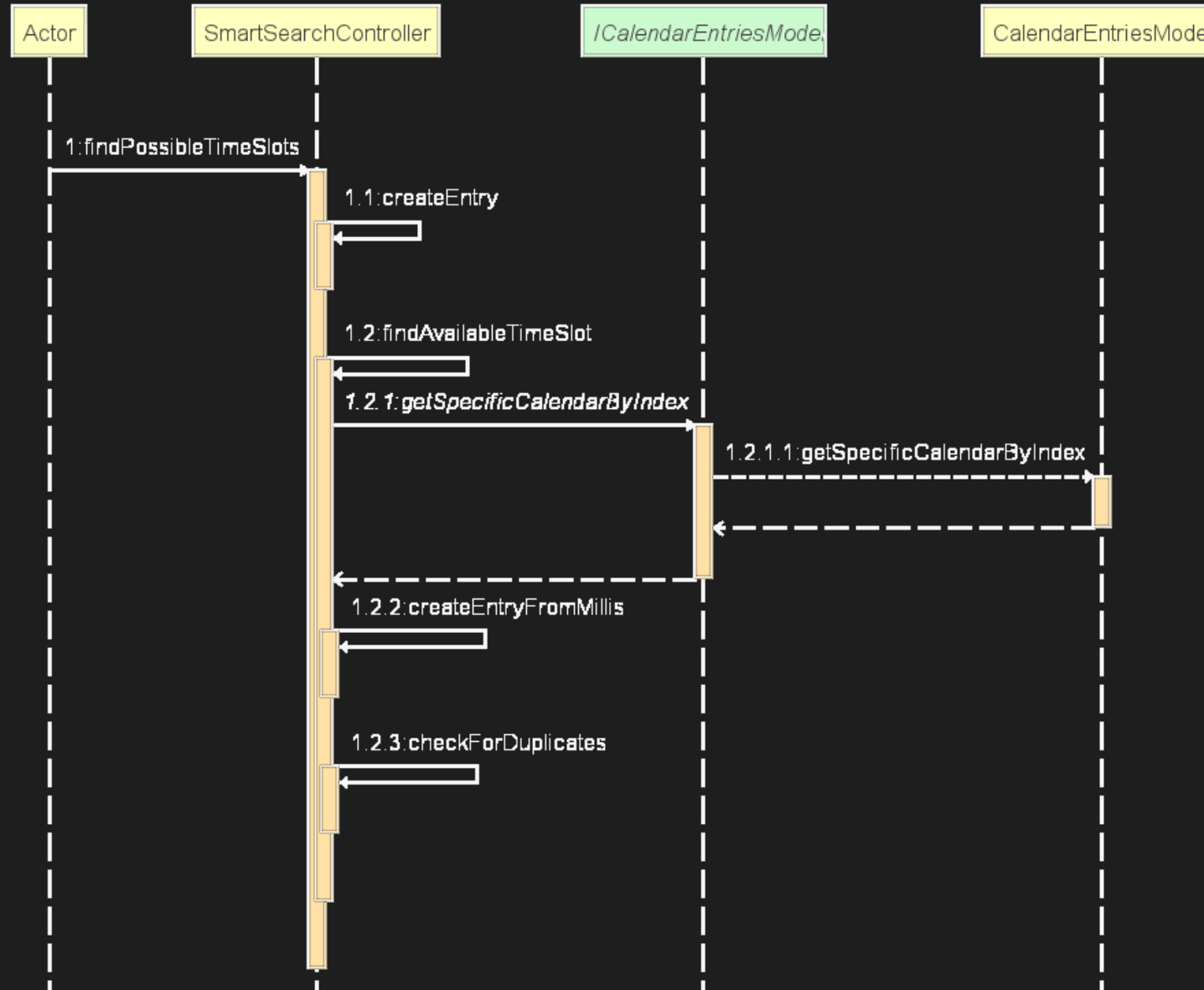
## Usecase Smarte Suche



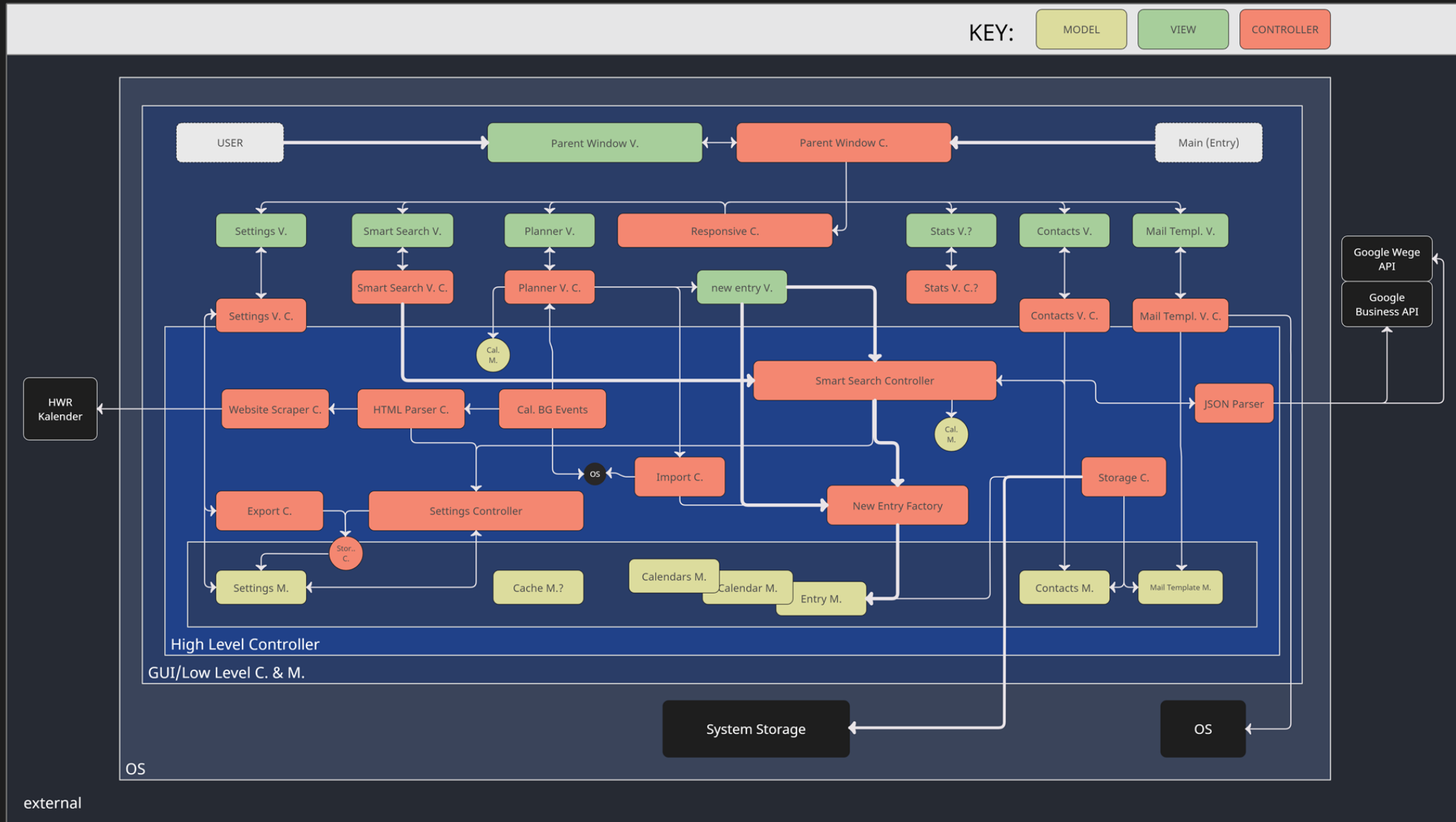
# UML – Klassendiagramm Usecase Smarte Suche



# Sequenzdiagramm Usecase Smarte Suche



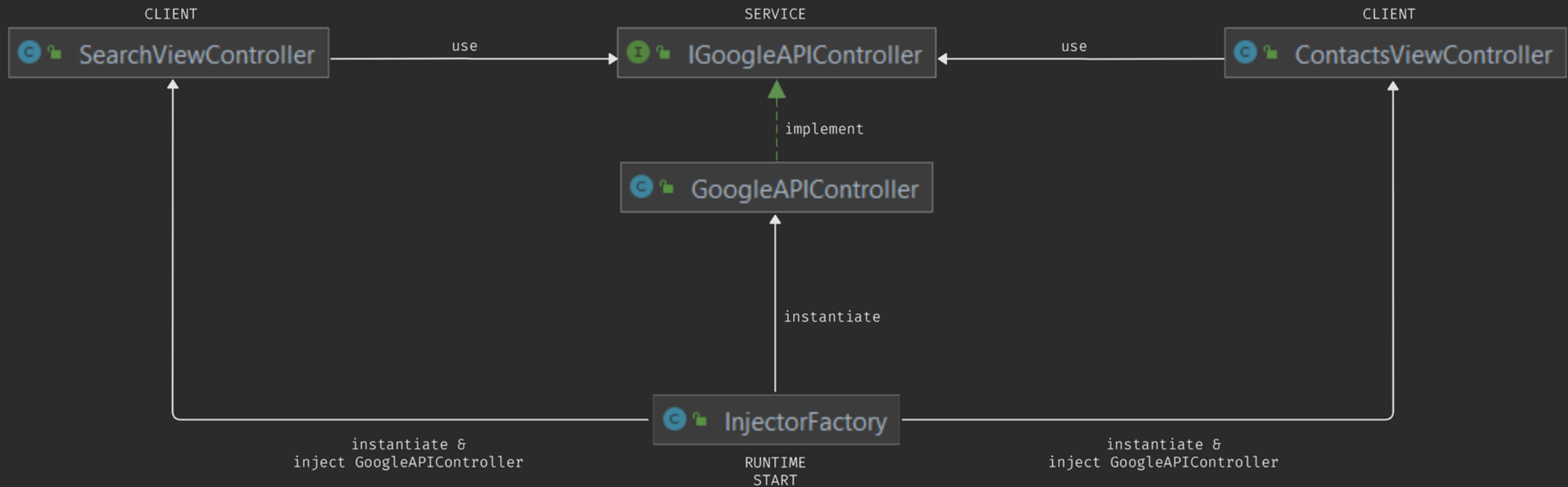
external





# Inversion of Control Principle (Fowler, Martin)

(hier implementiert durch Dependency Injection als Design Pattern)



```
public void createServices() throws IOException
{
    jMetroStyle = new JMetro();
    var customCalendarView = new CalendarViewOverride();
    var mailTemplates = new ArrayList<MailTemplateModel>();
    var contacts = new ArrayList<ContactModel>();
    var settings = new SettingsModel();

    IGoogleApiController apiCt = new GoogleApiController();
    IMailCreationController mailCreationCt = new MailCreationController(mailTemplates);
    ICalendarEntriesModel calendarEntriesModel = new CalendarEntriesModel();
    IContactFactory contactFactory = new ContactFactory(contacts);
    IWebsiteScraperController websiteCt = new WebsiteScraperController(settings);
    ISmartSearchController smartSearch = new SmartSearchController(calendarEntriesModel);
    IEntryFactory appointmentEntryCreator = new EntryFactory(calendarEntriesModel, customCalendarView, contacts);
    IIoController ioCt = new IoController(appointmentEntryCreator, contacts, settings, mailTemplates, calendarEntriesModel);
    IExportController exportCt = new ExportController(appointmentEntryCreator, contacts, settings, mailTemplates, calendarEntriesModel);
    IImportController importCt = new ImportController(appointmentEntryCreator, contacts, settings, mailTemplates, calendarEntriesModel);

    var settingsVCt = new SettingsViewController(settings);
    var statsVCt = new StatsViewController(contacts, calendarEntriesModel);
    var contactsVCt = new ContactsViewController(contacts, contactFactory, apiCt, ioCt);
    var mailVCt = new MailTemplateViewController(ioCt, settings, mailCreationCt, contacts, mailTemplates);
    var searchVCt = new SearchViewController(smartSearch, appointmentEntryCreator, contacts, contactFactory, mailTemplates, settings, apiCt);
    var plannerVCt = new PlannerViewController(calendarEntriesModel, appointmentEntryCreator, importCt, exportCt, customCalendarView);
    allViews = new ViewRootsModel(plannerVCt, searchVCt, statsVCt, contactsVCt, mailVCt, settingsVCt);
    guiSetup = new GuiSetupController(jMetroStyle, allViews);

    guiSetup.init();
    //ioCt.loadCalendarsFromFile();
    settings.addPropertyChangeListener(new ChangeListener());
    //websiteCt.startScraperTask();
}
```

DI am Beispiel der bei  
Start erzeugten Factory

letzte Folie: Gesamtarchitektur  
derzeitiger Stand als extra Datei  
(umlkomplett.png)