Probe IPA

VZ Outlook Web-Add-In

Inhaltsverzeichnis

[**Teil 1 – Umfeld und Ablauf** 3](#_Toc156578842)

[1 Abstract (Kurzfassung) 3](#_Toc156578843)

[1.1 Ausgangssituation 3](#_Toc156578844)

[1.2 Umsetzung 3](#_Toc156578845)

[1.3 Ergebnis 3](#_Toc156578846)

[2 Aufgabenstellung (gemäss PKOrg) 3](#_Toc156578847)

[2.1 Ausgangslage 3](#_Toc156578848)

[2.2 Detaillierte Aufgabenstellung 3](#_Toc156578849)

[3 Vorarbeiten 4](#_Toc156578850)

[4 Vorkenntnisse 5](#_Toc156578851)

[5 Projektaufbauorganisation 6](#_Toc156578852)

[6 Backup 7](#_Toc156578853)

[7 Projektmanagementmethode 7](#_Toc156578854)

[8 Versionsverwaltung 8](#_Toc156578855)

[9 Zeitplan 8](#_Toc156578856)

[10 Arbeitsjournal 9](#_Toc156578857)

[**Teil 2 – Projektdokumentation** 11](#_Toc156578858)

[11 Informieren 11](#_Toc156578859)

[11.1 Projektanforderungen 11](#_Toc156578860)

[11.2 Bestehendes VZ Outlook Plug-In 11](#_Toc156578861)

[11.3 Projektumfeld: Systemgrenzen / Schnittstellen zur Aussenwelt 12](#_Toc156578862)

[11.3.1 Graph API 13](#_Toc156578863)

[12 Planen 14](#_Toc156578864)

[12.1 Realisierungskonzept 14](#_Toc156578865)

[12.2 Konzeptionelle Umsetzung 14](#_Toc156578866)

[12.3 Testkonzept 15](#_Toc156578867)

[13 Entscheiden 15](#_Toc156578868)

[13.1 Darstellung 15](#_Toc156578869)

[13.2 Authentifizierung 15](#_Toc156578870)

[14 Realisieren 15](#_Toc156578871)

[14.1 Erste Build-Only Pipeline einrichten 15](#_Toc156578872)

[14.2 Bestehendes Projekt implementieren 17](#_Toc156578873)

[14.2.1 DTOs implementieren 17](#_Toc156578874)

[15 Glossar 18](#_Toc156578875)

[16 Quellenverzeichnis 18](#_Toc156578876)

# **Teil 1 – Umfeld und Ablauf**

# Abstract (Kurzfassung)

## Ausgangssituation

Um die Abwesenheiten (Ferien, Krankheit/Unfall) zu erfassen, haben wir aktuell ein selbst geschriebenes Plugin in Outlook integriert. In Zukunft wollen wir jedoch unseren Exchange auf die Cloud migrieren, dann würde das bestehende Plugin jedoch nicht mehr funktionieren, da es für onPrem ausgelegt ist. Darum müssen wir nun ein neues VZ Outlook Web-Add-In schreiben, mit dem zugehörigem backend, welches die onPrem-User und die Cloud-User behandelt.

## Umsetzung

Das Projekt wird umgesetzt vom 17.01.2024 bis zum 02.02.2024. Dabei wird die Projektmanagementmethode IPERKA verwendet. Das Frontend wird mithilfe von ‘Office-Add-Ins’ erstellt mit dem Framework Blazor. Das Backend ist ein ASP .NET Core Web API. Als IDE wird Visual Studio 2022 verwendet.

## Ergebnis

Interne Benutzer sollten, sobald ihr Konto auf der Cloud ist, auf Outlook Web zugreifen können und das neue Web-Add-In verwenden können. Sie können nun ihre Abwesenheiten damit erfassen. Jedoch sollte das neue Web-Add-In auch weiterhin für die Benutzer funktionieren, welche noch nicht zur Cloud migriert wurden.

# Aufgabenstellung (gemäss PKOrg)

## Ausgangslage

Aktuell setzt das VZ das selbst programmierte Outlook Plugin «VZ» ein. Dieses funktioniert jedoch nur mit unserer onPrem Umgebung. Sobald das VZ die Exchange onPrem nach Exchange online migriert, wird das Plugin nicht mehr funktionieren. Ebenso wird das Plugin in Zukunft von Microsoft Office nicht mehr unterstützt.

## Detaillierte Aufgabenstellung

**Frontend**Das Frontend wird als Outlook Web-Add-In umgesetzt. Mit der Technologie von Blazor und dem Manifest-File von Microsoft. Um mit Blazor arbeiten zu können nutzen wir dieses Basisprojekt: <https://github.com/OfficeDev/Office-Add-in-samples/tree/main/Samples/blazor-add-in/outlook-blazor-add-in>.

**Backend**Es wird ein neues .NET API-Projekt erstellt. Darin soll das alte VZ.EmployeeService Projekt ebenfalls integriert werden, um weiterhin die OnPrem Funktionen zu unterstützen. Zusätzlich soll das Backend so erweitert werden, dass auch Exchange Online kompatibel ist. Am besten mit separaten Controllern/Requests trennen.

**Funktionen**

*Out of Office Ferien*

* Setzen der OoO Funktion
* Vorgegebener Text, Dynamisch zentral änderbar
* Start und Enddatum
* Deutsch, Italienisch & Französisch
* Stellvertreterfunktion (Textänderung beachten)

*Out of Office Krankheit / Unfall*

* Setzen der OoO im Team (Manager & Peers)
* Auslösen eines Mail an (Manager & Peers)
* Auslösen eines Mail ans HR (Testadresse)
  + Mitarbeiter MWae ist abwesend.
    - Bei Markus Wälchli wurde die Abwesenheitsmeldung bei Krankheit respektive Unfall erfasst (Erfasser: Same)
* Mails sollen mit der Adresse **OutOfOffice@vzch.com** verschickt werden

**Zugriff Graph API**

* Auslesen des Azure AD über Delegated Permissions
* Setzen vom OoO über Application Permissions

**Test-Users**

OoOUser@vzch.com

OoOManager@vzch.com

PW: Probeipa2024

Absender: **vzeemployeeservice@vzch.com**

**Azure AD App**

VZ employee Service

Application (client) ID: 5454fbf7-2aff-4f42-99ba-4676c3bfbc7d

Directory ID : 82cd7a1e-9b8e-4ad6-b17c-2130952f0e52

PW: SMk8Q~Ngl1~fy1Fi9ZxNq\_EVyQprXa9XmMSObcpz

**Admin Gruppe**

AAD-App-vzemployeeservicepro\_Admin

Zugriff Postfächer: [**https://outlook.office.com/**](https://outlook.office.com/)

# Vorarbeiten

***Vorarbeiten***

* Neues Repo und Branch im Git erstellt
* Initiale Projekte auf Git hochgeladen
* Zugriff auf alte IPAs von vorherigen Praktikanten und Lehrlingen
* Im Voraus erste Testversuche mit Graph API und Outlook Web-Add-Ins

# Vorkenntnisse

In den vergangenen 6 Monaten hat der Kandidat folgende Technologien fast täglich eingesetzt:

* C#
* Visual Studio 2022
* .NET 6.0 / .NET Core
* Angular / Typescript

**Arbeiten in den letzten 6 Monaten:**

* Welche Art von Arbeiten hat die/der Lernende im letzten Halbjahr durchgeführt?
  + Mitarbeiten in Scrum-Team mit der Rolle "Entwickler"
  + Diverse Anpassung in bestehenden Applikationen (.Net Core, ASP.Net Web API, MSSQL)
* Welches waren die zwei grössten Aufträge?
  + Historisierung bzw. Snapshot-Funktionalität in eine von uns Berechnete Auswertung einbauen.
  + Redesign unseres Kundenreportings (inkl. migration Crystal → Devexpress)
* Welche Tools wurden dafür eingesetzt?
  + Visual Studio 2022
  + Microsoft SQL Server Management Studio 18
  + Microsoft Azure Devops Server

# Projektaufbauorganisation

**Lehrbetrieb und Durchführungsort:**  
VZ VermögensZentrum AG  
Gotthardstrasse 6, 8002 Zürich  
044 207 27 27

**Kandidat:**   
Elion Bajrami  
[elion.bajrami@vzch.com](mailto:elion.bajrami@vzch.com)

**Lehrmeister:**   
Daniel Roth  
[daniel.roth@vzch.com](mailto:daniel.roth@vzch.com)

**Verantwortliche Fachkraft:**   
Luca Eggenberg  
[luca.eggenberg@vzch.com](mailto:luca.eggenberg@vzch.com)

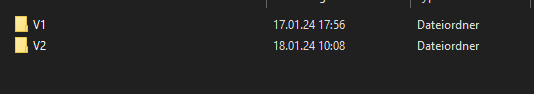
**Hauptexperte:** --

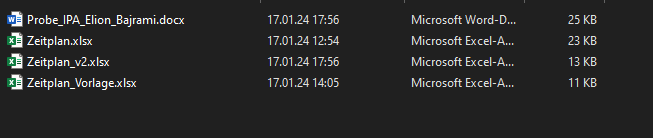
**Nebenexperte:** --

# Backup

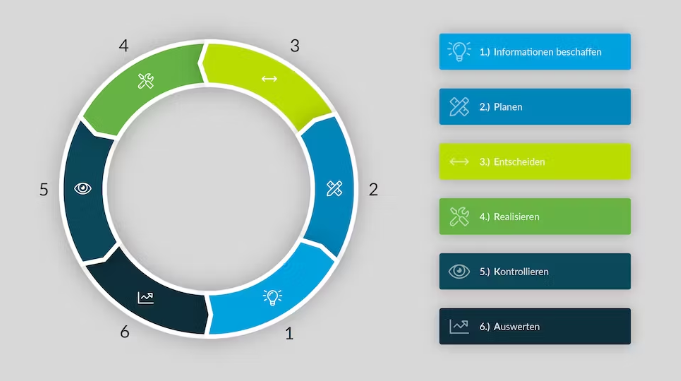
Alle Files die zum Bericht gehören werden in der Dokumentenablage abgelegt. Es wird für jeden Tag ein neuer Ordner erstellt, somit kann jederzeit auf die alten Versionen zugegriffen werden.

Die persönliche Dokumentenablage befindet sich auf einem Server, von dem täglich über Nacht automatisch ein Backup erstellt wird. Alle Dokumente und Dateien wie der IPA-Bericht, der aktuelle Zeitplan und Screenshots werden dort abgelegt.





# Projektmanagementmethode

Als Projektmanagementmethode wurde **IPERKA** gewählt. Durch diese Methode wird das Projekt in sechs Schritte aufgeteilt. Welche auch die Gliederung der Dokumentation im zweiten Teil darstellt.

# Versionsverwaltung

Git wird im Zusammenhang mit Azure DevOps als Versionsverwaltung verwendet. Für jedes Feature/PBI wird ein neuer Branch erstellt. Somit kann man beliebig entwickeln, ohne den Haupt-Branch zu zerstören. Man kann also jederzeit wieder auf die bestehende Version wechseln sollte der Feature-Branch nicht funktionieren.

Ausserdem hilft es einen besseren Überblick auf das ganze Projekt zu haben im Sinne davon, dass die einzelnen Features unabhängig voneinander implementiert werden. Das ist insbesondere hilfreich, wenn mehrere Features von verschiedenen Personen gleichzeitig umgesetzt werden. Aber auch wenn man wie in dieser IPA allein arbeitet hilft es, die Arbeit auf Branches aufzuteilen.

Der Main-Branch hat eine Policy mit der man es nicht möglich ist, auf den Branch zu pushen. Nur mit einem Pull Request ist es möglich den Branch zu verändern.

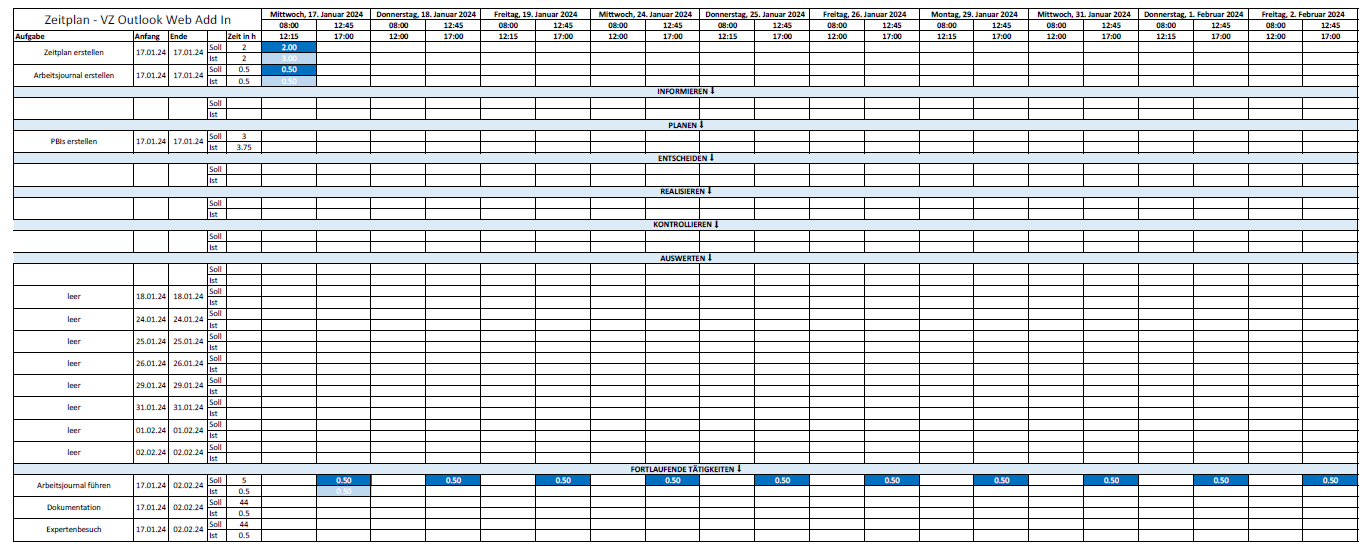
Sobald ein Feature/PBI abgeschlossen ist, wird ein Pull-Request erstellt, um den Feature-Branch in den Main-Branch zu mergen.

Um Sicherzustellen, dass immer alles gesichert ist, sollte man mindestens 1-mal pro Tag einen Commit erstellen und diesen auch pushen/hochladen ins git. Sollte nicht gepusht werden, kann die Gefahr bestehen, dass dieser verloren geht. Da er nur lokal auf dem PC existiert.

# Zeitplan

Starttermin: 17.01.2024

Enddatum: 02.02.2024

Mit Hilfe der Aufgabenstellung wurde folgender Zeitplan erstellt. Darin ist der Soll-ist Vergleich klar ersichtlich. Der Zeitplan ist aufgebaut im 2-Std-Raster (2h 15min), Blau ist die Soll-Zeit und hell-Blau die Ist-Zeit. Grün sind die Meilensteine, der erste Meilenstein ist das Abschliessen der Phase Entscheidung und der zweite das Abschliessen der Realisierung.

TODO vollständigen Zeitplan darstellen

# Arbeitsjournal

|  |  |
| --- | --- |
| Tägliches Arbeitsjournal, Tag 1, 17.01.2024 | |
| Geplante Ziele | * Zeitplan erstellen * Bericht erstellen und «Abstract», «Aufgabenstellung» und «Arbeitsjournal» erstellen |
| Tätigkeiten und erreichte Ziele | Der Zeitplan konnte initial erstellt werden. Danach habe ich mich an den Bericht gemacht und konnte die 3 Abschnitte erfolgreich beschrieben. |
| Aufgetretene Probleme | Der Zeitplan hat recht viel Zeit beansprucht : ( Pipeline Konfiguration unklar |
| Beanspruchte Hilfestellungen | Ich habe per YouTube Anleitung die Basis für meinen Zeitplan erstellt und dann nach meinen Wünschen erweitert: [Link zu Tutorial](https://www.youtube.com/watch?v=e86l9FNQGsU) |
| Vergleich mit dem Zeitplan/ Überzeiten | Der Zeitplan hat ein wenig mehr Zeit gekostet |
| Ungeplante Arbeiten | Es sind keine ungeplanten Arbeiten aufgetreten |
| Reflexion | Heute ging es nur um die Erstellung des Zeitplans und die Dokumentation. Ich erstellte zuerst den Zeitplan, damit ich ungefähr weiss, was als nächstes ansteht und wie lange. Jedoch war das nicht ganz einfach. Erstmal muss man sich alles gut vorausplanen und schätzen. Zusätzlich sollte es auch anständig aussehen, um die Übersicht nicht zu verlieren. |

|  |  |
| --- | --- |
| Tägliches Arbeitsjournal, Tag 2, 18.01.2024 | |
| Geplante Ziele | * Zeitplan verbessert * Dokumentation Teil 1 abschliessen was möglich * Informieren Teil schreiben * PBIs erfassen |
| Tätigkeiten und erreichte Ziele | Zuerst habe ich den Zeitplan überarbeitet. Dann den Teil 1 der Dokumentation wo möglich abgeschlossen. Und konnte nun mit der Projektdokumentation anfangen. Dort habe ich den Teil «Informieren» erledigt. Heute konnte ich noch die PBIs unter dem Abschnitt «Planen» erfassen. |
| Aufgetretene Probleme | Heute gab es keine Probleme :) |
| Beanspruchte Hilfestellungen | keine |
| Vergleich mit dem Zeitplan/ Überzeiten | Wird sind gut im Zeitplan drin |
| Ungeplante Arbeiten | keine |
| Reflexion | Heute konnte ich schon vieles erledigen und es lief zielorientierter als am ersten Tag, da der Zeitplan erstellt war. |

|  |  |
| --- | --- |
| Tägliches Arbeitsjournal, Tag 3, 19.01.2024 | |
| Geplante Ziele | * Planung abschliessen * Testkonzept erstellen * Entscheidung abschliessen * Pipeline konfigurieren * Altes Projekt implementieren |
| Tätigkeiten und erreichte Ziele | Ich konnte die Planung asbchliessen, danach ging es weiter mit dem Testkonzept. Sobald dies durch war kam die Entscheidung |
| Aufgetretene Probleme | Heute gab es keine Probleme, das alte Projekti ist jedoch mühsam |
| Beanspruchte Hilfestellungen | Keine |
| Vergleich mit dem Zeitplan/ Überzeiten | Wir sind gut im Zeitplan drin |
| Ungeplante Arbeiten | keine |
| Reflexion | Heute konnte ich schon vieles erledigen und es lief zielorientierter als am ersten Tag, da der Zeitplan erstellt war. |

|  |  |
| --- | --- |
| Tägliches Arbeitsjournal, Tag 4, 24.01.2024 | |
| Geplante Ziele | * Altes Projekt implementieren fertig * Frontend anfangen |
| Tätigkeiten und erreichte Ziele | Ich konnte heute das alte Backend im neuen Projekt implementieren, damit onPrem-User weiterhin unterstützt werden. Danach ging es ans erstellen des Frontends |
| Aufgetretene Probleme | Testing im Frontend |
| Beanspruchte Hilfestellungen | MWae um Hilfe gebeten, da Berechtigungen gefehlt haben, um das Add In hochzuladen. |
| Vergleich mit dem Zeitplan/ Überzeiten | Gut |
| Ungeplante Arbeiten | Viel Zeit gekostet bis optimale Testumgebung eingerichtet wurde |
| Reflexion | Ich konnte die Backend Funtkionen fürs onPrem alle übernehmen und konnte einen guten Fortschritt erzielen. Jedoch hat mich das Frontend recht viel Zeit gekostet. |

|  |  |
| --- | --- |
| Tägliches Arbeitsjournal, Tag 5, 25.01.2024 | |
| Geplante Ziele | * Frontend fertig implementieren |
| Tätigkeiten und erreichte Ziele | Heute konnte ich mein Ziel leider nicht erreichen. |
| Aufgetretene Probleme | Frontend Testing und implementieren hat probleme gemacht |
| Beanspruchte Hilfestellungen | MWae |
| Vergleich mit dem Zeitplan/ Überzeiten | :( |
| Ungeplante Arbeiten | Neue Lösungne suchen, welche auf allen Versionen funktioniert |
| Reflexion | Heute war kein effektiver Tag, ich konnte noch keine gute Lösung fürs Frontend finden. |

|  |  |
| --- | --- |
| Tägliches Arbeitsjournal, Tag 6, 26.01.2024 | |
| Geplante Ziele | * Frontend fertig implementieren |
| Tätigkeiten und erreichte Ziele | NICHTS |
| Aufgetretene Probleme | PROXY !!! |
| Beanspruchte Hilfestellungen | MWae |
| Vergleich mit dem Zeitplan/ Überzeiten | :( |
| Ungeplante Arbeiten | PROXY Lösung suchen |
| Reflexion | Heute war kein guter Tag, proxy hat leider Probleme gemacht. |
| Tägliches Arbeitsjournal, Tag 7, 29.01.2024 | |
| Geplante Ziele | * Proxy Probleme beheben |
| Tätigkeiten und erreichte Ziele | * Alternative Lösung gefunden |
| Aufgetretene Probleme | * Proxy |
| Beanspruchte Hilfestellungen | MWae |
| Vergleich mit dem Zeitplan/ Überzeiten | :( |
| Ungeplante Arbeiten | - |
| Reflexion | Wir konnten nun eine alternative Lösung finden, bis das Hauptproblem mit dem Proxy behoben ist. |

|  |  |
| --- | --- |
| Tägliches Arbeitsjournal, Tag 8, 31.01.2024 | |
| Geplante Ziele | * Neues Backend für onPrem mit bestehendem Frontend testen und vollständig implementieren |
| Tätigkeiten und erreichte Ziele | Ich konnte heute das Backend für onPrem vollständig implementieren sodass, das bestehende PlugIn weiterhin funktioniert auch mit dem neuen Backend |
| Aufgetretene Probleme | Codeversionen |
| Beanspruchte Hilfestellungen |  |
| Vergleich mit dem Zeitplan/ Überzeiten | :| |
| Ungeplante Arbeiten | keine |
| Reflexion | Heute war kein effektiver Tag, ich konnte noch keine gute Lösung fürs Frontend finden. |

# **Teil 2 – Projektdokumentation**

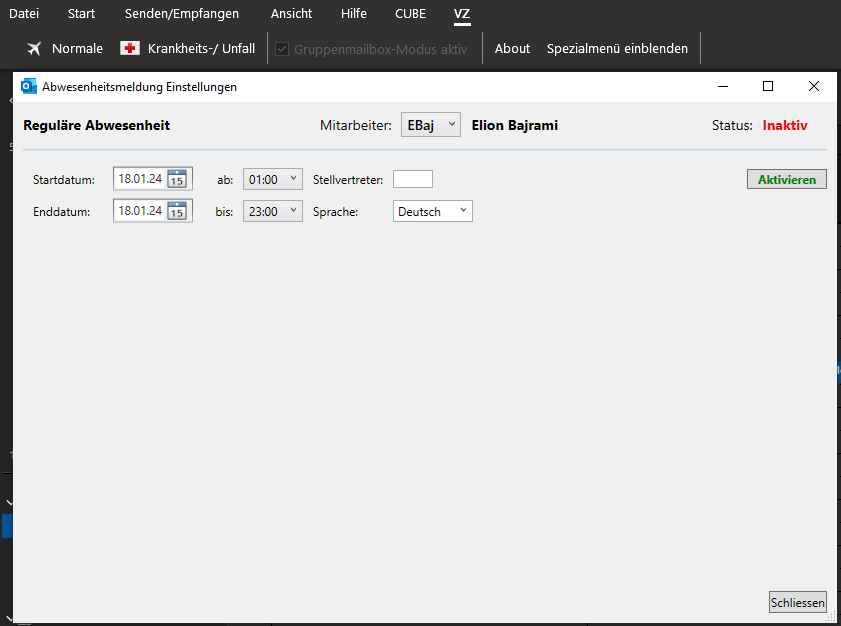
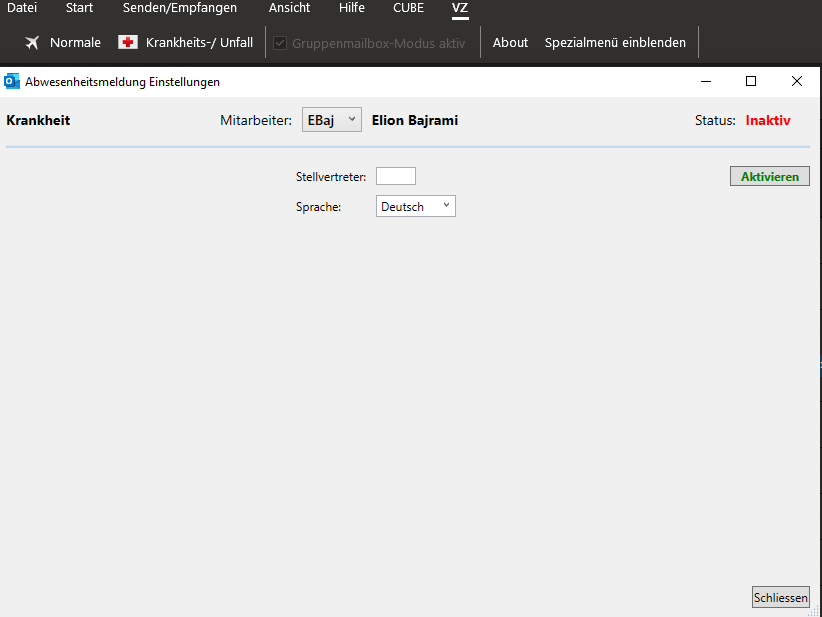
# Informieren

Die erste Phase der IPERKA-Methode ist Informieren. In diesem Schritt geht es darum, herauszufinden, worum es im Projekt genau geht. Was genau ist der Auftrag? Haben wir alle nötigen Informationen?

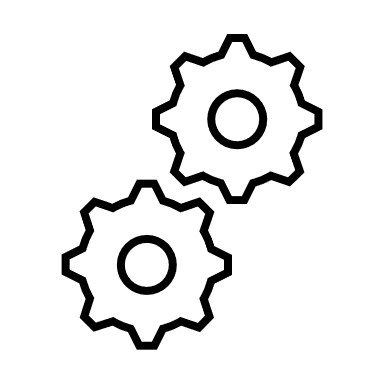
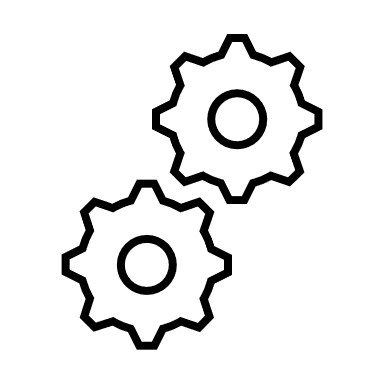
## Projektanforderungen

* Das Web-Add-In unterstützt onPrem sowie Cloud-User
* Das Backend prüft, ob der User bereits auf der Cloud ist.
* Sollte der User bereits auf der Cloud sein, verwenden wir die Graph API, um Abwesenheiten zu setzen.
* Das Web-Add-In soll es dem User ermöglichen, seine Abwesenheit für Ferien zu setzen. Das Start- und Enddatum, Sprache und Stellvertreter sollen gesetzt werden können. Ein Vorgesetzter hat die Möglichkeit, die Abwesenheit für seine Mitarbeiter zu setzten.
* Das Web-Add-In soll es dem User ermöglichen, seine Abwesenheit für Krankheit/Unfall zu setzen. Der Stellvertreter und die Sprache sollen gesetzt werden können. Der User hat zusätzlich die Möglichkeit die Abwesenheit für seine Mitarbeiter zu setzen.
* Beide Abwesenheiten sollten durch den User per Button bestätigt werden können.
* Der Status der Abwesenheit sollte ersichtlich sein.
* Um auf Graph API zugreifen zu können, muss entweder eine Delegated- oder Application-Permission vorhanden sein.
* Das Web-Add-In soll über Outlook aufrufbar sein.

## Bestehendes VZ Outlook Plug-In

So sieht es aktuell beim bestehenden VZ Outlook Plug-In für OnPrem aus:

## Projektumfeld: Systemgrenzen / Schnittstellen zur Aussenwelt



externe Schnittstelle

liefert Daten

liefert Daten

fordert Daten

fordert Daten

Graph API

API

Client



Client



### Graph API

Microsoft Graph ist eine RESTful-Web-API, mit der Sie auf Microsoft Cloud-Dienstressourcen zugreifen können. Nachdem wir einen Authentifizierungstoken für einen Benutzer oder einen Dienst abgerufen haben, können wir Requests and die Microsoft Graph-API tätigen.

#### Aufrufen einer REST-API-Methode

« {HTTP method} https://graph.microsoft.com/{version}/{resource}?{query-parameters} »

Der Request umfasst Folgendes:

* {http method} – Die HTTP-Methode, die im Request an Microsoft Graph API verwendet wird.
* {version} – Die Version der Microsoft Graph-API, die unsere Anwendung verwendet.
* {resource} – Die Ressource in Microsoft Graph-API, auf die wir verweisen wollen.
* {query-parameters} – Optionale Abfrageoptionen oder REST-Methodenparameter zur Anpassung der Antwort.
* {headers} – Anforderungsheader, die die Anforderung anpassen. Kann je nach API optional oder erforderlich sein.

# Planen

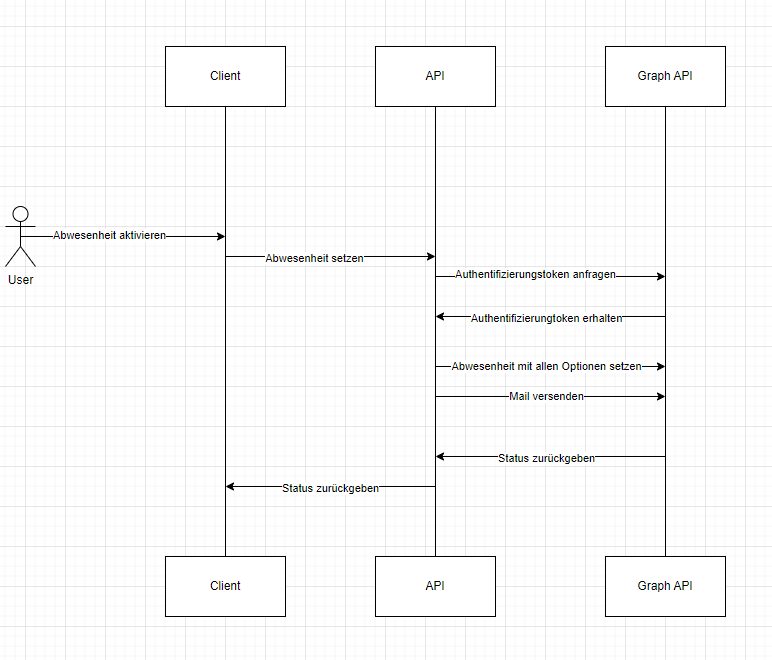
Das Planen ist die zweite Phase der IPERKA-Methode.

In dieser Phase werden die PBIs, das Realisierungskonzept und das Testkonzept erstellt.

## Realisierungskonzept

Das Frontend und das Backend wird in 2 separate Solutions erstellt. Für das Frontend verwenden war das bereits erwähnte Beispielprojekt «outlook-blazor-add-in» Darin befindet sich ein Blazor Projekt und ein Office Projekt, welches das Manifest File enthält. Für das Backend haben wir die bereits erstellte leere Solution «VZ.EmployeeService». Darin sollen 3 Projekte erstellt werden. Ein WebApi Projekt «VZ.EmployeeService» und zwei Class Libraries namens: «VZ.EmployeeService.Application» und «VZ.EmployeeService.Dto»

## Konzeptionelle Umsetzung



## Testkonzept

TODO

# Entscheiden

Das Entscheiden ist die dritte Phase der IPERKA-Methode.

## Darstellung

Für die Darstellung des Web-Add-Ins gibt es verschiedene Möglichkeiten. Man kann es aufrufen, wenn wir eine Mail schreiben, eine Mail lesen etc.

Da unser Web-Add-In eigentlich nichts mit Mails zu tun hat wollen wir es auch aufrufen können, ohne zuerst eine Mail zu öffnen. Daher wäre die beste Darstellung im Menü-Band von der «Startseite» oder wie bestehend, ein eigenes Menü-Band «VZ» erstellen.

## Authentifizierung

TODO ABKLÄREN

# Realisieren

In der vierten Phase der IPERKA-Methode haben wir Realisieren.

## Erste Build-Only Pipeline einrichten

trigger:

  branches:

    include:

      - refs/heads/master

      - refs/heads/feature/\*

      - refs/heads/dev

      - refs/heads/release/\*

parameters:

  - name: buildConfiguration

    displayName: BuildConfiguration

    type: string

    default: release

    values:

      - release

      - debug

  - name: projectsToPublish

    type: object

    default:

      - projectName: "ExchangeExtension EmployeeService"

        projectFilePath: "VZ.EmployeeService/VZ.EmployeeService/VZ.EmployeeService.csproj"

        buildOutput: ./publish/employeeService

      - projectName: "ExchangeExtension BlazorAddIn"

        projectFilePath: "VZ.Outlook.WebAddIn/outlook-blazor-add-in/BlazorAddIn.csproj"

        buildOutput: ./publish/blazorAddIn

      # TODO CHECK HOW IT WORKS WITH MANIFEST

      # - projectName: "ExchangeExtension Manifest"

      #   projectFilePath: "VZ.Outlook.WebAddIn/outlook-blazor-sideloader/outlook-blazor-sideloader.csproj"

      #   buildOutput: ./publish/manifest

name: $(date:yyyyMMdd)$(rev:.r)-$(Build.SourceBranchName)

jobs:

  - job: Job\_1

    displayName: ExchangeExtension-Build only

    pool:

      vmImage: "ubuntu-latest"

    steps:

      - checkout: self

        clean: true

      - task: NuGetAuthenticate@1

        displayName: NuGet Authenticate

        enabled: true

        inputs:

          nuGetServiceConnections: nugetFeedAzureDevOps

      - ${{ each project in parameters.projectsToPublish }}:

          - task: DotNetCoreCLI@2

            displayName: "Publish ${{ project.projectName }}"

            inputs:

              command: publish

              publishWebProjects: false

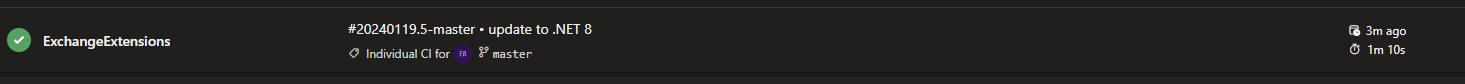
              projects: ${{ project.projectFilePath }}

              arguments: "-c ${{ parameters.buildConfiguration }} -o ${{ project.buildOutput }}"

              zipAfterPublish: false

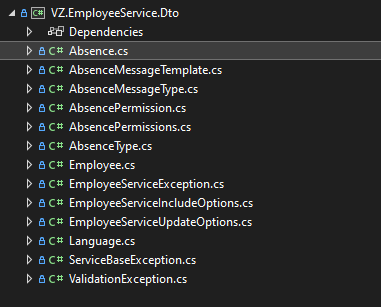
              modifyOutputPath: false

              configuration: "${{ parameters.buildConfiguration }}"



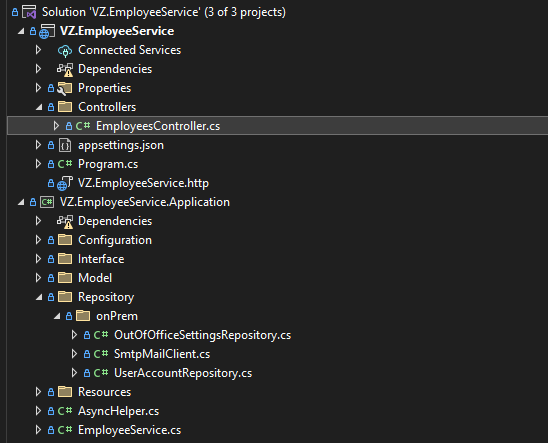
## Bestehendes Projekt implementieren

### DTOs implementieren



### Repository, Service und Controller implementieren

Repositories, Service und Contoller für die onPrem User wurden implementiert (die Funktionen wurden aus dem bestehenden Projekt übernommen):

s

# Glossar

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Quellenverzeichnis

Bild Iperka: <https://www.bexio.com/de-CH/blog/view/iperka-methode>

<https://learn.microsoft.com/de-de/office/dev/add-ins/overview/office-add-ins>

<https://github.com/AYIDouble/IPA-2018-Informatiker-EFZ-Applikationsentwicklung-Alpay-Yildirim?tab=readme-ov-file>