**Pflichtenheft Firmenprojekt SRZ**

**Schuljahr 2021/22**

**Cordic**

**Discord-Bot**

Schüler: Nick Edelhäuser, Ahmad Yosef, Florian Weinert

Kurs: Informatik 2, 3, 4

Firma: Communardo Software GmbH

Firmenvertreter: Enrico Kutscher

Vertretender AG-Leiter: Herr Stock

# Inhaltsverzeichnis

[Inhaltsverzeichnis 2](#_Toc90401791)

[Thema der Arbeit 3](#_Toc90401792)

[Beschreibung/Vorgaben des Projektes 4](#_Toc90401793)

[Terminverwaltung 4](#_Toc90401794)

[Projektverwaltung 4](#_Toc90401795)

[Sprach-Steuerung 4](#_Toc90401796)

[Optional 4](#_Toc90401797)

[Projektverwaltung 4](#_Toc90401798)

[Planung der Umsetzung 5](#_Toc90401799)

[Rahmenbedingungen und Voraussetzungen 5](#_Toc90401800)

[Anwendungsszenarien 5](#_Toc90401801)

[Schnittstellen / Daten 5](#_Toc90401802)

[Grafische Oberfläche 5](#_Toc90401803)

[Weitere Gedanken 5](#_Toc90401804)

[Grobgliederung 6](#_Toc90401805)

[Zeitplan 7](#_Toc90401806)

# Thema der Arbeit

Ein Discord-Bot (namens Cordic) soll die Jira-Projekte von Firmen über Discord, als grafische Oberfläche, anzeigen und verwalten können. Hinzu kommen noch weitere Funktionen wie eine Terminverwaltung, sowie eine Sprachsteuerung, mit der man die Funktionen des Discord-Bots auch anderweitig ausführen kann.

# Beschreibung/Vorgaben des Projektes

## Terminverwaltung

* + Konfigurierbar je Termin sind: Teilnehmer, Ort, Datum (von .. bis), Thema, Erinnerung (*Zeit vor dem Terminstart zu der eine Erinnerung an die Teilnehmenden gesendet wird*)
  + Termine können erstellt, abgesagt, bearbeitet werden
  + Jeder Teilnehmer hat eigenen (privaten) Termin-Channel
    - Dieser beinhaltet alle laufenden und anstehenden Termine die für den Teilnehmer relevant sind
    - (Optional) Admins haben keinen Zugriff
  + Zu jedem Termin existiert ein temporärer Text + Voice-Channel, welcher nur für die Terminteilnehmer verfügbar ist
  + Termin wird vom Host bzw. Leiter des Termins geschlossen und von ihm, falls nötig, nachträglich bearbeitet
  + Erinnerungsfunktion
    - sendet Direktnachrichten an Teilnehmer und an den Server-Channel bei Terminen, zu Deadlines, …

## Projektverwaltung

* + *einmalige administrative Tätigkeit*: Discord Bot wird mit bestehendem Jira-Cloud System verlinkt
  + existierende Tickets (Issues) können aus dem verlinktem Jira Projekt abgefragt werden.
  + Details zu diesen Tickets können auf Wunsch ebenfalls ermittelt werden
  + Die Datenabfrage erfolgt jeweils via ReST (immer von Discord [Servicenehmer] zu Jira [Service-Provider])

## Sprach-Steuerung

* + Als Ergänzung der Slash Commands. D.h. unterstütze Funktionen können zusätzlich zu den „/“-Befehlen (Slash Commands) auch über Sprachsteuerung ausgeführt werden

## Optional

### Projektverwaltung

* + Jira-Tickets verwalten

# Planung der Umsetzung

## Rahmenbedingungen und Voraussetzungen

Das Projekt soll als Open Source Projekt auf GitHub entstehen.

Der Quellcode wird auf Englisch und die Dokumentation auf Deutsch sein.

Der Discord-Bot befindet sich auf dem Firmen- oder Abteilungs-Discord-Server.

Es steht ein Jira Cloud Instanz zur Verfügung, die öffentlich verfügbar ist.

Der Discord Bot wird mit Nextcord geschrieben. Die Programmiersprache ist Python 3.9. Im Projekt nutzen wir das Venv-Modul, welches eine „virtual environment“ bereitstellt.

## Anwendungsszenarien

Man kann den Discord Bot mit einer Jira Cloud Instanz einer Firmen verknüpfen und mit Hilfe des Discord-Bots dedizierte Teile der Projekte verwalten sowie Projektdaten in Discord anzeigen lassen. Es wird außerdem möglich sein Firmen-Termine über Discord mit Hilfe des Discord-Bots zu führen.

## Schnittstellen / Daten

Alle benötigten Daten aus der jeweiligen Jira Cloud Instanz können über die bestehende Jira ReST API ermittelt werden.

Zur Beschreibung: <https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/>

## Grafische Oberfläche

Discord

## Weitere Gedanken

Nick wird die Sprachsteuerung für den Bot programmieren. Doch da das Empfangen von Sprachpaketen aus einem Discord Sprachkanal noch in der Testphase bei Nextcord ist, muss er einen experimentellen Github Branch des Moduls verwenden. Dieser könnte aber noch instabil sein und unbekannte Probleme aufweisen. Zur Spracherkennung wird das Python Modul "vosk" benutzt. Leider müssen wir bei der Genauigkeit der Spracheingabe Abzuüge machen wenn wir den Bot komplett auf Github hochladen wollen, da man zur Spracherkennung sogenannte Sprachmodelle braucht, welche das Vokabular und die Ausprache enthalten. Diese können aber bis zu 2GB groß sein, je nach Vokabular. Deshalb wird Nick (zumindest in der Testphase) ein kleineres, weniger genaues Sprachmodell mit ca. 50 MB Vokabular nutzen.

Die Termine werden in einer SQLite3 Datenbank temporär abgespeichert, bis zum Zeitpunkt des Termins. Danach werden sie entfernt. Die SQLite3 Datenbank enthält den Terminnamen, -zeitpunkt, -beschreibung, eine Liste an einbezogenen Mitgliedern und eine optionale Nachricht, die bei Terminstart verschickt wird. Termine können eingerichtet, angesehen und gelöscht werden.

Über der Jira Rest API werden die Daten als JSON Objekt übermittelt. Daraus kann man Projekte, Tickets, Nutzer, Kommentare, … auslesen. Man soll außerdem die Discord-Nutzer mit den Jira Nutzern verknüpfen können**.** Die Verknüpfungen werden ebenfalls in der SQLite3 Datenbank abgespeichert. Dadurch ist es möglich für jeden Nutzer einen Discord-Channel zu erstellen, in dem er einen Überblick über seine anstehenden Tickets hat, zusätzlich zum Gesamtüberblick zum Projekt.

# Grobgliederung

* Abfragen und veranschaulichen von Jira-Tickets, sowie den Details, über den Discord-Bot (Florian)
* Termine verwalten (Ahmad)
* Sprachsteuerung implementieren (Nick)
* Optional: Verwalten von Jira-Tickets über Discord (Florian + Nick)

# Zeitplan

Es ist jede Woche ein Meeting mit der Firma geplant.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Datum** | | **Termin** | **Aufgabe** | **Erledigt** |
| **von** | **bis** |
| 13.12 | 17.12 | Pflichtenheft Abgeben | Pflichtenheft vorbereiten (alle) |  |
| 20.12 | 24.12 | Weihnachtsferien | Experimentellen Branch von Nextcord-Modul ausprobieren (Nick)  Termine mit Konfiguration erstellen (+nachträglich bearbeiten) und in sqlite3 Datenbank abspeichern (Ahmad)  Verbindung Jira-Cloud mit Discord (Florian) |  |
| 27.12 | 31.12 | Weihnachtsferien | Audio Daten empfangen und in eigene Klasse umleiten (Nick)  Termine mit Konfiguration erstellen (+nachträglich bearbeiten) und in sqlite3 Datenbank abspeichern (Ahmad)  Abfragen von Jira-Tickets (Florian) |  |
| 03.01 | 07.01 |  | Audio Daten im richtigen Format in Sprachmodelle einspeisen (Nick)  Termine mit Konfiguration erstellen (+nachträglich bearbeiten) und in sqlite3 Datenbank abspeichern (Ahmad)  Abfragen von Jira-Tickets (Florian) |  |
| 10.01 | 14.01 |  | Sprachsteuerung beenden (Nick)  Termin-Chanels automatisiert erstellen, löschen und verwalten (Ahmad)  Veranschaulichen von Jira-Tickets + abspeichern der Verknüpfung von Discord Account mit Jira Nutzeraccount in Datenbank (Florian) |  |
| 17.01 | 21.01 |  | Erinnerungsfunktion (Ahmad)  Veranschaulichen von Jira-Tickets (Florian) |  |
| 24.01 | 28.01 |  | Terminverwaltung beenden (Ahmad)  Veranschaulichen von Jira-Tickets (Florian) |  |
| 31.01 | 04.02 |  | Terminverwaltung beenden (Ahmad) |  |
| 07.02 | 11.02 |  | Optional: Verwalten von Jira-Tickets über Discord (Florian + Nick) |  |
| 14.02 | 18.02 | Winterferien | Optional: Verwalten von Jira-Tickets über Discord (Florian + Nick) |  |
| 21.02 | 25.02 | Winterferien | Optional: Verwalten von Jira-Tickets über Discord (Florian + Nick) |  |
| 28.02 | 04.03 |  | Frei (Puffer) oder zusätzliche Ideen implementieren |  |
| 07.03 | 11.03 |  | Frei (Puffer) oder zusätzliche Ideen implementieren |  |
| 14.03 | 18.03 |  | Frei (Puffer) oder zusätzliche Ideen implementieren |  |
| 21.03 | 25.04 |  | Frei (Puffer) oder zusätzliche Ideen implementieren |  |
| 28.03 | 01.04 | Konsultation Projekt | Frei (Puffer) oder zusätzliche Ideen implementieren |  |
| 04.04 | 08.04 |  | Frei (Puffer) oder zusätzliche Ideen implementieren |  |
| 11.04 | 15.04 |  | Zwischenverteidigung vorbereiten (alle) |  |
| 18.04 | 22.04 | Osterferien | frei |  |
| 25.04 | 29.04 | Zwischenverteidigung | Erweiterungs- und Verbesserungsvorschläge bei Zwischenverteidigung mitberücksichtigen (alle) |  |
| 02.05 | 06.05 |  | Projekt fertigstellen + Erweiterungs- und Verbesserungsvorschläge bei Zwischenverteidigung mitberücksichtigen (alle) |  |
| 09.05 | 13.05 |  | Discord-Bot hochladen (alle) |  |
| 16.05 | 20.05 | Abgabe Projekt | Verteidigung vorbereiten (alle) |  |
| 23.05 | 27.05 |  | Verteidigung vorbereiten (alle) |  |
| 30.05 | 03.06 | Verteidigung |  |  |