Mittelstufenprojekt
Discord-Bot-Framework

BotHQ

5. Juni 2024

Philipp Batelka, Jan Mahnken, Daniel Quellenberg,
Fabian Reichwald, Justus Sieweke, Christopher Spencer

Was ist BotHQ?

BotHQ ist ein modulares Framework zur Erstellung und Verwaltung von Discord-Bots. Es bietet eine benutzerfreundliche Weboberfläche und eine robuste API zur Integration und Verwaltung von Plugins.

Durch die Verwendung von Plugins können Bots automatisch verschiedene Aufgaben ausführen, wie das Senden von Nachrichten, die Moderation von Benutzern, das Hosten von Spielen und sogar das Abspielen von Musik.

Projektübersicht

Ziel des Projekts:
Die Entwicklung eines modularen Discord-Bot-Frameworks, das als Cloud-Service angeboten wird und eine einfache Verwaltung von Plugins ermöglicht.

Zielgruppe:

- Gamer und Gaming-Communities
- Entwickler und Technikbegeisterte
- Unternehmen und Organisationen
- Content Creator und Streamer

Nutzen:
BotHQ erleichtert die Erstellung, Verwaltung und Konfiguration von Discord-Bots, indem es eine benutzerfreundliche Oberfläche und eine flexible API bietet.

Funktionen und Features

Hauptfunktionen

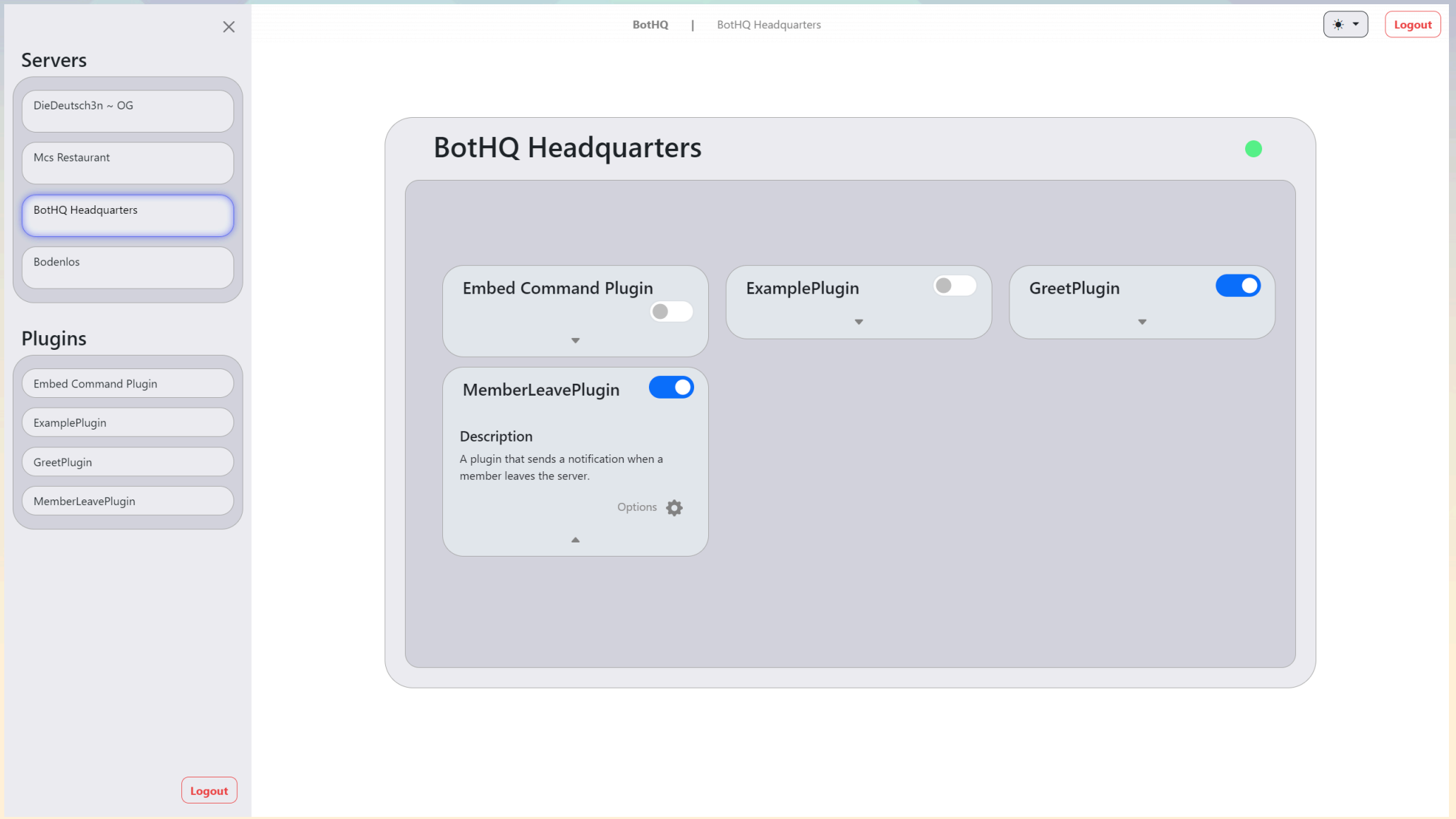
- **Plugin-Verwaltung:** Einfache Aktivierung, Deaktivierung und Konfiguration von Plugins
- **Benutzerfreundliche Weboberfläche:** Intuitive Verwaltung über das Frontend
- **Discord-Bot-Integration:** Nahtlose Einbindung in Discord-Server

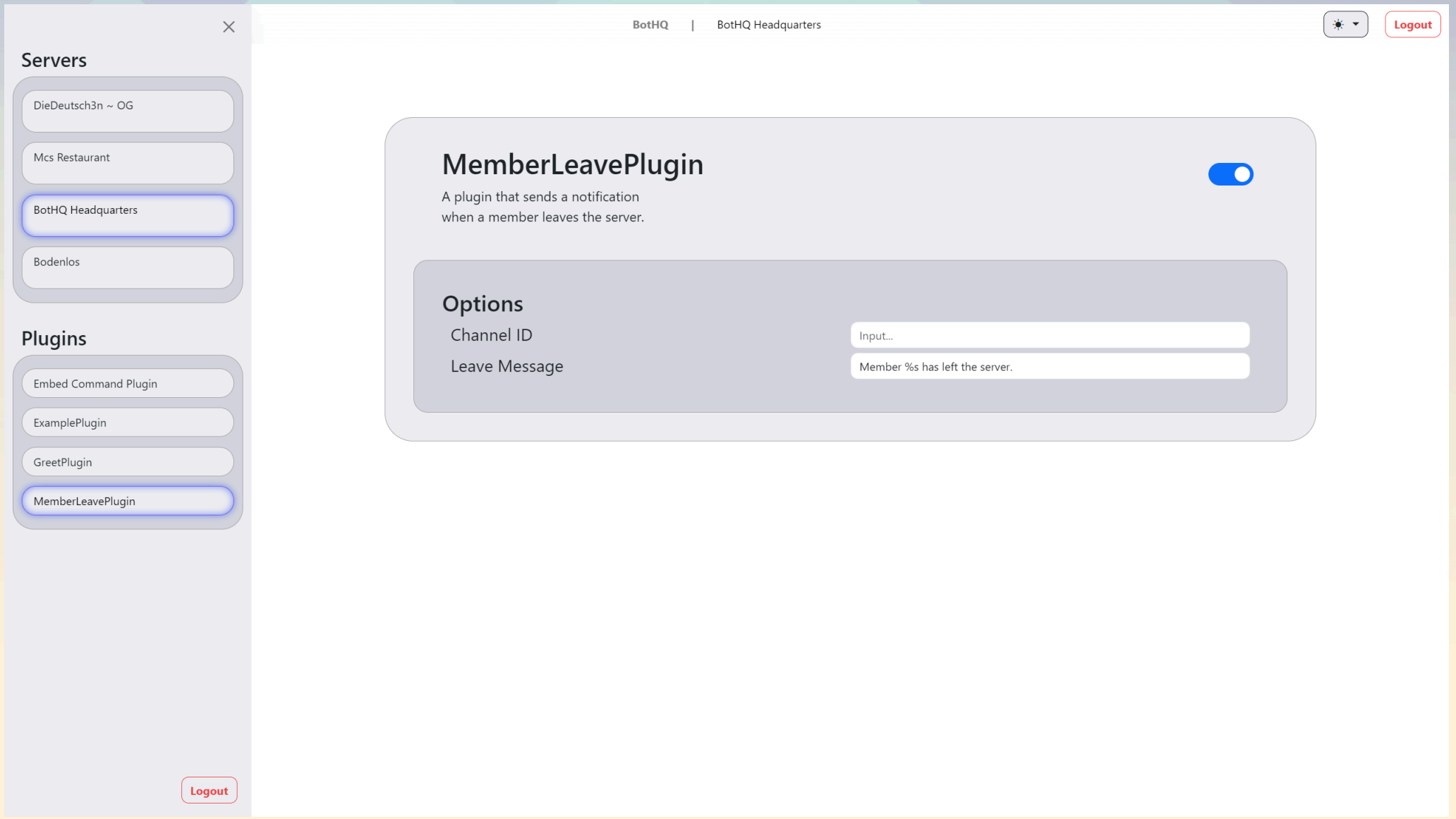
Anwendungsfälle

- **Moderation:** Automatische Verwaltung und Moderation von Discord-Servern
- **Engagement:** Interaktive Features für Community-Engagement
- **Automatisierung:** Automatisierung wiederkehrender Aufgaben und Prozesse

Vorteile des Frameworks

- **Automatisierte Einrichtung**: BotHQ automatisiert den Einrichtungsprozess für Discord-Bots.
- **Benutzerfreundliche Oberfläche**: Eine intuitive Weboberfläche zur effizienten Verwaltung von Bots und Plugins.
- **Modulares Framework**: Erweiterbar durch Hinzufügen neuer Plugins.
- **Sicher und Zuverlässig**: Fokus auf Sicherheit und Zuverlässigkeit.
- **Cloud-basierter Service**: Hosting auf zuverlässiger Cloud-Plattform.


Servers
DieDeutsch3n ~ OG
Mcs Restaurant
BotHQ Headquarters
Bodenlos
Plugins
Embed Command Plugin
ExamplePlugin
GreetPlugin
MemberLeavePlugin
Logout
BotHQ | BotHQ Headquarters
Logout
BotHQ Headquarters
Embed Command Plugin
ExamplePlugin
GreetPlugin
MemberLeavePlugin
Description
A plugin that sends a notification when a member leaves the server.
Options


Servers
DieDeutsch3n ~ OG
Mcs Restaurant
BotHQ Headquarters
Bodenlos
Plugins
Embed Command Plugin
ExamplePlugin
GreetPlugin
MemberLeavePlugin
Logout
BotHQ | BotHQ Headquarters
Logout
MemberLeavePlugin
A plugin that sends a notification
when a member leaves the server.
Options
Channel ID
Input...
Leave Message
Member %s has left the server.

Technische Betrachtung

Entwicklungsprozess

Vorgehensmodell:

Das Projekt wurde nach dem agilen Scrum-Framework durchgeführt, angepasst an die spezifischen Bedürfnisse des Teams.

Sprints und Meilensteine:

- **Sprint 1:** Anforderungsanalyse und Datenbankdesign
- **Sprint 2:** Implementierung der Kernfunktionen und Design
- **Sprint 3:** Erstellung von Plugins, Testen, Feinschliff

Herausforderungen und Lösungen:

- **Herausforderung:** Datenbankanpassungen während der Entwicklung
- **Lösung:** Flexibles Datenbankschema und kontinuierliche Integration

Technologie-Stack

Verwendete Technologien:

- **Backend:** Spring Boot
- **Frontend:** Angular
- **Datenbank:** PostgreSQL
- **API:** REST
- **Bot-Integration:** JDA (Java Discord API)

Warum diese Technologien?

- **Spring Boot:** Framework für robuste Backend-Entwicklung
- **Angular:** Frontend-Framework für dynamische Webanwendungen
- **PostgreSQL:** Skalierbare Datenbanklösung
- **JDA:** Bibliothek für die Integration von Discord-Bots in Java

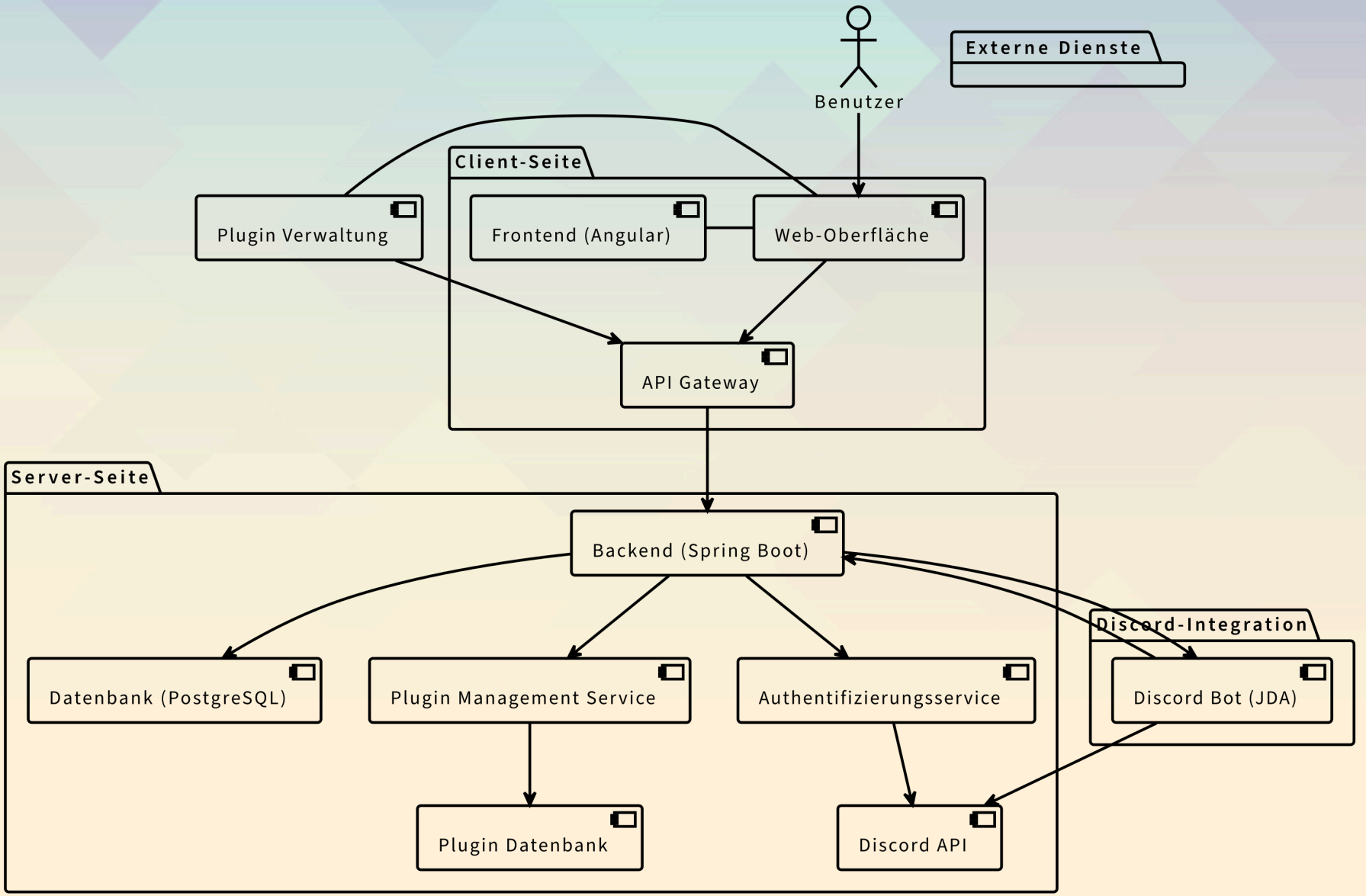
Architektur des Systems

Komponenten

- **Core:** Grundlegende Logik und Datenverarbeitung
- **Lib(rary):** Wiederverwendbare Bibliotheken und Utilities
- **Frontend:** Benutzeroberfläche zur Verwaltung von Servern, Bots und Plugins
- **Discord-Plugins:** Individuelle Plugins für spezifische Bot-Funktionen

Interaktionen

- **API:** Kommuniziert zwischen Frontend und Backend
- **Datenbank:** Speichert Nutzerdaten, Bot-Konfigurationen und Plugin-Einstellungen


Benutzer
Externe Dienste
Client-Seite
Plugin Verwaltung
Frontend (Angular)
Web-Oberfläche
API Gateway
Server-Seite
Backend (Spring Boot)
Datenbank (PostgreSQL)
Plugin Management Service
Authentifizierungsservice
Discord-Integration
Discord Bot (JDA)
Plugin Datenbank
Discord API

Wichtige Klassen und Module – Backend

Core

- Der Kern des Discord-Bot-Frameworks.
- Verantwortlich für die grundlegenden Funktionen und die Verwaltung der Bot-Operationen.
- Interagiert mit anderen Modulen und Diensten, um Befehle zu verarbeiten und Benutzer zu verwalten.

Core-Services	Beschreibung
`BotService`	Steuert den Betrieb des Discord-Bots.
	Verarbeitet Befehle und Ereignisse.
	Interagiert mit Plugins.
`UserService`	Verwalten der Benutzerinformationen und -rollen.
	Authentifizierung und Autorisierung.

Library

- Eine Sammlung von Hilfsfunktionen.
- Macht den Code sauberer und wiederverwendbar.
- Spart Zeit bei der Entwicklung.

Library	Beschreibung
`UtilityFunction`	Allgemeine Hilfsfunktionen.
	Methoden für String-Manipulationen, Formatierungen
	und Validierungen.
	Reduziert Code-Duplikationen.
`DatabaseHelper`	Vereinfachung von Datenbankoperationen.
	Verwaltung von Abfragen und Transaktionen.
	Unterstützung bei der Erstellung von Datenbankschemata.

Wichtige Klassen und Module – Frontend

- Die Benutzeroberfläche des Discord-Bot-Frameworks.
- Ermöglicht Nutzern, den Bot einfach zu verwalten.
- Zeigt wichtige Informationen und erlaubt die einfache Konfiguration.

Frontend-Komponenten	Beschreibung
`DashboardComponent`	Zentrale Übersicht und Verwaltung.
	Darstellung von Statistiken und Statusinformationen.
	Navigation zu Verwaltungskomponenten.
`PluginManagerComponent`	Verwaltung und Konfiguration von Plugins.
	Aktivierung, Deaktivierung und Anpassung von Plugins.
	Schnittstelle zur Backend-API.

Plugins

- Zusätzliche Funktionen, die der Bot ausführen kann.
- Erweitern die Fähigkeiten des Bots.
- Bieten spezifische Funktionen wie Begrüßungen und Moderation.

Plugins	Beschreibung
`WelcomePlugin`	Begrüßt neue Mitglieder auf dem Server.
	Anpassbare Begrüßungstexte.
	Unterstützt dynamische Inhalte.
`ModerationPlugin`	Verwalten von Servermoderation.
	Kick- und Ban-Funktionalitäten.
	Protokollierung von Moderationsaktionen.

Quellcode-Beispiele

Backend (Spring Boot)

```
@RestController
@RequestMapping("/api/bots")
public class BotController {
    @Autowired
    private BotService botService;

    @PostMapping
    public ResponseEntity<Bot> createBot(@RequestBody Bot bot) {
        return ResponseEntity.ok(botService.createBot(bot));
    }
}
```

Frontend (Angular)

```
@Component({
  selector: "app-plugin-manager",
  templateUrl: "./plugin-manager.component.html",
})
export class PluginManagerComponent {
  // Methoden zur Verwaltung von Plugins
}
```

```
@Component({
  selector: "app-dashboard",
  templateUrl: "./dashboard.component.html",
})
export class DashboardComponent {
  // Logik für das Dashboard
}
```

Plugin (Java)

```java
@DiscordEventListener(GuildMemberJoinEvent.class)
    public void onGuildMemberJoin(@NotNull GuildMemberJoinEvent event) {
        Guild guild = event.getGuild();
        log.debug("User joined on guild " + guild.getId());

        String channelId = channel.get(guild.getIdLong()).getValue();
        TextChannel greetingChannel = guild.getTextChannelById(channelId);
        if (greetingChannel == null) return;

        log.debug("Greeting channel is \"" + greetingChannel.getName() + "\"");


        String messageValue = message.get(guild.getIdLong()).getValue();

        // It would be nice to have a proper service which allows for proper usage of objects,
        // allowing things like {user.<name,mention>} etc.
        messageValue = messageValue.replace("{user.name}", event.getUser().getEffectiveName());
        messageValue = messageValue.replace("{user.mention}", event.getUser().getAsMention());

        greetingChannel.sendMessage(messageValue).queue();
```

Tests und Qualitätssicherung – Work in Progress

Teststrategie:

- **Unit Tests:** Überprüfung einzelner Methoden und Funktionen (JUnit, Mockito)
- **Integrationstests:** Überprüfung des Zusammenspiels verschiedener Komponenten
- **Systemtests:** Gesamtsystemtests unter realistischen Bedingungen

Testergebnisse:

- **Unit Tests:** geplant sind > 95% Abdeckung
- **Integrationstests:** Erfolgreiche Verbindungen zwischen Komponenten
- **Systemtests:** Alle Hauptfunktionen arbeiten wie erwartet

Qualitätssicherung:

- Regelmäßige Code Reviews
- Kontinuierliche Integration und Deployment (CI/CD)

Nutzerdokumentation – Work in Progress

Benutzerhandbuch:

- Schritt-für-Schritt-Anleitung zur Installation und Konfiguration
- FAQ und Problemlösungen

Anleitungen:

- **Installation:** Einfache Schritte zur Installation des Bots
- **Konfiguration:** Anpassen der Bot-Einstellungen über die Weboberfläche

Beispiele:

- **Willkommensnachricht:** Konfigurieren einer Begrüßungsnachricht für neue Mitglieder
- **Moderation:** Automatisiertes Kicken/Bannen von Mitgliedern

Wirtschaftliche Betrachtung – Marktuntersuchung

Zielgruppenanalyse und Marktpotenzial

- **Gaming-Communities**: Beliebt bei Gamern und Gaming-Communities für Management, Spielinformationen und Eventorganisation.
- **Technikbegeisterte und Entwickler**: Benutzerfreundliche Umgebung für Bot-Programmierung und Erstellung eigener Bots.
- **Unternehmen und Organisationen**: Genutzt für Teamkommunikation, Zusammenarbeit und Automatisierung.
- **Content-Ersteller und Streamer**: Interaktion mit der Community, Kontakt mit dem Publikum und Plattform für Fans.

Konkurenzanalyse

Konkurenzprodukte: Dyno Bot, Mee6, Carl-bot, GAwesome Bot, Nightbot, BotGhost

Vorteile von BotHQ:

- Automatisierte Installation und Konfiguration
- hohe Benutzerfreundlichkeit
- Aktivierung und Konfiguration von Plugins, ohne den Bot neu starten zu müssen
- keine manulle bearbeitung von Dateien nötig
- ohne technische Vorkenntnisse nutzbar

Marktvolumen und Marktpotenzial

- **Nutzerbasis von Discord**: Über 250 Mio. registrierte Nutzer
- **Verwendung von Discord-Bots**: Etwa 30% der Discord-Server nutzen Bots
- **Marktgröße**: Potenzielles Marktvolumen von 30 Mio. aktiven Nutzern
- **Wachstum des Bot-Marktes**: Wachstum des Bot-Marktes
- **Steigende Nachfrage**: Breite Zielgruppe und Erweiterungsmöglichkeiten
- **Wettbewerbsvorteile**: Automatisierte Einrichtung, benutzerfreundliche Oberfläche, modulares Framework, Sicherheit und Zuverlässigkeit

Wachstumschancen

- **Wachstum der Discord-Plattform**: Discord hat in den letzten Jahren eine große Nutzerbasis aufgebaut und wird in verschiedenen Communities und Branchen immer beliebter.
- **Steigende Nachfrage nach Automatisierung**: Immer mehr Discord-Server-Besitzer suchen nach Möglichkeiten, ihre Server zu automatisieren und repetitive Aufgaben zu vereinfachen.
- **Erweiterte Funktionen und Integrationen**: Discord bietet eine umfangreiche API, die es Entwicklern ermöglicht, ihre Bots mit anderen Diensten und Plattformen zu integrieren.
- **Marktexpansion**: Discord wird nicht nur in Gaming-Communities, sondern auch in Bereichen wie Bildung, Unternehmenskommunikation und sozialen Netzwerken immer häufiger genutzt.

Wirtschaftliche Betrachtung – Kostenplanung

Gesamtkosten des Projekts

Kostenart	Kostenbetrag
Personalkosten	18.384 EUR
Sachmittelkosten	9.060 EUR
Gesamtkosten	27.444 EUR

Wirtschaftlichkeitsberechnung

Gewinnschwellenberechnung

Kostenart	Kostenbetrag
Gesamtkosten des Projekts	27.444 EUR
Preis des Produktabos	10 EUR pro Monat
Gewinnschwelle	2.745 Abos für einen Monat

Amortisationsrechnung

Dauer	Einnahmen
Monatliche Einnahmen	500 Abos x 10 EUR = 5.000 EUR
Amortisationszeit	27.444 EUR / 5.000 EUR = 5,5 Monate

Fazit und Ausblick

Zusammenfassung:

- Erfolgreiche Entwicklung eines modularen Discord-Bot-Frameworks
- Bereitstellung als Cloud-Service
- Flexible und benutzerfreundliche Verwaltung von Plugins

Erkenntnisse und Lessons Learned:

- Wichtige Rolle der agilen Methoden in der Softwareentwicklung
- Bedeutung der Flexibilität bei der Anpassung an neue Anforderungen
- Erfolgreiche Teamarbeit und Kommunikation als Schlüsselfaktoren

Ausblick:

- Erweiterung der Plugin-Bibliothek
- Verbesserung der Benutzeroberfläche
- Integration zusätzlicher Funktionen und Dienste