|  |  |
| --- | --- |
|  | Fachhochschul-Bachelorstudiengang  **SOFTWARE ENGINEERING**  A-4232 Hagenberg, Austria |

**Vergleich der Extension-APIs  
in VS Code und IntelliJ**

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Science in Engineering

Eingereicht von

**Philipp Seiringer**

Begutachtet von Dr. Josef Pichler

Hagenberg, Februar 2024

**Erklärung**

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Die vorliegende, gedruckte Bachelorarbeit ist identisch zu dem elektronisch übermittelten Textdokument.

Datum Unterschrift

Kurzfassung

Abstract

Inhaltsverzeichnis

[Kurzfassung 3](#_Toc145439493)

[Abstract 4](#_Toc145439494)

[Inhaltsverzeichnis 5](#_Toc145439495)

[1 Einleitung 6](#_Toc145439496)

[1.1 Motivation 6](#_Toc145439497)

[1.2 Ziel 6](#_Toc145439498)

[1.3 Aufbau der Arbeit 6](#_Toc145439499)

[2 Vergleich der Entwicklungsplattformen 7](#_Toc145439500)

[2.1 Visual Studio Code 7](#_Toc145439501)

[2.2 IntelliJ IDEA 7](#_Toc145439502)

[3 Vergleich der angebotenen Funktionalität und deren Anwendbarkeit 11](#_Toc145439503)

[4 Codebeispiel: Recent Changes Plugin 12](#_Toc145439504)

[5 Vergleich der Code-Metriken 13](#_Toc145439505)

[6 Conclusion 14](#_Toc145439506)

[Referenzen 15](#_Toc145439507)

1. Einleitung
   1. Motivation

SoftwareentwicklerInnen arbeiten täglich mit verschiedensten Werkzeugen und Entwicklungsumgebungen, sogenannten IDEs (=Integrated Development Environment). Diese Plattformen bieten teils sehr unterschiedliche Funktionalitäten, die die Softwareentwicklung erleichtern sollen. Dabei bieten sie Unterstützung für verschiedenste Programmiersprachen und Technologien und binden zahlreiche Werkzeuge für spezifische Anwendungsfälle ein. Aufgrund des immer rascher werdenden Entstehens von neuen Technologien bieten mehr und mehr IDEs Möglichkeiten zur Entwicklung von eigenen Plugins, welche dann auch an andere EntwicklerInnen bereitgestellt werden können. So können in kürzester Zeit neue Technologien unterstützt werden und EntwicklerInnen haben selbst die Macht darüber zu entscheiden welche Plugins sie nutzen möchten und welche nicht.

Vor der Entwicklung solcher Plugins ist es wichtig zu entscheiden für welche IDE das Plugin erstellt werden soll. Dabei spielen Aspekte wie zum Beispiel die Einfachheit und Flexibilität in der Entwicklung, der Umfang an angebotener Funktionalität, die Möglichkeit die Nutzerinteraktion und somit die User Experience zu steuern und viele weitere eine Rolle. Diese Bachelorarbeit versucht in diesen Bereichen einen Überblick zu schaffen und vergleicht hierfür die Plugin Entwicklung in zwei der momentan beliebtesten IDEs, Visual Studio Code und IntelliJ IDEA. Durch den Vergleich der beiden Produkte und dem Herausarbeiten und Aufbereiten der Unterschiede wird es anderen EntwicklerInnen erleichtert diese Entscheidung zu treffen.

* 1. Ziel
  2. Aufbau der Arbeit

1. Vergleich der Entwicklungsplattformen
   1. Visual Studio Code

Die erste offizielle Version von Visual Studio Code, häufig abgekürzt auch als VS Code, wurde am 14. April 2016 von Microsoft veröffentlicht. Die Idee hinter VS Code war einen möglichst einfachen Code Editor anzubieten, welcher nur die wichtigsten und besten Funktionen für EntwicklerInnen beinhaltete. Es hob sich somit von anderen IDEs wie der Visual Studio Reihe von Microsoft ab, da es ein sehr leichtgewichtiger Editor war, welcher trotzdem mit einer großen Menge an Programmiersprachen arbeiten konnte und für diese auch Microsofts code completion namens „IntelliSense“ unterstützte. Weiters war Visual Studio Code das erste Produkt der Visual Studio Familie welches Cross-Plattform Entwicklung auf Windows, Linux und OSX unterstützte.

Aus den Stack Overflow developer surveys der vergangenen Jahre kann der rasche Aufstieg von VS Code beobachtet werden. Während es im Jahr 2016 nur von etwa 7,2 Prozent der EntwicklerInnen genutzt wurde, war es zwei Jahre später bereits (wenn auch knapp) das meistgenutzte IDE mit 34,9%. In der aktuellsten Umfrage von 2023 war es der klare Sieger und wurde vom 73,71% der Abstimmenden aktiv genutzt.

Ein Grund für diesen Erfolg mag vermutlich die Möglichkeit zur Entwicklung von Plugins sein. Durch die direkte Einbindung des Visual Studio Marketplace in VS Code bildete sich über die Jahre eine große Community die eine enorme Anzahl von Plugins entwickelt, verbessert und betreut.

Src:

<https://en.wikipedia.org/wiki/Visual_Studio_Code>

<https://de.wikipedia.org/wiki/Visual_Studio_Code>

<https://web.archive.org/web/20151009211114/http://blogs.msdn.com/b/vscode/archive/2015/04/29/announcing-visual-studio-code-preview.aspx>

<https://shiftmag.dev/vs-code-171/>

* 1. IntelliJ IDEA

IntelliJ IDEA wurde erstmals im Januar 2001 von dem Unternehmen JetBrains veröffentlicht. Im Gegensatz zu Visual Studio Code handelt es sich bei IntelliJ um ein IDE welches speziell auf die Entwicklung von Programmen in den Programmiersprachen Java und Kotlin ausgelegt ist. IntelliJ IDEA wird in einer frei zu verwendenden, open source „Community Edition“, sowie in einer kommerziellen Form als „IntelliJ IDEA Ultimate“ angeboten.

Aufgrund der Spezialisierung auf Java und Kotlin Projekte unterstützt die IntelliJ Community Edition nur eine relativ kleine Auswahl an Sprachen, Frameworks und Build Tools. Während IntelliJ IDEA Ultimate den Umfang an Features schon deutlich erweitert, bietet JetBrains auch noch weitere (kommerzielle) IDEs an. Diese sind alle für unterschiedliche Programmiersprachen oder Sprachfamilien ausgelegt. Einige der bekanntesten sind dabei CLion für die Sprachen C und C++, Rider für die .NET Sprachen, PhpStorm für PHP, WebStorm für JavaScript und viele weitere. Zum aktuellen Zeitpunkt sind es insgesamt elf verschiedene IDEs die von JetBrains angeboten werden und die alle auf der IntelliJ Platform basieren. Das bedeutet nicht nur, dass sich all diese IDEs in der Verwendung und im Aussehen sehr ähnlich sind, sondern auch, dass ein Plugin, welches für die allgemeine IntelliJ Platform entworfen wurde, relativ problemlos auch für mehrere IDEs dieser Form veröffentlicht werden kann.

Im Gegensatz zu Visual Studio Code ist IntelliJ ein eher schwergewichtiger Editor, der sehr viel Funktionalität schon von Grund auf eingebaut hat. Die EntwicklerInnen sind hier nicht so stark auf Plugins angewiesen. Dies lässt sich auch durch die Anzahl von Plugins erkennen, die auf dem JetBrains Marketplace angeboten werden. Für die IntelliJ Platform gibt es aktuell etwas über 7500 Plugins die in die IDE integriert werden können. Für Visual Studio Code sind es hingegen inzwischen fast 51000 Plugins.

Zum aktuellen Zeitpunkt bietet IntelliJ in den unterschiedlichen Versionen folgende features:

Sprachen

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Frameworks

Ein Bild, das Text, Screenshot, Dokument enthält.

Automatisch generierte Beschreibung

Build Tools

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

TODO irgendwas mit andere IDEs von Jetbrains

Src:

<https://en.wikipedia.org/wiki/IntelliJ_IDEA>

<https://www.jetbrains.com/products/compare/?product=idea-ce&product=idea>

1. Vergleich der angebotenen Funktionalität und deren Anwendbarkeit
   1. Visual Studio Code
      1. Das Plugin Projekt
         1. Setup eines Plugin Projekts

Zum Aufsetzen eines neuen Projektes kann bei VS Code ein einfaches Generator Programm verwendet werden, welches die Ordnerstruktur den Angaben entsprechend aufbaut. Funktionieren tut dies über die beiden npm Packages „Yeoman“ und „VS Code Extension Generator“, welche mit dem Befehl „npm install -g yo generator-code“ installiert werden können. Der Generator wird dann mit „yo code“ gestartet. Daraufhin können verschiedene Templates für das Projekt gewählt werden und Angaben zum Projektnamen und ähnlichen Parametern getroffen werden.

* + - 1. Aufbau eines Plugins

Beim Generieren eines einfachen Plugin Projekts in der Sprache TypeScript mit dem oben genannten Generator entsteht folgende Ordnerstruktur:

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

Automatisch generierte BeschreibungFür die Plugin Entwicklung am relevantesten sind hierbei die beiden markierten Dateien „package.json“ und „extension.ts“.

package.json wird als das „Extension Manifest“ bezeichnet und dient der Konfiguration des Plugins. Es enthält die Definitionen für ActivationEvents, Commands, Views, Menus, Keybindings und vieles mehr.

extension.ts ist die Datei die den eigentlichen Plugin-Code enthält. Sie enthält eine „activate“ Funktion die bei der ersten Aktivierung des Plugins ausgeführt wird. In den meisten Fällen wird in dieser Datei allerdings nur die Initialisierung aller für das Plugin nötigen Komponenten und zum Beispiel die Registrierung für verschiedene Events erledigt. Die eigentliche Logik befindet sich dann in den hier initialisierten Klassen. Bei der Aktivierung wird auch ein Objekt der „vscode.ExtensionContext“ Klasse übergeben. Über dieses Objekt kann auf die Schnittstellen der Extension API zugegriffen werden. Wann genau diese Aktivierung passiert, kann wiederum in der package.json Datei festgelegt werden.

* + 1. Funktionalität
       1. Ausführbarer Code (Commands)

Bei Visual Studio Code gibt es die Möglichkeit Plugin-definierten Code sozusagen „auf Befehl“ auszuführen. Diese Funktion eines Plugins wird passenderweise als „Command“ bezeichnet.

Um in einem Plugin einen Command einzubauen, muss dieser in der package.json Datei definiert werden. Dabei ist mindestens eine eindeutige Bezeichnung und ein Titel welcher später angezeigt wird festzulegen. Zusätzlich kann auch eine Kategorie, ein Icon, eine Kurzbezeichnung und eine Bedingung zu der der Command aktiv ist bestimmt werden. Welche der vorgenommenen Einstellungen für die Darstellung des Commands genutzt werden hängt unter anderem von dem Menü ab in dem der Command angezeigt wird.

Welcher Code ausgeführt wird kann bei der Aktivierung der Extension gesetzt werden. Dabei kann für jede (zuvor defin über die vscode.commands.registerCommand() oder die vscode.commands.registerTextEditorCommand() Funktion ein Callback angegeben, welches ausgeführt wird sobald der Command aufgerufen wird. Die register Funktionen retournieren ein Disposable Objekt welches beim ExtensionContext bekannt gegeben werden muss. Dieser kümmert sich dann um das Disposen der Commands wenn die Erweiterung deaktiviert werden sollte. /\*TODO evtl Ref Design Patterns GoF\*/

* + - 1. Persistente Speicherung (Data Storage, Settings, Secrets)
      2. Code Completion
      3. Benutzerdefinierte Sprachen
      4. Nutzerinteraktion
      5. Themes
      6. Debugger
      7. Unit Tests für den Plugin Code
  1. IntelliJ IDEA
     1. Das Plugin Projekt
        1. Setup eines Plugin Projekts
        2. Aufbau eines Plugins

Ausführbarer Code (Actions)

Persistente Speicherung (Data Storage, Settings, Secrets)

Event Listener

Services

Extension Points

Code Completion

Benutzerdefinierte Sprachen

Themes

Debugger?

Unit Tests für den Plugin Code

1. Codebeispiel: Recent Changes Plugin
2. Vergleich der Code-Metriken
3. Conclusion

Referenzen

<https://code.visualstudio.com/api>

<https://github.com/microsoft/vscode-extension-samples>

<https://plugins.jetbrains.com/docs/intellij/developing-plugins.html>

<https://github.com/JetBrains/intellij-sdk-code-samples/>

<https://github.com/JetBrains/intellij-community/tree/master>

<https://github.com/google/diff-match-patch>

<https://medium.com/cacher-app/building-code-editor-plugins-a-comparison-83b5c21657fe>

<https://survey.stackoverflow.co/2023/>

<https://insights.stackoverflow.com/survey>