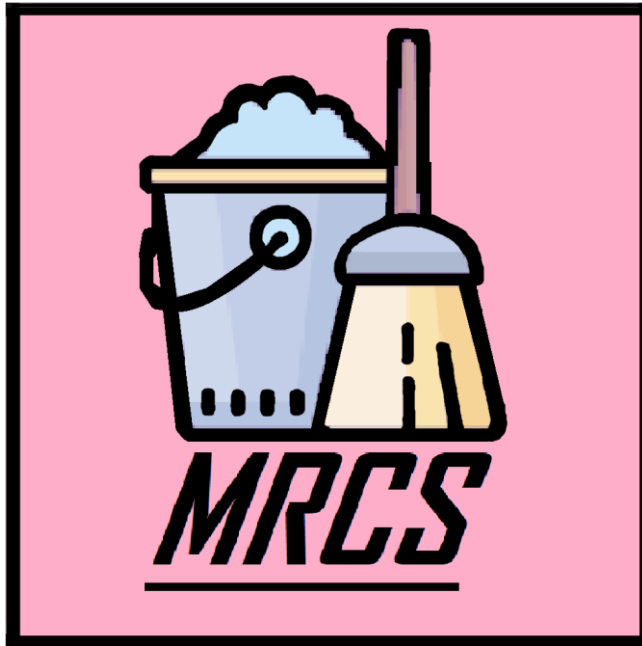


# MANAGEMENTPORTAL ZUR VERWALTUNG VON MRCS (MIXED REALITY CLEANING SYSTEM) – DATEN



Abschlussprüfung Winter 2019

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

## Prüfungsbewerber:

Lara Dörner

Breslauerstraße 5

73479 Ellwangen

29.06.1997



## Ausbildungsbetrieb:

PlanB. GmbH

Kocherstraße 15

73460 Hüttlingen



## Inhalt

1. Einleitung .....	3
1.1 Projektumfeld .....	3
1.2 Projektziel .....	4
1.3 Projektbegründung .....	4
1.4 Projektschnittstellen .....	4
1.5 Projektabgrenzung .....	4
1.6 Ansprechpartner .....	4
2. Projektplanung .....	4
2.1 Projektphasen und Ablaufplan.....	4
2.2 Ressourcenplanung .....	5
3. Durchführung .....	6
3.1 Ist-Analyse .....	6
3.2 Pflichtenheft .....	6
3.2.1 Musskriterien.....	6
3.2.2 Wunschkriterien .....	6
3.3 Prozessschritte .....	7
3.4 Auswahl des UI-Frameworks .....	8
3.5 Implementierungsphase .....	8
3.5.1 Servicebereitstellung .....	8
3.5.2 Projekterstellung.....	9
3.5.3 Implementierung des Backends .....	9
Zugriff auf die Datenbank.....	9
Zugriff auf Microsoft Azure Graph API.....	12
3.5.4 Implementierung des Frontends .....	15
3.5.5 Verwendete Technologien und Frameworks .....	16
3.6 Qualitätssicherung .....	16
4. Projektergebnis .....	17
4.1 Soll-/Ist-Vergleich .....	17
4.2 Abnahme.....	17
4.3 Fazit.....	18
5. Benutzerhandbuch.....	18
5.1 Login.....	18
5.2 Benutzerverwaltung.....	18
5.3 Navigation .....	20



5.4 Informationspunktverwaltung.....	20
6. Glossar .....	22
7. Tabellenverzeichnis.....	22
8. Codeverzeichnis .....	22
9. Modellverzeichnis .....	23
10. Abbildungsverzeichnis .....	23

# 1. Einleitung

## 1.1 Projektumfeld

Die PlanB. GmbH ist ein hochspezialisierter Anbieter von Beratungs- und Implementierungsdienstleistungen für anspruchsvolle IT – Infrastrukturlösungen und Anwendungen.

In den Bereichen Plattformen und Anwendungen für die Digitalisierung und verbundene digitale Transformation auf Basis von Microsoft Technologien zählt die PlanB. GmbH zu den technologieführenden Anbietern im deutschen Markt. Durch die Erschließung neuer Technologien sind wir in der Lage, wichtige technologische Neuerungen frühzeitig in die Planung unserer Kunden und in die Konzeption aktueller Projekte einfließen zu lassen. Für unsere Kunden bedeutet dies ein hohes Maß an Investitionssicherheit und die Möglichkeit zur langfristigen Zusammenarbeit mit einem strategisch orientierten und handelnden Beratungspartner. Die PlanB. GmbH begleitet eine Lösung im gesamten Lebenszyklus.



Abbildung 1: PlanB. Aufgabenbereiche

Das Managementportal zur Verwaltung von Mixed Reality Cleaning System (nachfolgend nur noch MRCS) – Daten ist derzeit ein internes Projekt und ermöglicht ein einfacheres Arbeiten mit dem MRCS. Durch das MRCS ist es möglich, einen Raum zu scannen und an gewünschten Punkten Informationspunkte zu setzen, um Informationen über Aufgaben an dieser Stelle zu speichern und auch wieder abzurufen. Das spätere Ziel des MRCS ist es, die professionelle Gebäudereinigung durch die gesetzten Informationspunkte zu vereinfachen, indem die gewünschten Reinigungen am Informationspunkt beschrieben werden.



## 1.2 Projektziel

Beim Managementportal zur Verwaltung von MRCS – Daten handelt es sich um eine Webapplikation, die es möglich macht, Benutzer zum MRCS hinzuzufügen und zu verwalten, genauso wie die Verwaltung der Informationspunkte.

Ziel ist es Benutzer und Informationspunkte hinzuzufügen, löschen und bearbeiten zu können, um somit eine professionelle Verwaltung der Daten zu ermöglichen.

## 1.3 Projektbegründung

Hintergrund des Managementportals für MRCS ist der Wunsch nach einer zentralen Verwaltungsplattform für das MRCS. Diese soll die Datenverwaltung des MRCS einfacher und device – unabhängig machen. Das Managementportal ist nur ein Teilprojekt der Produktentwicklung des MRCS.

## 1.4 Projektschnittstellen

Das Managementportal hat eine Verbindung zu einer Azure CosmosDB, um die Daten der Informationspunkte abrufen und aktualisieren zu können. Des Weiteren gibt es eine Verbindung zu einer Azure Graph API, welche die Benutzerverwaltung möglich macht.

Die Entwicklungsumgebung ist Visual Studio 2019 mit einem zusätzlich installierten WindowsAzure.Storage Paket und einem Microsoft.Identity.Client Paket, um sowohl auf die Azure CosmosDB als auch auf die Azure Graph API zugreifen zu können.

Die größte fachliche Schnittstelle bildet Felix Rohmeier, der sowohl [Product Owner](#) beim Managementportal als auch beim MRCS ist. Eine weitere Schnittstelle ist Liana Unsel, die als Entwicklerin beim MRCS tätig ist.

## 1.5 Projektabgrenzung

Das Managementportal zur Verwaltung von MRCS – Daten kann wie im Pflichtenheft beschrieben umgesetzt werden.

## 1.6 Ansprechpartner

Der Ansprechpartner für das Managementportal ist Herr Felix Rohmeier. Er ist Teil des MRCS – Teams und wird das Managementportal auch auf korrekte Umsetzung prüfen und abnehmen.

# 2. Projektplanung

## 2.1 Projektphasen und Ablaufplan

Das Managementportal soll innerhalb von 9 Tagen mit einem Aufwand von 8 Stunden pro Tag umgesetzt werden. Die Umsetzung des Projekts erfolgt immer in Absprache mit dem Product Owner, welcher den Zwischenstand am Ende jeder Phase abnimmt. Die Projektphasen sind in einzelne Tätigkeiten gegliedert und ergeben einen Gesamtaufwand von 70 Stunden.

Unterteilt ist die Entwicklung in die Phasen Vorbereitung, Implementierung, Qualitätssicherung und die Abnahme. Die folgende Tabelle zeigt die Zeitschätzung der einzelnen Tätigkeiten.

Phase	Tätigkeit	Aufwand in Stunden
1-Vorbereitung	Ist-/Soll-Analyse	1
	Pflichtenheft und Projektplan	3
	Auswahl des UI-Frameworks	2
2-Implementierung	Servicebereitstellung	4
	Backend Implementierung	25
	Frontend Implementierung	10
3-Qualitätssicherung	Testen	4
4-Abnahme	Besprechung und Abnahme	2
	Projektbericht und Dokumentation	19
<b>Summe</b>		<b>70</b>

Tabelle 1: Projektphasen

## 2.2 Ressourcenplanung

Für die Umsetzung des Managementportals werden verschiedene Ressourcen benötigt. Zu betrachten sind die Personalplanung, die Arbeitsmittel und die Microsoft Azure Cloud – Dienste.

### Arbeitsmittelplanung:

Arbeitsmittel	Stückzahl
Laptop	1
Arbeitsplatz	1
Internetanschluss	1
Lizenzen (Azure, Visual Studio, ...)	1

Tabelle 2: Arbeitsmittelplanung

### Personalplanung:

Name, Vorname	Position	Stundenlohn	Zeitaufwand in Stunden	Kosten
Dörrer, Lara	Entwickler	47,50€	70	3325€
Rohmeier, Felix	Product Owner	95€	5	475€
Unsold, Liana	Entwickler MRCS	47,50€	3	142,5€
<b>Summe</b>				<b>3.942,5€</b>

Tabelle 3: Personalplanung

### Microsoft Azure Ressourcen:

Microsoft Azure Dienst	Anzahl
Cosmos DB	1
Azure Graph API	1
Azure App Service	1

Tabelle 4: Ressourcenplanung

## 3. Durchführung

### 3.1 Ist-Analyse

Vor der Umsetzung des Managementportals konnten die Daten des MRCS nur bedingt verwaltet werden. Benutzer mussten manuell zum MRCS hinzugefügt werden. Die Informationspunkte konnten nur innerhalb der Microsoft Azure CosmosDB aktualisiert werden und nur innerhalb dieser Datenbank gesammelt angezeigt werden.

### 3.2 Pflichtenheft

#### 3.2.1 Musskriterien

Um die Verwaltung der Benutzer und Informationspunkte zu ermöglichen soll eine Webapplikation erstellt werden. Diese Webapplikation beinhaltet zum einen die Verwaltung der Benutzer und zum anderen die Verwaltung der Informationspunkte. Auf beiden Seiten sollen die vorhandenen Daten in einer Tabelle angezeigt werden. Innerhalb dieser Tabellen soll es möglich sein, nach bestimmten Datensätzen zu suchen. Des Weiteren soll es möglich sein die einzelnen Datensätze zu bearbeiten, zu löschen und neue Datensätze sollen erstellt werden können. Am Ende jeder Phase soll das Managementportal durch den Azure App Service veröffentlicht werden.

- Informationspunktverwaltung
  - Übersicht über die Informationspunkte
  - Neue Informationspunkte erstellen
  - Bestehende Informationspunkte bearbeiten und löschen
  - Bearbeitung über eine zusätzliche Bearbeitungsseite
- Benutzerverwaltung
  - Übersicht über alle Benutzer
  - Neue Benutzer hinzufügen
  - Benutzer bearbeiten und löschen
  - Bearbeitung über eine zusätzliche Bearbeitungsseite

#### 3.2.2 Wunschkriterien

Da es sich um eine Produktentwicklung im Anfangsstadium handelt soll das Managementportal möglichst simpel gehalten werden und nur die Informationspunktverwaltung und die Benutzerverwaltung beinhalten.

Dennoch wäre eine [OpenID Connect](#) Authentifizierung, um eine Authentifizierung der Nutzer des Managementportals zu bieten, wünschenswert.

### 3.3 Prozessschritte

Die einzelnen Projektphasen (siehe [2.1 Projektphasen und Ablaufplan](#)) wurden, um eine bessere Umsetzung zu ermöglichen, in weitere Teilschritte unterteilt. Daraus ergibt sich eine detaillierte Zeitplanung.

Tätigkeit	Arbeitsschritt	Aufwand in Stunden
1.Vorbereitung	1.1 Analyse der Anforderungen	1
	1.2 Erstellen des Pflichtenhefts und des Projektplans	2
	1.3 Auswahl eines UI-Frameworks	2
	1.4 Abstimmungen mit MRCS Entwicklungsteam und Abnahme durch Product Owner	1
2. Implementierung	2.1 Bereitstellung der benötigten Services und Ressourcen	4
	2.2 Implementierung des Backends	
	2.2.1 Verbindung zur Datenbank erstellen und <a href="#">CRUD</a> -Methoden erstellen	12
	2.2.2 Verbindung zur Azure Graph API herstellen und CRUD-Methoden erstellen	12
	2.2.3 Abstimmung mit Product Owner	1
	2.3 Implementierung des Frontends	
	2.3.1 <a href="#">HTML</a> -Masterlayout erstellen	4
	2.3.2 HTML-Übersichtsseiten erstellen	3
	2.3.3 HTML-Bearbeitungsseiten erstellen	2
	2.3.4 Abnahme durch Product Owner	1
3. Qualitätssicherung	3.1 Testen der Benutzerverwaltung	1.5
	3.2 Testen der Informationspunktverwaltung	1.5



	3.3 Abnahmetest mit dem Product Owner	1
4. Dokumentation	4.1 Erstellung der Dokumentation	
	4.1.1 Erstellen eins Layouts	1
	4.1.2 Verfassen der Dokumentation	11
	4.2 Erstellung des Benutzerhandbuchs	7
	4.3 Finale Abnahme durch Product Owner	2
<b>Summe</b>		<b>70</b>

Tabelle 5: Detaillierte Zeitplanung

### 3.4 Auswahl des UI-Frameworks

Da sowohl die Benutzerübersicht als auch die Informationspunktübersicht eine Tabelle beinhalten soll stellt sich die Frage, wie diese Tabelle aussehen soll. Die Tabellen sollen die Möglichkeit bieten, nach bestimmten Datensätzen zu suchen und die einzelnen Tabellenspalten auf – bzw. absteigend zu ordnen. Zudem soll das Framework keine zusätzlichen Kosten verursachen und kompatibel mit der ASP.Net Core Anwendung sein.

In Betrachtung dieser Kriterien fiel die Wahl auf das „DataTables“ Plug-In. Dieses erfüllt alle gewünschten Kriterien und ermöglicht es Tabellen einfach und schnell mit verschiedenen Konfigurationen umzusetzen. Dieses Plug-In verfügt über eine gute Dokumentation und kann dadurch einfach und schnell angewendet werden. Weitere Informationen zu „DataTables“ finden sich unter <https://datatables.net/>.

### 3.5 Implementierungsphase

#### 3.5.1 Servicebereitstellung

Um die Umsetzung des Managementportals zu ermöglichen müssen zu erst einige Vorbereitungen getroffen werden.

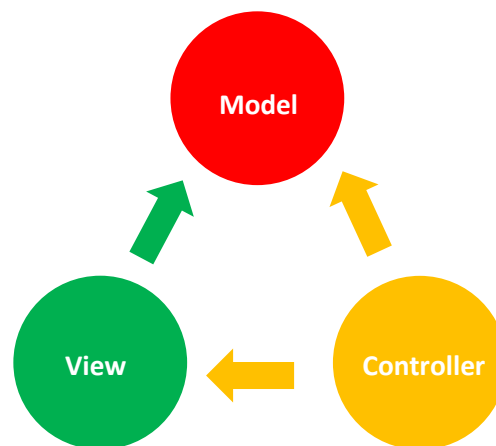
Die Informationspunkte sollen innerhalb der Microsoft Azure Cloud gespeichert werden, da sie sowohl vom Managementportal als auch vom MRCS abgerufen werden. Für die Speicherung der Informationspunkte wurde eine Azure Cosmos DB mit der Azure Tables API erstellt. Auf diese Datenbank können Managementportal und MRCS zugreifen und die Informationspunkte werden korrekt gespeichert. Um das Managementportal letztendlich veröffentlichen zu können wird ein Azure App Service benötigt, welcher ebenfalls erstellt wird.

Das Managementportal des MRCS und das MRCS selbst ist momentan noch in der vertraulichen Produktentwicklung. Geplant ist die beiden White Label Anwendungen später mit einem Kunden zu vermarkten. Um dies zu gewährleisten muss die Benutzerverwaltung als Azure Active Directory B2C erstellt werden. Das Azure Active Directory B2C ist eine Kunden Identitäts- und Zugriffsverwaltungs Services für

Anwendungen und bietet schon im Standard, Funktionen wie zum Beispiel Registrieren und Anmelden. Weiterhin beinhaltet das B2C auch die notwendige Azure Active Directory Graph API. Die Azure Active Directory Graph API ermöglicht den Zugriff auf das Azure Active Directory B2C über [REST-API](#) Endpunkte und bietet somit die grundlegenden Funktionen, um die Benutzerverwaltung im Managementportal mithilfe von Funktionen und Querys umzusetzen.

### 3.5.2 Projekterstellung

Das Projekt wird mit Visual Studio 2019 umgesetzt und als Projekt im ASP.NET Core Framework erstellt. Das ASP.NET Core Framework beruht auf dem MVC-Konzept, welches eine Anwendung in drei Gruppen einteilt: Models, Views und Controller. Diese Gruppen sind untereinander verbunden und stehen in Beziehung miteinander.



Modell 1: MVC-Konzept

### 3.5.3 Implementierung des Backends

#### Zugriff auf die Datenbank

Um den Zugriff auf die Datenbank herzustellen muss eine Verbindung zur Microsoft Azure CosmosDB hergestellt werden. Um diese Verbindung herzustellen wird der AccountName, der Accountkey und der Tableendpoint der Datenbank benötigt. Nachdem die Verbindung hergestellt wurde können nun die CRUD – Methoden ausgeführt werden.

- GetAnchorAsync  
Diese Methode erwartet als Parameter den Partitionkey und den Rowkey des gewünschten Informationspunkt. Dieser Informationspunkt wird von der Datenbank abgerufen und als Objekt der Klasse „Anchor“ zurückgegeben. Die Klasse „Anchor“ beinhaltet die Properties „Key“, „Name“, „Description“, „RowKey“ und „PartitionKey“. Diese Methode wird aufgerufen, sobald ein einzelner Informationspunkt bearbeitet werden soll.



```

/// <summary>
/// Gets Anchor from Database
/// </summary>
/// <param name="partitionKey"></param>
/// <param name="rowKey"></param>
/// <returns>Selected Anchor</returns>
1-Verweis | 0 Ausnahmen
public async Task<Anchor> GetAnchorAsync(string partitionKey, string rowKey)
{
    TableOperation retrieveOperation = TableOperation.Retrieve<AnchorCacheEntity>(partitionKey, rowKey);
    TableResult result = await dbCache.ExecuteAsync(retrieveOperation);
    AnchorCacheEntity anchorEntity = result.Result as AnchorCacheEntity;
    Anchor anchor = new Anchor();
    anchor.Name = anchorEntity.Name;
    anchor.Key = anchorEntity.AnchorKey;
    anchor.Description = anchorEntity.Description;

    return anchor;
}

```

Codeauszug 1: Einzelnen Informationspunkt abrufen

- GetAllAnchorsAsync

Dieser Methode werden keine Parameter übergeben. Die Methode ruft alle bestehenden Informationspunkte ab und gibt sie in einer Liste zurück.

```

/// <summary>
/// Gets all Anchors
/// </summary>
/// <returns>All Anchors from Database</returns>
2 Verweise | 0 Ausnahmen
public async Task<List<AnchorCacheEntity>> GetAllAnchorAsync()
{
    await this.InitializeAsync();

    List<AnchorCacheEntity> results = new List<AnchorCacheEntity>();
    TableQuery<AnchorCacheEntity> tableQuery = new TableQuery<AnchorCacheEntity>();
    TableQuerySegment<AnchorCacheEntity> previousSegment = null;
    while (previousSegment == null || previousSegment.ContinuationToken != null)
    {
        TableQuerySegment<AnchorCacheEntity> currentSegment =
            await this.dbCache.ExecuteQuerySegmentedAsync<AnchorCacheEntity>(tableQuery, previousSegment?.ContinuationToken);
        previousSegment = currentSegment;
        results.AddRange(previousSegment.Results);
    }
    return results;
}

```

Codeauszug 2: Alle Informationspunkte abrufen

- CreateAnchorAsync

Als Parameter wird dieser Methode ein Objekt der Klasse „Anchor“ übergeben. Die Methode berechnet die Id dieses Informationspunktes anhand der Id des letzten Punktes und fügt diesen dann in die Datenbank ein.



```

/// <summary>
/// Creates Anchor
/// </summary>
/// <param name="anchor"></param>
/// <returns>AnchorNumber</returns>
1-Verweis | 0 Ausnahmen
public async Task<long> CreateAnchorAsync(Anchor anchor)
{
    await this.InitializeAsync();

    if (lastAnchorNumberIndex == long.MaxValue)
    {
        lastAnchorNumberIndex = -1;
    }
    if (lastAnchorNumberIndex < 0)
    {
        var anchors = await this.GetAllAnchorAsync();
        if(anchors.Count != 0)
        {
            var Lastanchor = anchors[anchors.Count - 1];
            var rowKey = Lastanchor.RowKey;
            long.TryParse(rowKey, out lastAnchorNumberIndex);
        }
        else
        {
            lastAnchorNumberIndex = -1;
        }
    }

    long newAnchorNumberIndex = ++lastAnchorNumberIndex;

    AnchorCacheEntity anchorEntity = new AnchorCacheEntity(newAnchorNumberIndex, CosmosDbCache.partitionSize, anchor)
    {
        Anchor = anchor
    };

    await this.dbCache.ExecuteAsync(TableOperation.Insert(anchorEntity));
    return newAnchorNumberIndex;
}

```

Codeauszug 3: Informationspunkt erstellen

- DeleteAnchorAsync

Durch die an diese Methode übergebenen Parameter, „PartitionKey“ und „RowKey“, wird der Informationspunkt abgerufen und aus der Datenbank gelöscht.

```

/// <summary>
/// Deletes selected Anchor
/// </summary>
/// <param name="partitionKey"></param>
/// <param name="rowKey"></param>
1-Verweis | 0 Ausnahmen
public async void DeleteAnchorAsync(string partitionKey, string rowKey)
{
    TableOperation retrieveOperation = TableOperation.Retrieve<AnchorCacheEntity>(partitionKey, rowKey);
    TableResult result = await dbCache.ExecuteAsync(retrieveOperation);
    AnchorCacheEntity anchorEntity = result.Result as AnchorCacheEntity;
    TableOperation deleteOperation = TableOperation.Delete(anchorEntity);
    await dbCache.ExecuteAsync(deleteOperation);
}

```

Codeauszug 4: Informationspunkt löschen

- UpdateAnchorAsync

Das übergebene Objekt der Klasse „Anchor“ beinhaltet die Änderungen des Informationspunktes. Anhand der fixen Id wird der gewünschte Informationspunkt mit den neuen Werten aktualisiert und in der Datenbank gespeichert.



```

/// <summary>
/// Updates selected Anchor
/// </summary>
/// <param name="anchor"></param>
1-Verweis | 0 Ausnahmen
public async void UpdateAnchorAsync(Anchor anchor)
{
    TableOperation retrieveOperation = TableOperation.Retrieve<AnchorCacheEntity>(anchor.PartitionKey, anchor.RowKey);
    TableResult result = await dbCache.ExecuteAsync(retrieveOperation);
    AnchorCacheEntity anchorEntity = result.Result as AnchorCacheEntity;

    anchorEntity.Name = anchor.Name;
    anchorEntity.AnchorKey = anchor.Key;
    anchorEntity.Description = anchor.Description;
    TableOperation insertOrMergeOperation = TableOperation.InsertOrMerge(anchorEntity);

    TableResult resultUpdate = await dbCache.ExecuteAsync(insertOrMergeOperation);
    AnchorCacheEntity updatedAnchor = result.Result as AnchorCacheEntity;
}

```

Codeauszug 5: Informationspunkt aktualisieren

### Zugriff auf Microsoft Azure Graph API

Damit durch das Managementportal Änderungen in der Microsoft Azure Graph API vorgenommen werden können muss mithilfe der ClientId, des ClientSecrets und des Tenants eine Verbindung aufgebaut werden. Danach können Benutzer mithilfe der CRUD-Methoden bearbeitet werden.

- GetAllUsers

Hier wird ein Get-Request an die Microsoft Azure Graph API gesendet und alle Benutzer werden abgerufen und zurückgegeben.

```

/// <summary>
/// Sends Get-Request
/// </summary>
/// <param name="api"></param>
/// <returns>Response Content</returns>
2 Verweise | 0 Ausnahmen
public async Task<string> SendGraphGetRequest(string api)
{
    AuthenticationResult result = await authContext.AcquireTokenAsync("https://graph.windows.net", credential);

    HttpClient http = new HttpClient();
    string url = "https://graph.windows.net/" + tenant + api + "?" + Globals.aadGraphVersion;

    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, url);
    request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", result.AccessToken);
    HttpResponseMessage response = await http.SendAsync(request);

    if (!response.IsSuccessStatusCode)
    {
        string error = await response.Content.ReadAsStringAsync();
        object formatted = JsonConvert.DeserializeObject(error);
        throw new WebException("Error Calling the Graph API: \n" + JsonConvert.SerializeObject(formatted, Formatting.Indented));
    }

    return await response.Content.ReadAsStringAsync();
}

```

Codeauszug 6: Graph API Get-Request

- GetUser

Die Methode ruft mithilfe eines Get-Requests einen einzelnen Benutzer ab. An diese Methode wird die Benutzer-ID übergeben, welche im Request mitgegeben wird, um den richtigen Benutzer abzurufen.



```

/// <summary>
/// Sends Get-Request
/// </summary>
/// <param name="api"></param>
/// <returns>Response Content</returns>
2 Verweise | 0 Ausnahmen
public async Task<string> SendGraphGetRequest(string api)
{
    AuthenticationResult result = await authContext.AcquireTokenAsync("https://graph.windows.net", credential);

    HttpClient http = new HttpClient();
    string url = "https://graph.windows.net/" + tenant + api + "?" + Globals.aadGraphVersion;

    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, url);
    request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", result.AccessToken);
    HttpResponseMessage response = await http.SendAsync(request);

    if (!response.IsSuccessStatusCode)
    {
        string error = await response.Content.ReadAsStringAsync();
        object formatted = JsonConvert.DeserializeObject(error);
        throw new WebException("Error Calling the Graph API: \n" + JsonConvert.SerializeObject(formatted, Formatting.Indented));
    }

    return await response.Content.ReadAsStringAsync();
}

```

Codeauszug 7: Graph API Get-Request

- CreateUser

An diese Methode wird ein Objekt der Klasse „UserViewModel“ übergeben. Diese Klasse beinhaltet Properties, die den Benutzer näher beschreiben. Dieses Objekt wird zu einem Json-Objekt konvertiert und anschließend mithilfe eines Post-Requests zur Microsoft Azure Graph API hinzugefügt.

```

/// <summary>
/// Sends Post-Request
/// </summary>
/// <param name="api"></param>
/// <param name="json"></param>
/// <returns>Response Content</returns>
1-Verweis | 0 Ausnahmen
private async Task<string> SendGraphPostRequest(string api, string json)
{
    AuthenticationResult result = await authContext.AcquireTokenAsync(Globals.aadGraphResourceId, credential);
    HttpClient http = new HttpClient();
    string url = Globals.aadGraphEndpoint + tenant + api + "?" + Globals.aadGraphVersion;

    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Post, url);
    request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", result.AccessToken);
    request.Content = new StringContent(json, Encoding.UTF8, "application/json");
    HttpResponseMessage response = await http.SendAsync(request);

    if (!response.IsSuccessStatusCode)
    {
        string error = await response.Content.ReadAsStringAsync();
        object formatted = JsonConvert.DeserializeObject(error);
        throw new WebException("Error Calling the Graph API: \n" + JsonConvert.SerializeObject(formatted, Formatting.Indented));
    }

    return await response.Content.ReadAsStringAsync();
}

```

Codeauszug 8: Graph API Post-Request

- UpdateUser

Dieser Methode wird die Benutzer-ID, sowie ein Objekt der Klasse „UserViewModel“ übergeben. Im Objekt werden die Änderungen am Benutzer übergeben und die Benutzerid dient dazu, die Änderungen am gewünschten Benutzer vorzunehmen. Die Änderungen werden dann mithilfe eines Patch-Requests in der Microsoft Azure Graph API vorgenommen.



```

/// <summary>
/// Sends Patch-Request
/// </summary>
/// <param name="api"></param>
/// <param name="json"></param>
/// <returns>Response Content</returns>
1-Verweis | 0 Ausnahmen
private async Task<string> SendGraphPatchRequest(string api, string json)
{
    AuthenticationResult result = await authContext.AcquireTokenAsync(Globals.aadGraphResourceId, credential);
    HttpClient http = new HttpClient();
    string url = Globals.aadGraphEndpoint + tenant + api + "?" + Globals.aadGraphVersion;

    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Patch, url);
    request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", result.AccessToken);
    request.Content = new StringContent(json, Encoding.UTF8, "application/json");
    HttpResponseMessage response = await http.SendAsync(request);

    if (!response.IsSuccessStatusCode)
    {
        string error = await response.Content.ReadAsStringAsync();
        object formatted = JsonConvert.DeserializeObject(error);
        throw new WebException("Error Calling the Graph API: \n" + JsonConvert.SerializeObject(formatted, Formatting.Indented));
    }

    return await response.Content.ReadAsStringAsync();
}

```

Codeauszug 9: Graph API Patch-Request

- DeleteUser

Die Benutzer-ID, die der Methode übergeben wird, wird mittels eines Delete-Requests an die Microsoft Azure Graph API weitergeleitet und der Benutzer wird gelöscht.

```

/// <summary>
/// Sends Delete-Request
/// </summary>
/// <param name="api"></param>
/// <returns>Response Content</returns>
1-Verweis | 0 Ausnahmen
private async Task<string> SendGraphDeleteRequest(string api)
{
    AuthenticationResult result = await authContext.AcquireTokenAsync(Globals.aadGraphResourceId, credential);
    HttpClient http = new HttpClient();
    string url = Globals.aadGraphEndpoint + tenant + api + "?" + Globals.aadGraphVersion;
    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Delete, url);
    request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", result.AccessToken);
    HttpResponseMessage response = await http.SendAsync(request);

    if (!response.IsSuccessStatusCode)
    {
        string error = await response.Content.ReadAsStringAsync();
        object formatted = JsonConvert.DeserializeObject(error);
        throw new WebException("Error Calling the Graph API: \n" + JsonConvert.SerializeObject(formatted, Formatting.Indented));
    }

    return await response.Content.ReadAsStringAsync();
}

```

Codeauszug 10: Graph API Delete-Request

In diesem Teil der Implementierung wurde auch das Wunschkriterium der OpenID Connect Authentifizierung umgesetzt. Hierzu wurde eine Verbindung zum Azure Active Directory erstellt. Mithilfe dieser Verbindung können die User authentifiziert werden und das Managementportal nach erfolgreicher Authentifizierung nutzen. Durch die Umsetzung des Wunschkriteriums ist es nun möglich sich im Managementportal an- und abzumelden.



### 3.5.4 Implementierung des Frontends

Die Benutzeroberfläche des Managementportals ist in einem einheitlichen und schlichten Design gehalten. Das Frontend wurde mit HTML, [CSS](#) und [JavaScript](#) umgesetzt. Insgesamt besteht das Frontend aus vier CSHTML-Seiten und einer Layoutseite. CSHTML-Dateien sind HMTL-Webseiten, die von ASP.NET Core für die Erstellung von Webseiten für Browser verwendet werden.

Auf jeder CSHMTL-Seite gibt es ein Menü, einen Header und einen Inhaltsbereich. Das Menü und der Header werden von der Layoutseite vorgegeben.

```
//Implementation Navigation
<div class="mySidenav" onmouseover="moveBody()" onmouseout="expandBody()">
  <a id="Benutzer" asp-area="" asp-controller="Home" asp-action="Index">
    <i class="glyphicon glyphicon-user"></i>
  </a>
  <a id="Points" asp-area="" asp-controller="InfoPunkt" asp-action="Index">
    <i class="glyphicon glyphicon-record"></i>
  </a>
  <a id="Logout" asp-area="AzureAD" asp-controller="Account" asp-action="SignOut">
    <i class="glyphicon glyphicon-log-out"></i>
  </a>
</div>
```

Codeauszug 11: Umsetzung Menü

Auf den Übersichtsseiten der beiden Bereiche wird jeweils eine Tabelle dargestellt, die alle bestehenden Daten beinhaltet. Diese Tabelle wurde mit dem Plug-In „DataTables“ in JavaScript umgesetzt. Das Plug-In ermöglicht es, Tabellen mit Features, wie z.B. einer Suchleiste, zu versehen.

```
$.ajax({
  type: "GET",
  url: "InfoPunkt/GetAllAnchors",
  contentType: "application/json; charset=utf-8",
  dataType: "json",
  success: function (result) {
    var JsonResult = result;
    var AnchorTable = $("#Anchors").dataTable({
      "processing": false,
      "ordering": true,
      "paging": true,
      "data": JsonResult,
      "columns": [
        { "data": "anchorKey" },
        { "data": "name" },
        { "data": "description" },
        {
          "data": null, "render": function(data) {
            return '<input type="button" class="ActionButtons" value="Edit" onClick="openEdit(\'' + data.partitionKey + '\';' + data.rowKey + '\')' />' +
              '<input class="ActionButtons" type="button" value="Delete" onClick="Delete(\'' + data.partitionKey + '\';' + data.rowKey + '\')' />';
          }
        }
      ],
    });
  },
  error: function (response) {
    alert('error');
  }
});
```

Codeauszug 12: Tabellenimplementierung



Innerhalb dieser Tabellen befinden sich hinter jedem Datensatz HTML-Buttons, um den jeweiligen Datensatz zu bearbeiten oder zu löschen. Wird die Option „Edit“ ausgewählt, so öffnet sich eine neue Seite, die Textboxen zum Bearbeiten der einzelnen Attribute des Datensatzes bietet. Diese Textboxen sind, je nach Attribut, schreibgeschützt oder bearbeitbar. Um die Textboxen schreibgeschützt zu erstellen wird sie als „readonly“ erstellt.

```
<div class="col-sm-5 text-right">
    @Html.TextBoxFor(m => m.RowKey, new { @class = "form-control", style = "width:500px;" , @readonly = true})
</div>
```

Codeauszug 13: Implementierung Textbox

Am Ende der Seite lassen sich zwei Buttons finden, mit welchen man die Bearbeitung verlassen oder die Änderungen speichern kann.

Oberhalb der Tabellen befindet sich jeweils ein HTML-Button, welcher es ermöglicht neue Datensätze zu erstellen. Die Buttons werden als HTML Input erstellt und rufen mithilfe der onClick-Funktion eine JavaScript Funktion auf, die eine neue Seite öffnet.

```
<input type="button" value="Create" class="ActionButtons" onclick="CreatePoint()" />
```

Codeauszug 14: Buttonimplementierung Informationspunkt

```
<input type="button" value="Create" class="ActionButtons" onclick="CreateUser()" />
```

Codeauszug 15: Buttonimplementierung Benutzer

### 3.5.5 Verwendete Technologien und Frameworks

Das Managementportal wurde als ASP.Net Core Projekt mit C# und JavaScript umgesetzt. Die Azure Cosmos DB zur Speicherung der Informationspunkte und die Graph API zur Benutzerverwaltung sind Microsoft Azure Dienste. Die Azure Graph API liegt im Azure Active Directory B2C, welches eine Authentifizierung der Nutzer ermöglicht.

Um das Managementportal wie beschrieben umsetzen zu können wurden zusätzlich zu den Basis Frameworks folgende Frameworks genutzt:

- System.Linq
- Microsoft.AspNetCore.Mvc
- Newtonsoft.Json
- Microsoft.WindowsAzure.Storage
- Microsoft.WindowsAzure.Storage.Table
- Microsoft.IdentityModel.Clients.ActiveDirectory

### 3.6 Qualitätssicherung

Durch Testen wird sichergestellt, dass das Managementportal einwandfrei funktioniert und die Qualität wird garantiert. Zu Beginn wurden alle Funktionen der Benutzerverwaltung getestet. Hierbei wurden verschiedene Benutzer hinzugefügt, Lara Dörner

bearbeitet und am Ende gelöscht. Hier traten anfangs Probleme beim Updaten von bestimmten Benutzerattributen auf, die daraufhin ausgebessert wurden.

Danach wurden die Funktionen der Informationspunkte getestet. Auch hier wurden Informationspunkte erstellt, bearbeitet und gelöscht. Innerhalb dieser Tests traten keine Probleme auf.

Durch diese Tests wurde das Managementportal ausführlich geprüft und Fehler wurden ausgebessert.

## 4. Projektergebnis

### 4.1 Soll-/Ist-Vergleich

Die folgende Tabelle stellt die [Muss-Kriterien](#) und deren Umsetzung dar. Die Umsetzung der Muss-Kriterien verlief problemlos und das Managementportal konnte wie vorgesehen umgesetzt werden.

Muss-Kriterium	Umsetzung
<b>Informationspunktverwaltung</b>	
Bestehende Informationspunkte darstellen	Darstellung in einer HTML-Tabelle mit "DataTables" Plug-In
Neue Informationspunkte erstellen	HTML-Seite mit Textboxen für Attribute
Vorhandene Informationspunkte bearbeiten	HTML-Button, der auf Bearbeitungsseite weiterleitet
Vorhandene Informationspunkte löschen	HTML-Button, der Delete-Methode aufruft
Bearbeitungsseite für Informationspunkte	HTML-Seite mit ausgefüllten Textboxen für Attribute
<b>Benutzerverwaltung</b>	
Bestehende Benutzer darstellen	Darstellung in einer HTML-Tabelle mit "DataTables" Plug-In
Neue Benutzer hinzufügen	HTML-Button, der HTML-Seite aufruft
Vorhandene Benutzer bearbeiten	Weiterleitung auf Bearbeitungsseite durch HTML-Button
Vorhandene Benutzer löschen	Aufrufen der Delete-Methode durch HTML-Button
Bearbeitungsseite für Benutzer	HTML-Seite mit den Werten des Benutzers

Tabelle 6: Soll-/Ist-Vergleich

### 4.2 Abnahme

Das Managementportal wurde wie vorgesehen umgesetzt und wurde durch den Projektleiter abgenommen. Die Abnahme erfolgte nach jedem Projektschritt durch den Product Owner.

### 4.3 Fazit

Das Managementportal erleichtert die Arbeit mit dem MRCS, da es dadurch möglich wird Änderungen vorzunehmen. Die Umsetzung konnte wie geplant vorgenommen werden und alle Muss-Kriterien wurden umgesetzt. Die Produktentwicklung des MRCS in Verbindung mit dem Managementportal soll in Zukunft zusammen mit einem Kunden aus dem Bereich der professionellen Reinigung weiterentwickelt werden.

## 5. Benutzerhandbuch

Um das MRCS bestmöglich nutzen zu können ist eine Verwendung des Managementportals nötig. Das Managementportal verfügt bisher über die Verwaltung der Benutzer und die Verwaltung der Informationspunkte.

### 5.1 Login

Beim Aufruf der Seite werden Sie zunächst aufgefordert sich anzumelden. Hierzu müssen Sie die E-Mail-Adresse, mit welcher Sie im Managementportal registriert sind, und das Passwort eingeben.

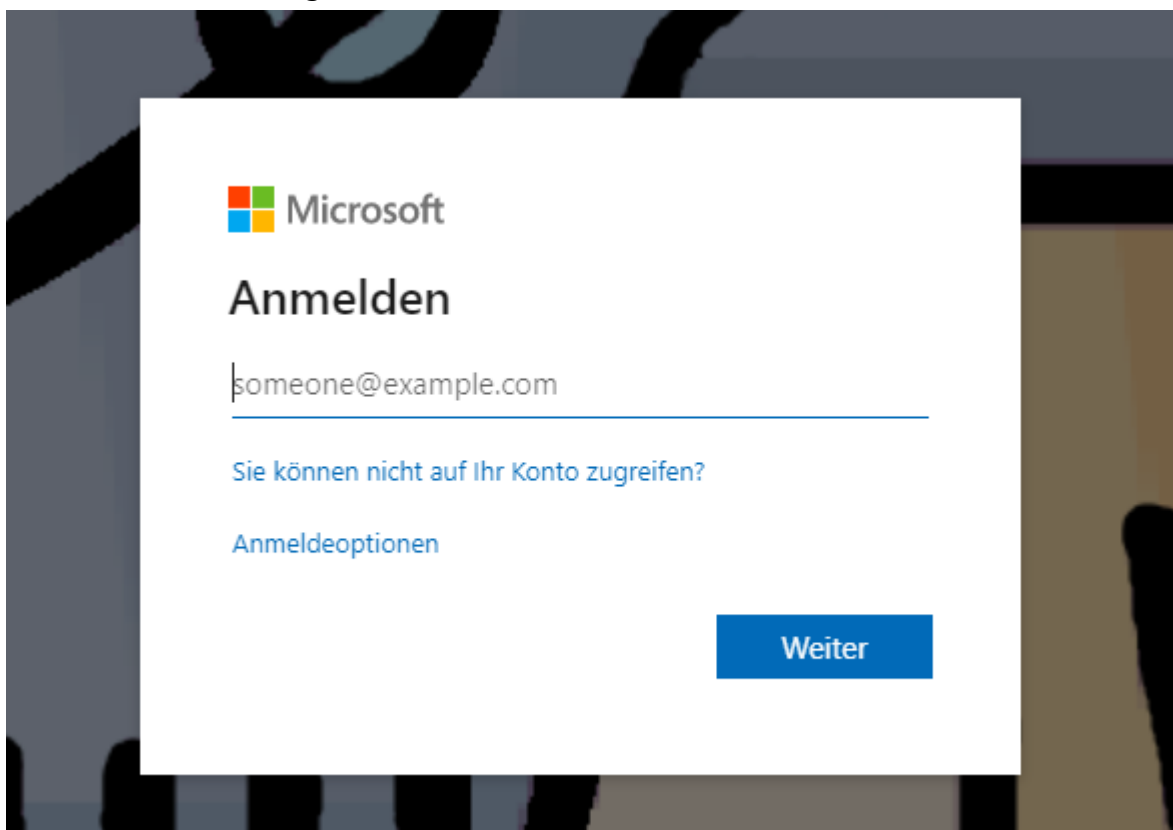


Abbildung 2: Login

### 5.2 Benutzerverwaltung

Die Übersicht der Benutzerverwaltung ist die erste Seite, die sich öffnet. Hier werden alle aktuellen Benutzer in einer Tabelle aufgelistet.



MRCS Managementportal Lara.Doerrerr@plan-b-gmbh.com

Create

C

Search:

Name	Id	Mail	Actions
Felix Rohmeier (Test)	50f075cd-dab4-4bd8-afd8-b6fb1f246035		<div style="border: 1px solid black; padding: 2px; display: inline-block;">Edit</div>
Felix.Rohmeier@plan-b-gmbh.com Rohmeier	276afe58-f399-4c40-8da9-4b8b604bcc4b		<div style="border: 1px solid black; padding: 2px; display: inline-block;">Edit</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Delete</div>
Lara Dörrer	4a9374fb-7f99-4582-b7ce-52fd5a7f6c75	Lara.Doerrerr@plan-b-gmbh.com	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Edit</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Delete</div>
Liana Unseld	efc00451-8c79-44be-91a3-e74e3f903cd6	Liana.Unseld@plan-b-gmbh.com	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Edit</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Delete</div>
Test Client1	1acf13f9-afbb-48dd-9d78-3b7b99ad4a7e	testclient6@web.de	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Delete</div>

Showing 1 to 5 of 5 entries

Abbildung 3: Benutzerübersicht

Um bestehende Benutzer zu löschen gibt es den Delete-Button (B), welcher den gewünschten Benutzer entfernt. Möchten Sie Änderungen an einem Benutzer vornehmen, werden Sie durch den Edit-Button (A) auf die Bearbeitungsseite weitergeleitet.

MRCS Managementportal Lara.Doerrerr@plan-b-gmbh.com

Name

Test Client1

Mail

testclient6@web.de

Enable Account

☒

User Type

Guest

Address

Teststraße.5

Postalcode

00000

Country

Deutschland

City

Aalen

Phone

Save

Cancel

D

Abbildung 4: Benutzerbearbeitung

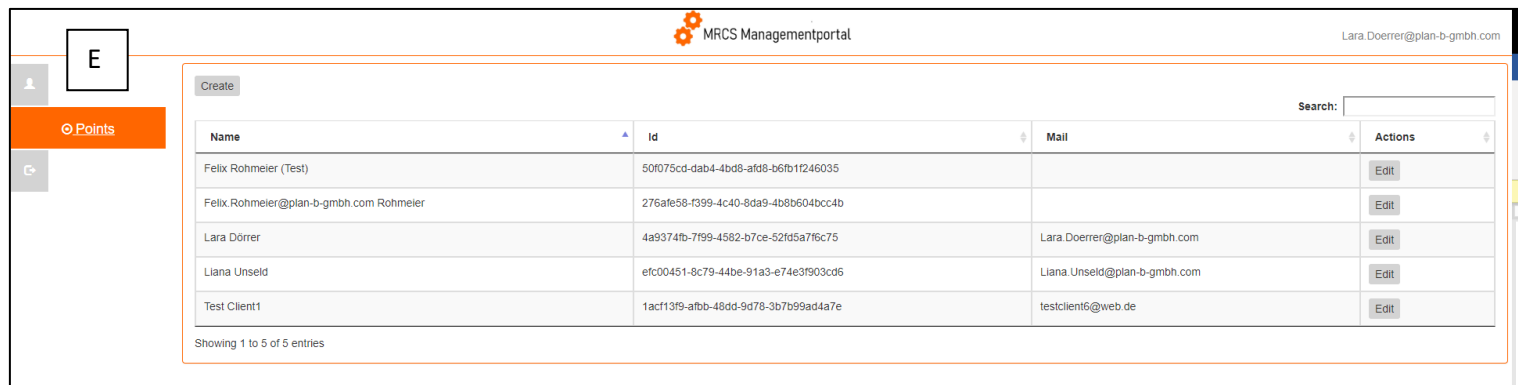
Auf der Bearbeitungsseite können Sie nun den ausgewählten Benutzer bearbeiten. Die grauhinterlegten Felder können nicht bearbeitet werden.

Möchten Sie Ihre Änderungen speichern, klicken Sie auf den Save-Button (D). Möchten Sie Ihre Änderungen verwerfen klicken sie den Cancel-Button (D). In beiden Fällen werden Sie auf die Benutzerübersichtsseite weitergeleitet.

Um einen neuen Benutzer anzulegen gibt es auf der Benutzerübersichtsseite (Abbildung 3) eine Create-Option (C). Hier werden Sie auf eine leere Benutzerbearbeitungsseite (Abbildung 4) weitergeleitet und können so einen neuen Benutzer anlegen.

### 5.3 Navigation

Durch das Navigationsmenü (E) an der linken Seite können Sie von der Benutzerverwaltung zur Informationspunktverwaltung wechseln. Der letzte Navigationspunkt meldet Sie vom Managementportal ab.



MRCS Managementportal

Lara Doerr@plan-b-gmbh.com

Create

Search:

Name	Id	Mail	Actions
Felix Rohmeier (Test)	50f075cd-dab4-4bd8-afd8-b6fb1f246035		Edit
Felix.Rohmeier@plan-b-gmbh.com Rohmeier	276afe58-f399-4c40-8da9-4b8b604bcc4b		Edit
Lara Dörrer	4a9374fb-7f99-4582-b7ce-52fd5a7f6c75	Lara.Doerr@plan-b-gmbh.com	Edit
Liana Unseld	efc00451-8c79-44be-91a3-e74e3f903cd6	Liana.Unseld@plan-b-gmbh.com	Edit
Test Client1	1acf13f9-afbb-48dd-9d78-3b7b99ad4a7e	testclient6@web.de	Edit


Showing 1 to 5 of 5 entries

Abbildung 5: Navigationsmenü

### 5.4 Informationspunktverwaltung

Auf der Informationspunktverwaltungsseite befindet sich, wie auf der Benutzerverwaltungsseite, eine Tabelle mit den aktuellen Informationspunkten im Mittelpunkt. Diese lassen sich mit dem jeweiligen Delete-Button (F) entfernen und durch den Edit-Button (G) werden Sie auf die Bearbeitungsseite der Informationspunkte weitergeleitet.



 MRCS Managementportal Lara.Doerrerr@plan-b-gmbh.com

Create H

Show 10


Search:

Key	Name	Description	Actions
68c26f04-6bf3-48b1-80e9-3778837145db	Der name der eingegeben wird	die description die eingegeben wird	<span>Edit</span> <span style="border: 1px solid black; padding: 2px 5px;">G</span>
6a345316-053a-4514-90db-b321caff2dd6	Der name der eingegeben wird	die description die eingegeben wird	<span>Edit</span> <span>Delete</span>
6aec42c8-8ee4-4356-88fa-9c8b4307cfc9	Der name der eingegeben wird	die description die eingegeben wird	<span>Edit</span> <span>Delete</span>
6e607186-e9a5-421f-828c-5849208e4ca5	Der name der eingegeben wird	die description die eingegeben wird	<span>Edit</span> <span>Delete</span>
701ec998-f165-424a-a08a-c4e1010a88ed	Der name der eingegeben wird	die description die eingegeben wird	<span>Edit</span> <span>Delete</span>
7276d935-b1a8-4c1c-a8df-c0e96af56e8b	Der name der eingegeben wird	die description die eingegeben wird	<span>Edit</span> <span>Delete</span>
74e1daf8-db9c-436b-94d7-b0f45e9473c7	Hallo 2	Please go	<span>Edit</span> <span>Delete</span>
7923ebb0-5083-43ce-ae60-a66f17cc0e02	Bitte Wischen Toiletten und Waschbecken sauber machen	Toilette	<span>Edit</span> <span>Delete</span>
7c3ec13c-57dc-4ffe-82c5-76125817b597	Der name der eingegeben wird	die description die eingegeben wird	<span>Edit</span> <span>Delete</span>
7f943cd4-d44a-4c03-84ca-942236e4b9f2	Der name der eingegeben wird	die description die eingegeben wird	<span>Edit</span> <span style="border: 1px solid black; padding: 2px 5px;">F</span> <span>Delete</span>

Showing 51 to 60 of 97 entries Previous 1 ... 5 6 7 ... 10 Next

Abbildung 6: Informationspunktübersicht

Auf dieser Bearbeitungsseite können nun die einzelnen Informationspunkte bearbeitet werden. Hier können die weiß-hinterlegten Felder bearbeitet werden und die Änderungen können durch den Save-Button (I) gespeichert oder durch den Cancel-Button (J) verworfen werden.

 MRCS Managementportal Lara.Doerrerr@plan-b-gmbh.com

Rowkey

Key

Name

Description

I Save Cancel

J

Abbildung 7: Informationspunktbearbeitung

Um einen neuen Informationspunkt zu erstellen gibt es auf der Informationspunktübersichtsseite (Abbildung 6) einen Create-Button (H). Dieser öffnet eine leere Bearbeitungsseite (Abbildung 7) und der Informationspunkt kann erstellt werden.

Sie haben nun alle Funktionen des Managementportals kennengelernt. Bei Fragen und Anmerkungen können Sie sich gerne bei mir melden.



## 6. Glossar

API	= Application Programming Interface
OpenID Connect	= prüft Identität des Nutzers mit Hilfe eines Autorisierungsservers
CRUD	= Create, Read, Update and Delete
HTML	= Hyper Text Markup Language
Product Owner	= Projekt Rolle im Scrum-Vorgehen, der die
REST-API	= Representational State Transfer – API
CSS	= Cascading Sytle Sheets
JavaScript	= Skriptsprache

## 7. Tabellenverzeichnis

Tabelle 1:	Projektphasen	– Seite 5
Tabelle 2:	Arbeitsmittelplanung	– Seite 5
Tabelle 3:	Personalplanung	– Seite 5
Tabelle 4:	Ressourcenplanung	– Seite 6
Tabelle 5:	Detaillierte Zeitplanung	– Seite 7/8
Tabelle 6:	Soll-/Ist-Vergleich	– Seite 17

## 8. Codeverzeichnis

Codeauszug 1:	Einzelnen Informationspunkt abrufen	– Seite 10
Codeauszug 2:	Alle Informationspunkte abrufen	– Seite 10
Codeauszug 3:	Informationspunkte erstellen	– Seite 11
Codeauszug 4:	Informationspunkt löschen	– Seite 11
Codeauszug 5:	Informationspunkt aktualisieren	– Seite 12
Codeauszug 6:	Graph API Get-Request	– Seite 12
Codeauszug 7:	Graph API Get-Request	– Seite 13
Codeauszug 8:	Graph API Post-Request	– Seite 13
Codeauszug 9:	Graph API Patch-Request	– Seite 14
Codeauszug 10:	Graph API Delete-Request	– Seite 14
Codeauszug 11:	Umsetzung Menü	– Seite 15
Codeauszug 12:	Tabellenimplementation	– Seite 15



Codeauszug 13: Implementierung Textbox – Seite 16

Codeauszug 14: Buttonimplementierung Informationspunkt – Seite 16

Codeauszug 15: Buttonimplementierung Benutzer – Seite 16

## 9. Modellverzeichnis

Modell 1: MVC-Konzept – Seite 9

## 10. Abbildungsverzeichnis

Abbildung 1: PlanB. Aufgabenbereiche – Seite 3

Abbildung 2: Login Seite – Seite 18

Abbildung 3: Benutzerübersicht – Seite 19

Abbildung 4: Benutzerbearbeitung – Seite 19

Abbildung 5: Navigationsmenü – Seite 20

Abbildung 6: Informationsübersicht – Seite 21

Abbildung 7: Informationspunktbearbeitung – Seite 21