**Persönliche News-Feed / Reminder App**

Ein zentralisiertes Benachrichtigungssystem für das Intranet der Berenberg Bank

von

Niklas Grieger

Fachinformatiker Schwerpunkt Anwendungsentwicklung

Abschlussprüfung Winter 2017/2018

Niklas Grieger

Carsten-Reimers-Ring 120

22175 Hamburg

# 

# Inhaltsverzeichnis

[1. Einleitung 1](#_Toc500230965)

[1.1. Vorstellung des Ausbildungsbetriebes 1](#_Toc500230966)

[1.2. Projektbeschreibung 1](#_Toc500230967)

[1.3. Projektbegründung 2](#_Toc500230968)

[1.4. Projektziel 2](#_Toc500230969)

[1.5. Projektschnittstellen 3](#_Toc500230970)

[1.6. Projektabgrenzung 3](#_Toc500230971)

[2. Projektplanung 4](#_Toc500230972)

[2.1. Zeitplanung 4](#_Toc500230973)

[2.2. Ressourcenplanung 4](#_Toc500230974)

[3. Analyse 5](#_Toc500230975)

[3.1. Ist-Analyse 5](#_Toc500230976)

[3.2. Mindestanforderung / Akzeptanzkriterien 5](#_Toc500230977)

[3.3. Kosten/Nutzen-Analyse 6](#_Toc500230978)

[4. Entwurf 8](#_Toc500230979)

[4.1. Datenzugriff 8](#_Toc500230980)

[4.2. Meilensteine 9](#_Toc500230981)

[4.3. Programmkonzept 9](#_Toc500230982)

[4.4. Aktivitätsdiagramm zum Statuswechsel 10](#_Toc500230983)

[4.5. Programmablaufplan 11](#_Toc500230984)

[5. Implementierung 11](#_Toc500230985)

[5.1. Datenbank 11](#_Toc500230986)

[5.2. Backend / Webservice 12](#_Toc500230987)

[5.3. Frontend 12](#_Toc500230988)

[5.4. Unit-Tests 13](#_Toc500230989)

[6. Test 13](#_Toc500230990)

[7. Abnahme 13](#_Toc500230991)

[8. Dokumentation 14](#_Toc500230992)

[8.1. Projektdokumentation 14](#_Toc500230993)

[8.2. Entwicklerdokumentation 14](#_Toc500230994)

[8.3. Benutzerdokumentation 14](#_Toc500230995)

[9. Glossar 15](#_Toc500230996)

[10. Anhang 16](#_Toc500230997)

[10.1. Ressourcen 16](#_Toc500230998)

[10.2. Datenzugriff 17](#_Toc500230999)

[10.3. Aktivitätsdiagramm bei Statusänderung 18](#_Toc500231000)

[10.4. Programmablaufplan 19](#_Toc500231001)

[10.5. Datenbank 20](#_Toc500231002)

[10.6. Backend 21](#_Toc500231003)

[10.7. Frontend 22](#_Toc500231004)

[10.8. Unit-Tests 23](#_Toc500231005)

[10.9. Entwicklerdokumentation 24](#_Toc500231006)

[10.10. Benutzerdokumentation 25](#_Toc500231007)

# Einleitung

In dieser Projektdokumentation werden im Rahmen der IHK-Abschlussarbeit die

Projektphasen und das entstandene Produkt detailliert beschrieben.

## Vorstellung des Ausbildungsbetriebes

Die Berenberg Bank existiert seit 1590 und ist damit die älteste Privatbank Deutschlands. Seit langer Zeit betreibt Berenberg einen internen IT-Dienstleister, der inzwischen mehr als 250 Mitarbeiter beschäftigt.

Seit dem Jahre 2013 bildet die IT von Berenberg aus. Unter anderem den Ausbildungsberuf Fachinformatiker Schwerpunkt Anwendungsentwicklung.

## Projektbeschreibung

Die IT von Berenberg ist unter anderem für die Weiterentwicklung und den Betrieb des Intranets verantwortlich.

Im Rahmen dieser Verantwortung entstand ein Gesamtprojekt.

Ziel des Gesamtprojekts ist die Entwicklung eines neuen Webportals, welches dem Anwender einen komfortablen Umgang mit dem Intranet bietet. Es soll selbsterklärend und leicht zu bedienen sein, in dem es eine Übersicht über alle für den Anwender relevanten Systeme gibt.

Der Aufbau des Gesamtprojektes erfolgt Modular.

Mit einem solchen abgegrenzten Teilbereich beschäftigt sich dieses Projekt der

„Reminder App“.

Das Modul der „Reminder App“ wird im Frontend Bereich mit der Programmiersprache „TypeScript“ und dem Framework „Angular“ entwickelt. Im Backend Bereich wird die Programmiersprache „C#“ und das Framework „.Net Core“ angewandt. Im Bereich der Datenbank wird die Programmiersprache „PL/SQL“ verwendet.

Im Rahmen der Abnahme durch den Product Owner wird dieses Modul ein Teil des Gesamtprojektes und damit Bestandteil des neuen produktiven Intranets.

## Projektbegründung

Im Hause Berenberg werden von den Fachbereichen viele verschiedene Systeme genutzt.

Aktuell muss der Anwender in allen Systemen gleichzeitig aktiv sein, um

Benachrichtigungen des jeweiligen Systems zu erhalten.

Einige Benachrichtigungen werden per Email an den Anwender versandt, während andere erst über alte Java Applikationen abgerufen oder über das alte Intranet umständlich gesucht werden müssen.

Aufgrund der veralteten Technologien können die Ladezeiten, die damit einhergehen, sehr lang sein.

Daraus resultiert ein zeitlicher Verlust in Bezug auf die Kenntnisnahme. Ebenso ist die Gefahr gegeben, dass Benachrichtigungen ihren Empfänger nicht rechtzeitig erreichen, weil diese das jeweilige System noch nicht geöffnet hatten.

Die Reminder, die über Email versandt werden erreichen mehrere Anwender und die Aufgabe, die aus diesem Reminder entsteht muss nur von einem Empfänger bearbeitet werden. Falls die Aufgabe bereits erledigt ist, kann es passieren, dass die anderen Empfänger den neuen Status nicht mitbekommen und Zeit mit einer bereits gelösten Aufgabe verschwenden.

## Projektziel

Das Ziel des Projekts ist es, ein auf den Anwender abgestimmtes Benachrichtigungssystem zu entwickeln, in dem Daten aus verschiedenen Quellen gesammelt angezeigt werden können und eine zentrale Möglichkeit zur Statuspflege der Benachrichtigung geboten wird.

Zum Start werden Benachrichtigungen aus zwei Systeme integriert:

* Oracle Datenbank – Daten, die durch eine Datenbanklogik jeden Tag in eine Tabelle gespeichert werden. Dies sind Benachrichtigungen wie Events, Geburtstage von Kunden, Formulare die bis zu einem bestimmten Stichtag ausgefüllt werden müssen etc.
* ROM (Rechte Organisation Management) – ein eingekauftes System zum Verwalten von Mitarbeiterrechten innerhalb des Intranets. Diese Benachrichtigungen sind Aufgaben, die berechtigten Anwendern zugeteilt werden und zur Freischaltung von Mitarbeiterrechten dienen.

Die Architektur soll offen für die Integration von Benachrichtigungen weiterer Systeme sein.

Der Anwender soll auf der Startseite des neuen Webportals eine App angezeigt bekommen, die ein- und ausgeblendet werden kann.

Verschiedene Informationen, wie anstehende Aufgaben, Events, Geburtstage von Kunden, Freigaben und Aufgaben verschiedener Workflows sollen dem Anwender angeboten werden.

Der Anwender bekommt durch diese App eine zentralisierte, transparente und

einheitliche Anzeige vieler verschiedener Benachrichtigungen.

## Projektschnittstellen

Die „Reminder App“ wird in das Webportal integriert und ist somit von der

Verfügbarkeit des Webportals abhängig.

Der Benutzer wird in dem Webportal über das hausinterne Active Directory

authentifiziert. Ohne diese Authentifizierung ist es dem Benutzer nicht möglich die Seite zu nutzen. Die „Reminder App“ wird ebenfalls diese Authentifizierung benötigen, damit die Daten personenspezifisch abgerufen werden können.

Das ROM System ist ein eingekauftes System, das heißt es muss mit dem Dienstleister abgesprochen werden, wie der Zugriff über die Schnittstelle erfolgen kann.

Ebenfalls die Oracle Datenbank muss verfügbar sein, da diese Daten ein Hauptbestandteil der „Reminder App“ sein werden.

Die Entwicklung der „Reminder App“ findet parallel zu der Weiterentwicklung des Webportals statt. Dies erfordert eine enge Absprache mit den anderen Entwicklern, so dass die „Reminder App“ letztendlich in das Webportal integriert werden kann.

## Projektabgrenzung

Dieses Projekt bezieht sich auf die Entwicklung sowie die Integration der „Reminder App“ in das Webportal des Intranets.

Das gesamte Projekt der „Reminder App“ ist somit als eigenständiges Modul des Gesamtprojekts einzustufen.

# Projektplanung

## Zeitplanung

|  |  |
| --- | --- |
| Phase | Zeit (h) |
| Analyse | **5** |
| * Ist-Analyse | 2 |
| * Mindestanforderungen an das Produkt | 2 |
| * Kosten/Nutzen-Analysen | 1 |
| Entwurf | **5** |
| * Datenzugriff visualisieren | 2 |
| * Aktivitätsdiagramm zum Statuswechsel einer * Benachrichtigung | 1 |
| * Programmablaufplan | 2 |
| Implementierung | **38** |
| * Datenbank Funktionalitäten | 10 |
| * Backend Funktionalitäten | 12 |
| * Frontend / Anzeige und Konfiguration der App | 13 |
| * Unit-Tests schreiben | 3 |
| Test | **4** |
| Refactoring | **4** |
| Abnahme der Anwendung | **2** |
| Dokumentation | **12** |
| * Projektdokumentation | 8 |
| * Entwicklerdokumentation | 2 |
| * Benutzerdokumentation | 2 |
| Gesamtzeit | **70** |

Die Zeit für die Entwicklung der Unit Tests hat sich um **3 Stunden** verkürzt, da sich die Unit Tests in grober Form generieren lassen. Dieses Feature war zurzeit der

Planung im Prüfungsantrag noch nicht bekannt.

Der Frontend- und Backend Entwicklungszeit mussten diese **3 Stunden** hinzugerechnet werden, da viele Funktionalitäten in den verschiedenen Frameworks erst noch erlernt werden mussten.

## Ressourcenplanung

Eine Auflistung der verwendeten Ressourcen für die Umsetzung des Projekts befindet sich im Anhang „[Ressourcen](#_Ressourcen)“. Zu den Ressourcen zählen Hard- und Software sowie das benötigte Personal.

Bei der Auswahl der Software und der Programmteile, die verwendet werden, wird darauf geachtet, dass es sich um Open-Source Produkte handelt oder die Lizenzen bereits vorhanden sind.

# Analyse

## Ist-Analyse

Wie bereits in der [„Projektbegründung“](#_Projektbegründung) beschrieben, muss der Anwender aktuell in vielen verschiedenen Systemen aktiv sein, um über Neuigkeiten benachrichtigt zu werden.

Zur Vereinfachung und einer besseren Übersichtlichkeit wird aktuell ein neues Intranet aufgebaut und im Zuge dessen ein neues Webportal entwickelt, welches als Einstiegsseite für alle möglichen internen Applikationen fungieren soll.

## Mindestanforderung / Akzeptanzkriterien

Die Akzeptanzkriterien sind die von dem Auftraggeber festgelegten Funktionalitäten, die das Produkt mindestens leisten können muss, bis eine Abnahme erfolgt.

Für die „Reminder App“ sind die Akzeptanzkriterien folgende:

* Eine übersichtliche und kompakte Anzeige der Benachrichtigungen
* Möglichkeiten der Statuspflege durch den Anwender
* Anzeige der Benachrichtigungen aus der Oracle Datenbank, innerhalb eines

festgelegten Zeitraumes

* Anzeige der Benachrichtigungen aus dem ROM-System
* Möglichkeit den Status einer ROM Benachrichtigung / Aufgabe zu ändern
* Das Design muss zu dem Design des Webportals passen
* Die App ist vollständig in das Webportal zu implementieren
* Die App soll ein – und ausgeblendet werden können

Über die Mindestanforderungen hinaus:

* Der Anwender hat die Möglichkeit den Zeitraum, in dem die Benachrichtigungen der „Reminder App“ angezeigt werden, individuell einzustellen und zu speichern
* Die Anzeige der nächsten 5 Termine aus Outlook (Exchange).
* Eine Übersicht über alle Benachrichtigungen, die dem Anwender zugeteilt sind, ohne Einschränkungen auf einen Zeitraum
* Trotz ausgeblendeter „Reminder App“ werden die Anzahl der aktuell anstehenden Benachrichtigungen angezeigt und über ein Popup sofort einsehbar
* Status der ROM Benachrichtigungen wird mit beteiligten Anwendern in Echtzeit geteilt

## Kosten/Nutzen-Analyse

Da die Funktionalitäten und die Datenquellen der „Reminder App“ im fachlichen sowie technischen Sinne sehr individuell sind, kam weder ein Open Source noch ein Lizenzpflichtiges Produkt in Frage.

Diese Individualität ermöglicht einem Entwickler die „Reminder App“ jederzeit um Systeme oder allgemeine Funktionalitäten zu erweitern.

Im Zuge der Realisierung dieses Projekts ergeben sich folgende Kosten.

Ein Auszubildender wird mit 15€/Stunde und alle anderen Mitarbeiter mit 76€/Stunde verrechnet.

Diese Kosten setzen sich aus dem jeweiligen Lohn, den Betriebskosten für den Arbeitsplatz und den Softwarelizenzen zusammen.

|  |  |  |  |
| --- | --- | --- | --- |
| Vorgang | Mitarbeiter | Zeit (h) | Kosten (€) |
| Entwicklungskosten | 1x Auszubildender | 70 | 1050 |
| Fachlicher Test | 1x Product Owner | 4 | 304 |
| Abnahme | 1x Product Owner | 2 | 152 |
| Gesamt |  |  | **1506** |

Die Darstellung einer Benachrichtigung in den ursprünglichen Systemen geschieht teilweise mit langen Ladezeiten, da diese Systeme alte Technologie anwenden.

Daraus resultiert eine Kundenunzufriedenheit und bei der großen Menge an Benachrichtigungen, die täglich übermittelt werden, kommt es dadurch zu einem erheblichen Zeitverlust für den Anwender.

Bei den Ladezeiten der verschiedenen Systeme spielen viele Faktoren eine Rolle, wie zum Beispiel die Performance des Clients. Zur Anzeige der durchschnittlichen Amortisationsdauer wurde folgende Tabelle erstellt.

|  |  |  |  |
| --- | --- | --- | --- |
| System | Ladezeit (alt)  Pro Aufruf | Ladezeit der  „Reminder App“  Pro Aufruf | Einsparung  Pro Aufruf |
| Alte Intranet Seite | 0,55 min | 0,25 min |  |
| ROM | 0,55 min |  |
| Exchange | 0,15 min |  |
| Gesamt | 1,25 min | 0,25 min | 1 min |

Berechnung der Amortisationsdauer (Ein Aufruf des jeweiligen Systems pro Tag):

* Ein Aufruf des jeweiligen Systems pro Tag
* 1500 Mitarbeiter der Firma Berenberg

Kosten pro Tag mit der alten Lösung:

Kosten pro Tag mit der „Reminder App“:

Resultat:

Sobald ca. 1180/1500 Mitarbeiter, die „Reminder App“ nutzen, entsteht bereits ab dem 1. Tag eine gewaltige Kostenersparnis.

Wenn 1500 Mitarbeiter die App nutzen, werden 390,2 € Kosten / Tag gespart.

Grafische Darstellung der Kosten pro Tag:

Zusätzlich zu der positiven Kosten/Nutzen-Prognose kann noch ein weiterer Nutzenvorteil der „Reminder App“ festgestellt werden, der darin besteht, dass die Gefahr, dass der Anwender die Benachrichtigung nicht rechtzeitig erhält, weil er das jeweilige System nicht geöffnet hat, komplett vermieden wird.

In dem Fall, dass die Benachrichtigung mehreren Anwendern zugeordnet ist und sie bereits als Erledigt gekennzeichnet wurde, wird jedem anderen dies sofort mitgeteilt, so dass kein überflüssiger Arbeitsaufwand entsteht.

Fehler, die aus der unterbliebenen Kenntnisnahme erwachsen, sind kaum in einen monetären Wert zu beziffern, da der Schaden unter Umständen extrem hoch ausfallen kann.

Durch die „Reminder App“ wird dieses Schadensrisiko, bei fehlerfrei funktionierendem Webportal, bis zu 100% minimiert und zwar einfach dadurch, dass dem Anwender systemunabhängig alle Benachrichtigungen sofort zentral gebündelt über das Webportal ersichtlich werden.

Über die moderne, transparente und einheitliche Darstellung der Benachrichtigungen, verbunden mit den schnellen Ladezeiten, die sich durch die verwendeten Technologien ergeben, werden die Kundenzufriedenheit und ein effizienteres Arbeiten gefördert.

# Entwurf

## Datenzugriff

In der Oracle Datenbank wurde eine, mit der Programmiersprache PL/SQL

geschriebene, Funktion entwickelt. Diese Funktion fragt anhand von verschiedenen Parametern Daten aus einer Tabelle ab, trägt diese in einem definierten Schema zusammen und gibt das Ergebnis als Tabelle zurück.

Für den Datenzugriff im Web wird das Framework ASP.Net Core in der Version 2.0 genutzt. Dieses Framework wird zum Entwickeln eines Webservices genutzt.

Dieser Webservice hat mehrere Vorteile:

Der Webservice kann…

* unabhängig von dem Frontend weiterentwickelt, geändert und veröffentlicht werden
* unabhängig von der Frontend Programmiersprache gleichermaßen am Client abgerufen werden

Im Webservice wurde eine Methode entwickelt, die die vorherig genannte PL/SQL Funktion aufruft und das Ergebnis als eine Liste speichert.

Daten anderer Systeme werden im Webservice ebenfalls über Methoden abgefragt.

Zu anderen Systemen gelten unter anderem:

* ROM
  + Die Daten werden über eine Schnittstelle bereitgestellt.
* Exchange
  + Die Daten werden über eine API Schnittstelle von Microsoft bereitgestellt. Diese Schnittstelle nennt sich „Exchange Web Services (EWS)“.

Sobald die Daten der jeweiligen Systeme geladen wurden, werden diese in ein einheitliches Schema zusammengeführt und als eine Liste zurückgegeben.

Darauf folgend wird im Frontend eine Methode auf dem Webservice aufgerufen, die ein JSON Array mit den zusammengeführten Ergebnissen zurückgibt.

Dieser ganze Vorgang wird im Anhang unter dem Punkt „[Datenzugriff](#_Datenzugriff)“ visualisiert.

## Meilensteine

Um einen strukturierten Ablauf der Entwicklung planen zu können, wurden folgende Meilensteine festgelegt:

1. Analyse der verfügbaren Benachrichtigungen in der Datenbank
2. Datenbankfunktion entwickeln
   * Sammeln der Daten mit einem SQL-Statement
   * Table Type (Schema der Ergebnis Liste) entwickeln
   * PL/SQL Funktion entwickeln
3. Webservice erweitern
   * Struktur: Controller -> Interface -> Repository
   * Datenabfragen entwickeln
   * Testen der Rückgabewerte
4. Frontend / Oberfläche
   * Analyse der Anzeige-/Designmöglichkeiten
   * Funktionen zum Abrufen des Webservices entwickeln
   * Funktionen zur Statusänderung der Benachrichtigungen entwickeln
   * Testen der Anzeige und Funktionalitäten
   * Implementierung in die vorhandene Funktionsweise der Apps
5. Test
   * Finaler Frontend und Backend Test des Entwicklers
   * Finaler Oberflächen Test des Product Owners

## Programmkonzept

Allgemein:

Ein Anwender öffnet das Webportal des Intranets und wird per Windows Authentifizierung automatisch authentifiziert. Wenn die Authentifizierung erfolgreich war, wird das Webportal angezeigt und die personalisierten Daten werden geladen.

„Reminder App“:

Der Anwender hat die Möglichkeit die „Reminder App“ ein- oder auszublenden.

Wenn der Anwender die „Reminder App“ eingeblendet hat, erfolgt im Hintergrund ein Aufruf des Webservices.

Im Webservice wird eine Methode mit folgenden Parametern aufgerufen:

* NT-User: Name des authentifizierten Anwenders
* Start-Datum: Ein Datum, ab dem die Daten angezeigt werden sollen
* Ende-Datum: Ein Datum, bis zu dem die Daten angezeigt werden sollen

Der Client ruft diese Methode über eine HTTP-Anfrage auf. Das Resultat dieser

Anfrage ist ein JSON-Array mit allen für die „Reminder App“ erforderlichen

Informationen. Bevor die Daten jedoch angezeigt werden können, werden diese

anhand der Art des Systems gruppiert um die Übersichtlichkeit zu erhöhen.

Der Anwender hat zusätzlich die Möglichkeit den Status der angezeigten Benachrichtigungen zu ändern. Der Anwender hat in den verschiedenen Systemen folgende Möglichkeiten:

* Daten aus der Oracle Datenbank:
  + Der Anwender hat die Möglichkeit die Benachrichtigung auf „Erledigt“ zu setzen
    - „Erledigt“ bedeutet, dass der Status der Benachrichtigung in der Datenbank geupdatet und somit aus der Anzeige, der zugeteilten Anwender entfernt wird.
* ROM:
  + Bei den ROM Benachrichtigungen handelt es sich um Aufgaben, die mehreren Anwendern zugeteilt werden können.
  + Eine ROM Aufgabe bedeutet, der benachrichtigte Anwender muss für den Anwender xy eine Berechtigung genehmigen oder abweisen.
  + Der Anwender hat die Möglichkeit die Benachrichtigung auf „in Arbeit“ zu setzen, sodass andere Anwender wissen, dass diese Aufgabe bereits bearbeitet wird.
* Exchange
  + Bei den Exchange Benachrichtigungen handelt es sich um die nächsten 5 Termine, die der Anwender in Outlook eingestellt hat.
  + Bei diesen Benachrichtigungen ist noch keine klare Definition zur Statusänderung vorhanden.

## Aktivitätsdiagramm zum Statuswechsel

Anders als bei den Systemen Oracle Datenbank und ROM, ist bei dem System Exchange noch nicht klar definiert, ob nur eine Anzeige oder auch eine Statuspflege der Benachrichtigungen möglich sein soll. Deshalb ist es in dem folgenden Diagramm nicht enthalten.

Eine Visualisierung der Hintergrundprozesse bei einer Statusänderung ist im Anhang unter dem Punkt „[Aktivitätsdiagramm bei Statusänderung](#_Aktivitätsdiagram_bei_Statusänderun)“ zu finden.

## Programmablaufplan

Eine Visualisierung der Prozesse, die ausgeführt werden, sobald ein Anwender das Webportal öffnet, ist im Anhang unter dem Punkt „[Programmablaufplan](#_Programmablaufplan)“ zu finden.

# Implementierung

## Datenbank

Der erste Schritt der Entwicklung war das Entwickeln einer PL/SQL Datenbankfunktion, die alle relevanten Daten aus der Datenbank zusammenführt und als einen Oracle Table Type zurückgibt.

Außerdem können der Funktion Parameter übergeben werden, mit denen das

Rückgabeergebnis gefiltert werden kann.

Folgende Parameter sind aktuell vorhanden:

* Mitarbeiter: Pflicht (Einschränkung auf einen Mitarbeiter, damit die Benachrichtigungen personenspezifisch abgerufen werden können)
* Status: Optional (Möglichkeit die Benachrichtigungen nach einem bestimmten Status zu filtern)
* ReminderTypes: Optional (Möglichkeit nach bestimmten Benachrichtigungsarten zu filtern)

Diese Funktion ist jederzeit erweiterbar.

Ein Ausschnitt des Quellcodes dieser Funktion sowie ein Beispielaufruf der Funktion ist im Anhang „[Datenbank](#_Datenbank)“ zu finden.

## Backend / Webservice

Als Nächstes wurde der Webservice erweitert.

Es kamen folgende Methoden hinzu:

|  |  |
| --- | --- |
| Name der Methode | Beschreibung / Rückgabewert |
| GetReminder | Gibt alle Daten zurück, ohne Einschränkungen |
| GetLatestReminder | Gibt alle Daten zurück, eingeschränkt auf eine Art sowie einen Zeitraum |
| GetReminderToday | Gibt alle Daten zurück, eingeschränkt auf das aktuelle Datum |
| GetReminderTypes | Gibt alle Benachrichtigungsarten zurück, eingeschränkt auf einen Zeitraum |
| GetExchangeAppointments | Gibt die letzten 5 Termine der Benutzers aus Outlook zurück |
| SetState | Updatet den Status des Datensatzes. Gibt bei Erfolg die Anzahl der betroffenen Datensätze zurück. |
| SaveConfig | Speichert die Konfiguration des Anwenders in die Datenbank.  Gibt einen Boolean (true / false) Wert zurück. |
| GetConfig | Gibt eine Liste der Konfigurationen des Anwenders aus der Datenbank zurück.  Eine Konfiguration ist z.B. ein Zeitraum in dem die Benachrichtigungen angezeigt werden. |

Der Webservice wird mit dem Framework ASP.Net Core in der Version 2.0 betrieben und ist in der Programmiersprache C# geschrieben.

Im Anhang „[Backend](#_Backend)“ ist ein Ausschnitt des Quellcodes der Methode

„GetLatestReminder“ zu finden - eine Methode, welche die Daten, eingeschränkt auf einen Zeitraum und eine Benachrichtigungsart, zurückgibt.

## Frontend

Im Frontend Bereich wurde das UI Framework Angular in der Version 4.4.5 und die Programmiersprache TypeScript genutzt.

Für die Anzeige der Daten wurde das Modul „PanelBar“ aus dem UI Framework KendoUI genutzt. Dieses Modul wurde verwendet, da es die Daten gruppiert anzeigen kann und es viele Möglichkeiten zur individuellen Konfiguration gibt.

Dieses Modul benötigt ein verschachteltes JSON Array zum Gruppieren der Daten.

Zum verdeutlichen, wie dieser JSON Array in der „Reminder App“ aufgebaut wird, gibt es im Anhang „[Frontend](#_Frontend)“ einen Ausschnitt des Quellcodes.

Screenshots und Beschreibungen zur Darstellung der „Reminder App“ befinden sich im Anhang „[Benutzerdokumentation](#_Benutzerdokumentation)“.

## Unit-Tests

Parallel zur Frontend Entwicklung wurden Unit Tests zum automatisierten Testen der Oberfläche entworfen.

Zum Entwickeln der Unit Tests wurde das Test Framework „Karma“ genutzt.

Falls „Karma“ initialisiert ist, kann es unter anderem die Dateien „beobachten“ und bei Änderung automatisch einen Test durchführen.

Der Quellcode für einen Unit Test ist im Anhang „[Unit Tests](#_Unit-Tests)“ zu finden.

# Test

Die manuellen Tests wurden vom Entwickler sowohl während der Entwicklungszeit als auch danach durchgeführt.

Der fachliche Test wurde vom Product Owner durchgeführt. Dieser fachliche Test fand in enger Absprache mit dem Entwickler statt, so dass mögliche Fehler schnell beseitigt und ein erneuter Test erfolgen konnte.

Erst nach erfolgreicher Testphase kam es zur Abnahme des Produkts.

# Abnahme

Die Abnahme des Produkts erfolgte durch den Product Owner.

Die [Akzeptanzkriterien](#_Mindestanforderung_/_Akzeptanzkrite) wurden vollständig erfüllt.

Zudem wurden folgende weitere Features mit abgenommen:

* Anzeige der nächsten 5 Termine aus Outlook.
* Eine Übersicht über alle Benachrichtigungen, die dem Anwender zugeteilt sind, ohne Einschränkungen auf einen Zeitraum
* Trotz ausgeblendeter „Reminder App“ werden die Anzahl der aktuell anstehenden Benachrichtigungen angezeigt und über ein Popup sofort einsehbar
* Der Anwender hat die Möglichkeit den Zeitraum, in dem die Benachrichtigungen der „Reminder App“ angezeigt werden, individuell einzustellen und zu speichern

Aufgrund der kurzen Entwicklungszeit wird das optionale Feature der Echtzeitübertragung einer ROM Statusänderung zu einem späteren Zeitpunkt erfolgen.

# Dokumentation

## Projektdokumentation

Dieses Dokument stellt die Projektdokumentation dar.

## Entwicklerdokumentation

Die Entwicklerdokumentation soll einem Entwickler einen erleichterten Einstieg in die Programmierung der „Reminder App“ geben.

Neben den Kommentierungen im Code gibt es auch eine komplette Entwicklerdokumentation im XML Format (Generiert durch das Tool GhostDoc für Visual Studio 2017). Einen Auszug der Entwicklerdokumentation befindet sich im Anhang „[Entwicklerdokumentation](#_Entwicklerdokumentation)“.

## Benutzerdokumentation

Die Benutzerdokumentation soll einem Anwender einen erleichterten Einstieg in die Funktionsweise der „Reminder App“ geben.

Die vollständige Benutzerdokumentation befindet sich im Anhang „[Benutzerdokumentation](#_Benutzerdokumentation)“.

# Glossar

|  |  |
| --- | --- |
| Fachbegriff | Erläuterung |
| Reminder App | Der Name des Produkts. Eine persönliche News-Feed App für den Mitarbeiter. |
| Scrum | Ein Vorgehensmodell im agilen Projektmanagement |
| Igel Thin Client | Ein Client der Firma Igel, der sich über das Netzwerk mit einem virtuellen Rechner auf einem Server verbindet. |
| Telerik (Kendo UI) | Eine Bibliothek der Firma Telerik mit wiederverwendbaren Klassen und Oberflächen Komponenten |
| ASP.Net Core | Ein Open-Source Framework von der  Firma Microsoft zur Programmierung von Plattform übergreifenden Applikationen |
| Angular | Ein Oberflächen Framework zur Entwicklung einer SPA Webseite |
| SPA (Single Page Application) | Eine Applikation, die beim Navigieren nicht die komplette Seite neu lädt, sondern nur die neu empfangenen Daten. Dies verringert die Ladezeiten einer Webseite erheblich. |
| ROM | Rechte Organisation Management. Eine eingekaufte Software, die zur Verwaltung der Berechtigungen zwischen Anwender und System genutzt wird. |
| Exchange | Eine E-Mail-Transport-Server-Software von Microsoft. Zentrale Ablage und Verwaltung von E-Mails, Terminen, Kontakten etc. Quelle: Wikipedia |
| Table Type | Ein in PL/SQL geschriebenes Modell,  welches eine Tabellenstruktur repräsentiert. |
| Product Owner | Eine zentrale Rolle im Agilen Scrum Team. Eine Person, die mit dem Fachbereich in Kontakt steht und die Abnahme der im Team entwickelten Produkte durchführt. |
| JSON Array | Ein Datentyp |
| TFS | Der Team Foundation Server bietet dem Entwickler sehr viele Features zum verwalten seiner Workflows, u.a. enthält der TFS eine Versionsverwaltung und die Möglichkeit Unit Tests zu automatisieren. |

# Anhang

## Ressourcen

Hardware:

Zur Entwicklung wurde folgende Hardware genutzt:

Client:

* + Igel Thin Client zur Verbindung mit einem virtuellen Rechner

Betriebssystem:

* + Windows Server 2012 (64 Bit)

Software:

Folgende Software wurde für das Projekt genutzt (alle Lizenzen vorhanden):

* + Visual Studio 2017 (Community Edition)
  + Visual Studio Code (Community Edition)
  + Microsoft Visio 2016 (Grafiken für die Dokumentation)
  + Ghost Doc for Visual Studio 2017 (Community Edition)

Frameworks / Bibliotheken:

Folgende Frameworks / Bibliotheken wurden für die Entwicklung genutzt

(alle Lizenzen vorhanden):

* + Telerik (Kendo UI)
  + ASP.Net Core 2.0
  + Angular

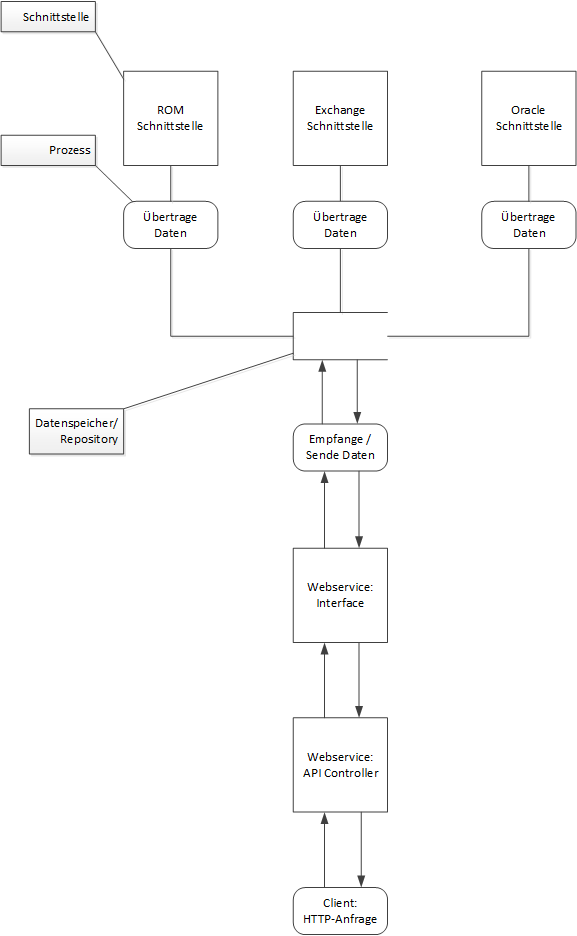
Personal

Folgende Personen waren in das Projekt (teilweise) involviert:

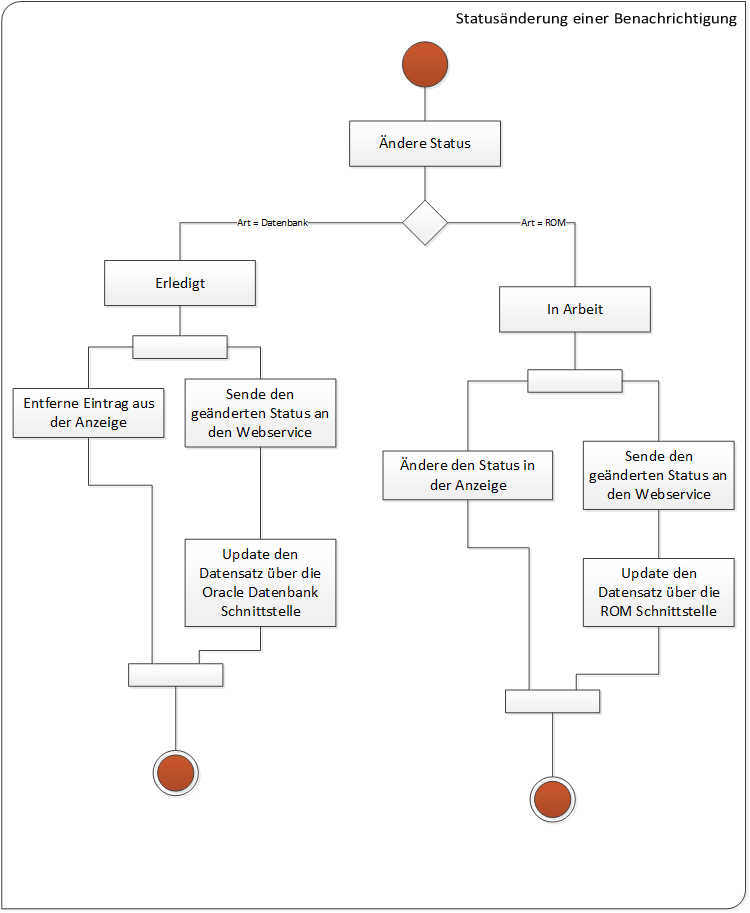
* + 1x Auszubildender (Projektbeauftragter)
  + 1x Auftraggeber (Abnahme und fachlicher Test)

## Datenzugriff

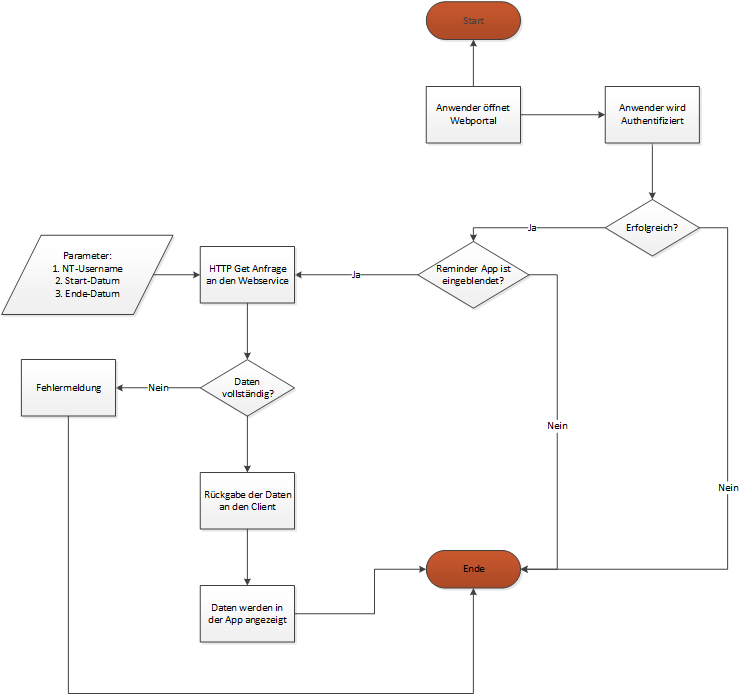
Datenflussmodelldiagramm: Erstellt mit Microsoft Visio 2016



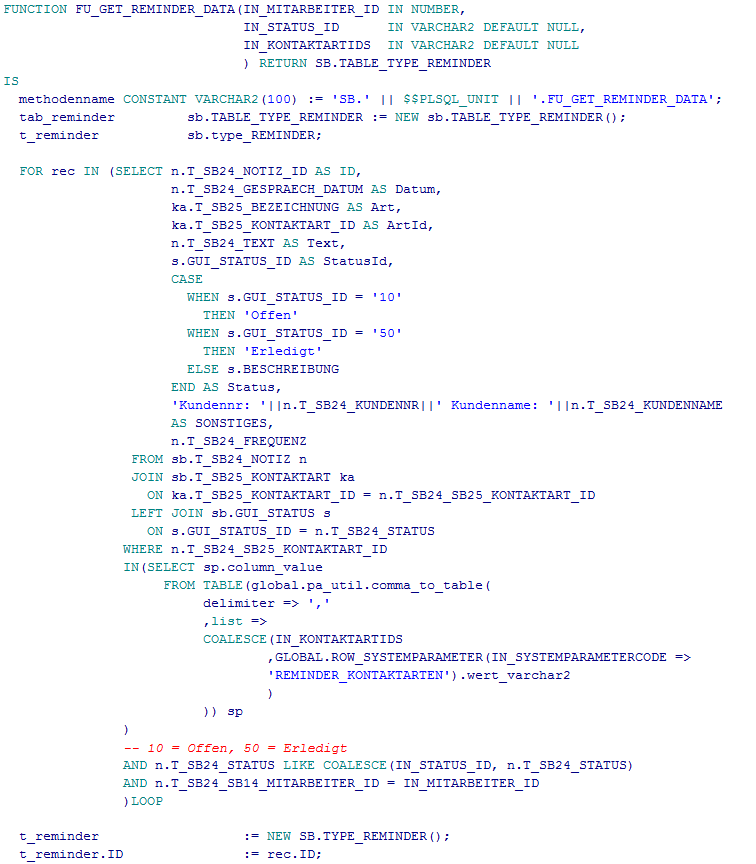
## Aktivitätsdiagramm bei Statusänderung



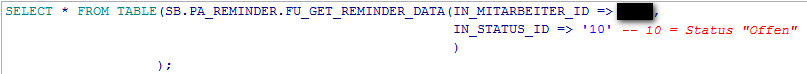
## Programmablaufplan



## Datenbank

Ausschnitt der Oracle Funktion:

…

Aufruf der Funktion:

## Backend

Ausschnitt aus dem Reminder Controller im Web Service:

[HttpGet("GetLatestReminder")]

public async Task<IActionResult> GetLatestReminder(

int artId, string startDateString, string endDateString, int mitarbeiterId)

{

try

{

if (artId == (int)ReminderEnums.Rom)

{

var username = \_userService.GetUserName();

var rom = \_romQueries.CurrentTasks(username);

var romData = await rom;

var migratedRomData = new List<ReminderModel>();

foreach (var romItem in romData)

{

var model = new ReminderModel();

model.Id = romItem.TaskId;

model.Text = romItem.NAME;

model.Datum = romItem.STARTTIMESTAMP;

model.Sonstiges = "Status: " + romItem.State;

model.StatusId = romItem.State.ToString();

model.Status = "Offen";

model.Art = "Rom";

model.ArtId = (int)ReminderEnums.Rom;

migratedRomData.Add(model);

}

return Ok(migratedRomData.OrderBy(r=>r.Datum));

}

if (artId == (int)ReminderEnums.Exchange)

{

var exchange = \_reminderQueries.GetExchangeAppointments();

return Ok(exchange);

}

var oracleResult = \_reminderQueries.GetLatestReminder(

artId, startDateString, endDateString, mitarbeiterId);

var oracleData = await oracleResult;

return Ok(oracleData);

}

catch (Exception ex)

{

return BadRequest(ex);

}

}

## Frontend

Formatieren der JSON Objekte vom Webservice in ein, für das KendoUI Modul, lesbares JSON Array.

Programmiersprache: TypeScript

private loadLatestItems() {

//Holt die Configs des Users aus der Datenbank

this.userService.getCurrentUser().subscribe(user => {

this.reminderService.getConfigs(user.userId).subscribe(configs => {

if (configs.length > 0) {

//Zeitraum der Reminder: Tage zurück

this.daysBackward = Number.parseInt(configs[configs.findIndex(c => c.title == 'timeline')].value1);

//Zeitraum der Reminder: Tag Vorwärts

this.daysForward = Number.parseInt(configs[configs.findIndex(c =>

c.title == 'timeline')].value2);

}

//Subtrahiert vom Heutigen Tag die aus den Configs ausgelesenen Tage zurück

this.startDateString = addDays(new Date(), this.daysBackward)

.toLocaleString('de-DE', this.dateStringOptions);

//Addiert dem Heutigen Tag die aus den Configs ausgelesenen Tage Vorwärts

this.endDateString = addDays(new Date(), this.daysForward)

.toLocaleString('de- DE', this.dateStringOptions);

//Fragt die die Reminder Arten ab, die der User innerhalb des Zeitraums besitzt

this.reminderService.getReminderArten(

this.startDateString, this.endDateString)

.subscribe((data) => {

// Die Reminder Art repräsentiert den ‘Parent’ des Reminders

this.latestReminder = data.map(item => {

let data = <PanelBarItemModel>{

title: item.title,

id: item.id.toString(),

children: []

};

//Nun kommen die eigentlichen Reminder der jeweiligen Art als ‘Childs’ hinzu

this.queryChildItems(item.id, this.startDateString, this.endDateString)

.subscribe(child => {

data.children = child.map(child => {

return <IReminder>{

id: child.id,

artId: child.artId.toString(),

statusId: child.statusId,

status: child.status,

dateformatted: child.dateformatted,

datum: child.datum,

text: child.text,

sonstiges: child.sonstiges

};

});

});

return data;

});

}, (errData) => {

// error

console.log("Die Daten für den Reminder konnten nicht geladen werden!: ", errData);

}, () => {

// complete

});

});

});

}

//Die Abfrage der Reminder ‘Childs’ der jeweiligen Reminder Art

private queryChildItems(artId: number, startDate: string, endDate: string): any {

return this.reminderService.getLatestReminder(artId, startDate, endDate);

}

## Unit-Tests

Der Unit Test für die „Reminder App“ sieht wie folgt aus:

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { ReminderPanelbarComponent } from './reminder-panelbar.component';

import { NO\_ERRORS\_SCHEMA } from '@angular/core';

import { ReminderService } from "../reminder.service";

//fdescribe tell karma to only test this component

fdescribe('ReminderPanelbarComponent', () => {

let component: ReminderPanelbarComponent;

let fixture: ComponentFixture<ReminderPanelbarComponent>;

let reminderServiceStub = {

userId: 1862 //Dummy

};

beforeEach(async(() => {

TestBed.configureTestingModule({

providers: [{provide: ReminderService, useValue: reminderServiceStub }],

declarations: [ReminderPanelbarComponent],

schemas: [NO\_ERRORS\_SCHEMA]

})

.compileComponents();

}));

beforeEach(() => {

fixture = TestBed.createComponent(ReminderPanelbarComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should be created', () => {

expect(component).toBeTruthy();

});

});

Diese Unit Tests können individuell angepasst werden.

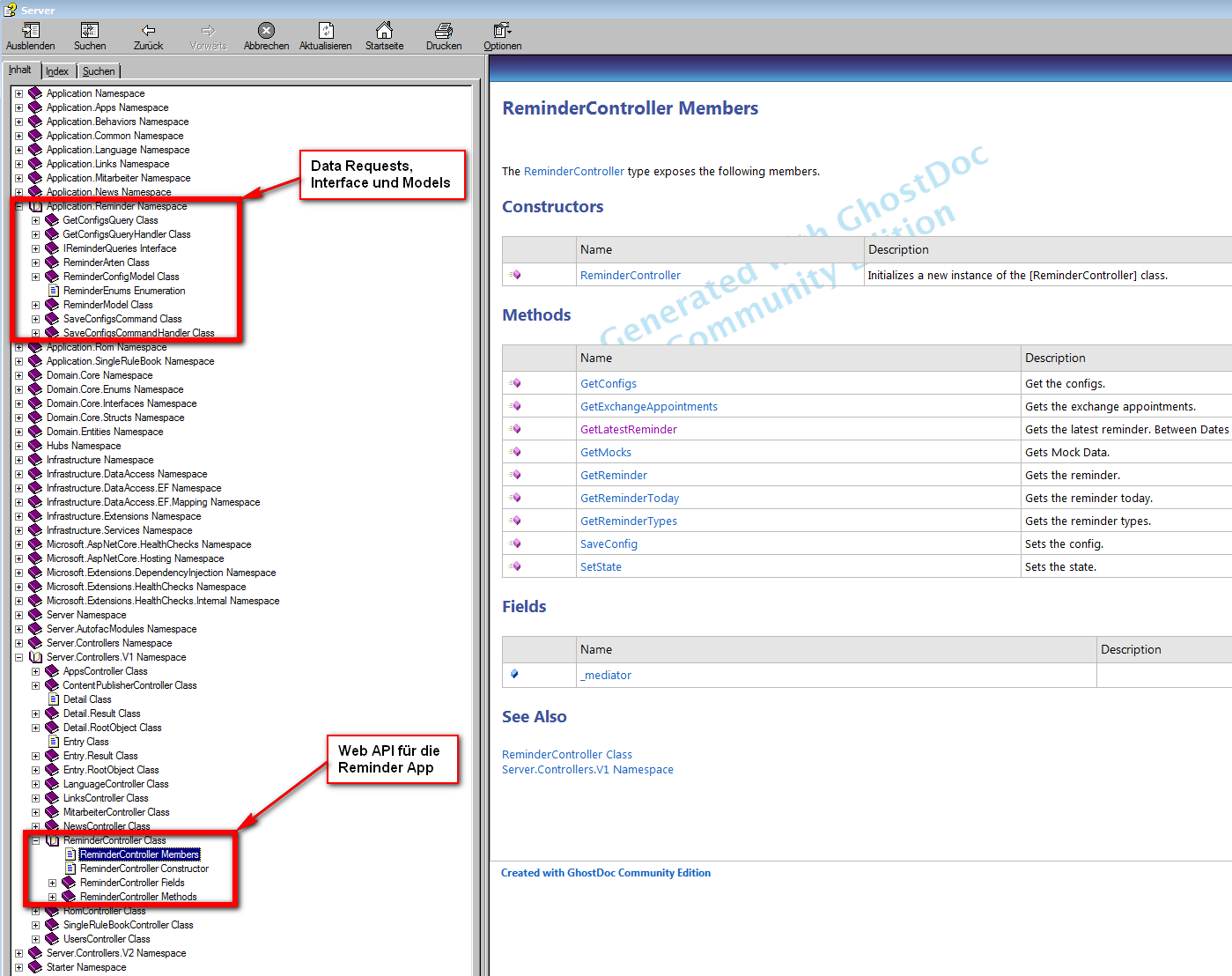
In diesem Fall wird getestet ob die Komponente initialisiert werden kann.

Auf der Kommandozeile von Windows in der Entwicklungsumgebung kann der Node Befehl „npm test“ ausgeführt werden. Darauf öffnet sich nach einer Zeit der Browser, der Test wird durchgeführt und es wird ein durch das „Karma“ Framework eine HTML Datei mit den Ergebnissen generiert.

Sobald die „Reminder App“ in das Produktive Intranet übergeht, wird dieser Testvorgang in den Bauprozess des TFS (Team Foundation Server) integriert und somit vor jedem veröffentlichen der Projektmappe automatisch durchgeführt.

## Entwicklerdokumentation

Ein Ausschnitt aus der generierten Dokumentation des Webservices.

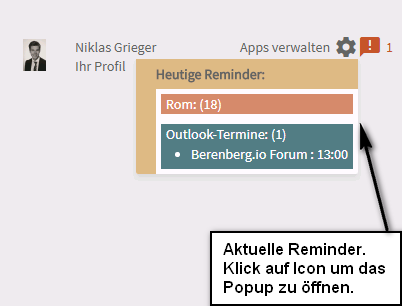


Die Funktionalitäten für die „Reminder App“ sind in dem Bild markiert.

## Benutzerdokumentation

Dies ist die Standard Anzeige der Sidebar im neuen Intranet Webportal. Sie können die „Reminder App“ über den Button „Apps verwalten“ ein- und ausblenden.

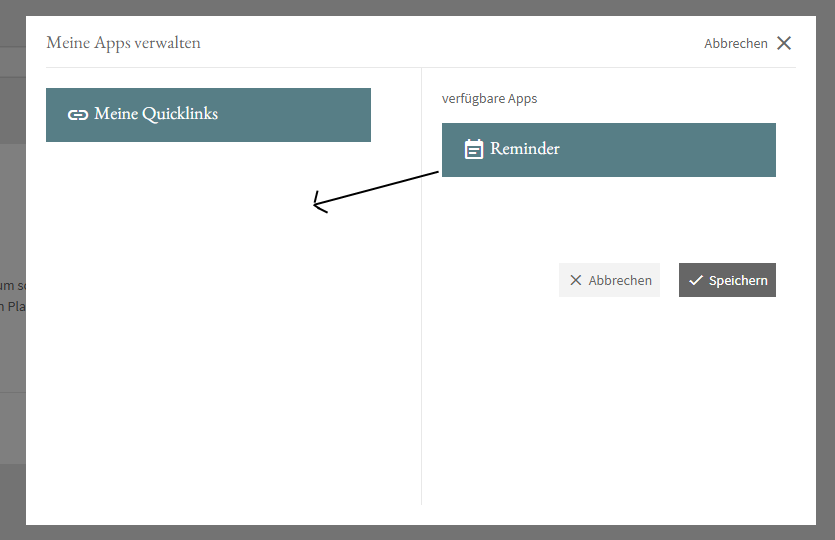
Außerdem sehen Sie, auch wenn die „Reminder App“ ausgeblendet ist, die Reminder mit einer heutigen Deadline. Die Anzeige der heutigen Reminder können Sie, über Klick auf das Icon mit der Anzahl der Reminder, bequem ein- und ausklappen.



Um die „Reminder App“ einzublenden, klicken Sie auf „Apps verwalten“ und es wird sich folgender Dialog öffnen:

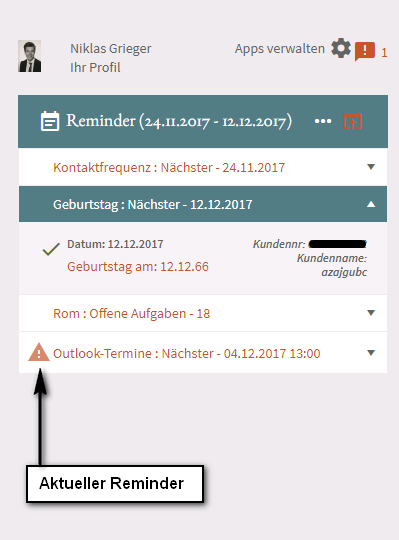
Per „Drag and Drop“ können sie den Reminder in Ihre Favoritenliste schieben.

Speichern Sie nun Ihre Auswahl.



Sobald Sie Ihre Auswahl gespeichert haben, schließt sich der Dialog und sie bekommen die App in der Sidebar angezeigt:

Hier sehen Sie eine Ansicht der „Reminder App“.

U:\Abschlussprojekt v2\Dokumentation\img\ic_more_horiz_black_24dp\web\ic_more_horiz_black_24dp_1x.pngDer Zeitraum, in dem die Reminder angezeigt werden, kann über die Einstellungen geändert werden. Diese Einstellungen können Sie über das Icon aufrufen.

Sie können die verschiedenen Reminder Arten beliebig auf- und zuklappen.

Per Klick auf das „Haken Symbol“ können Sie Reminder auf den Status „Erledigt setzen“.

Ebenfalls Ihre zugeteilten ROM Aufgaben und die nächsten 5 Termine ihres Outlook Kalenders sind verfügbar.

Bei dem ROM-Reminder werden alle unbearbeiteten Aufgaben angezeigt, unabhängig von dem ausgewählten Zeitraum.

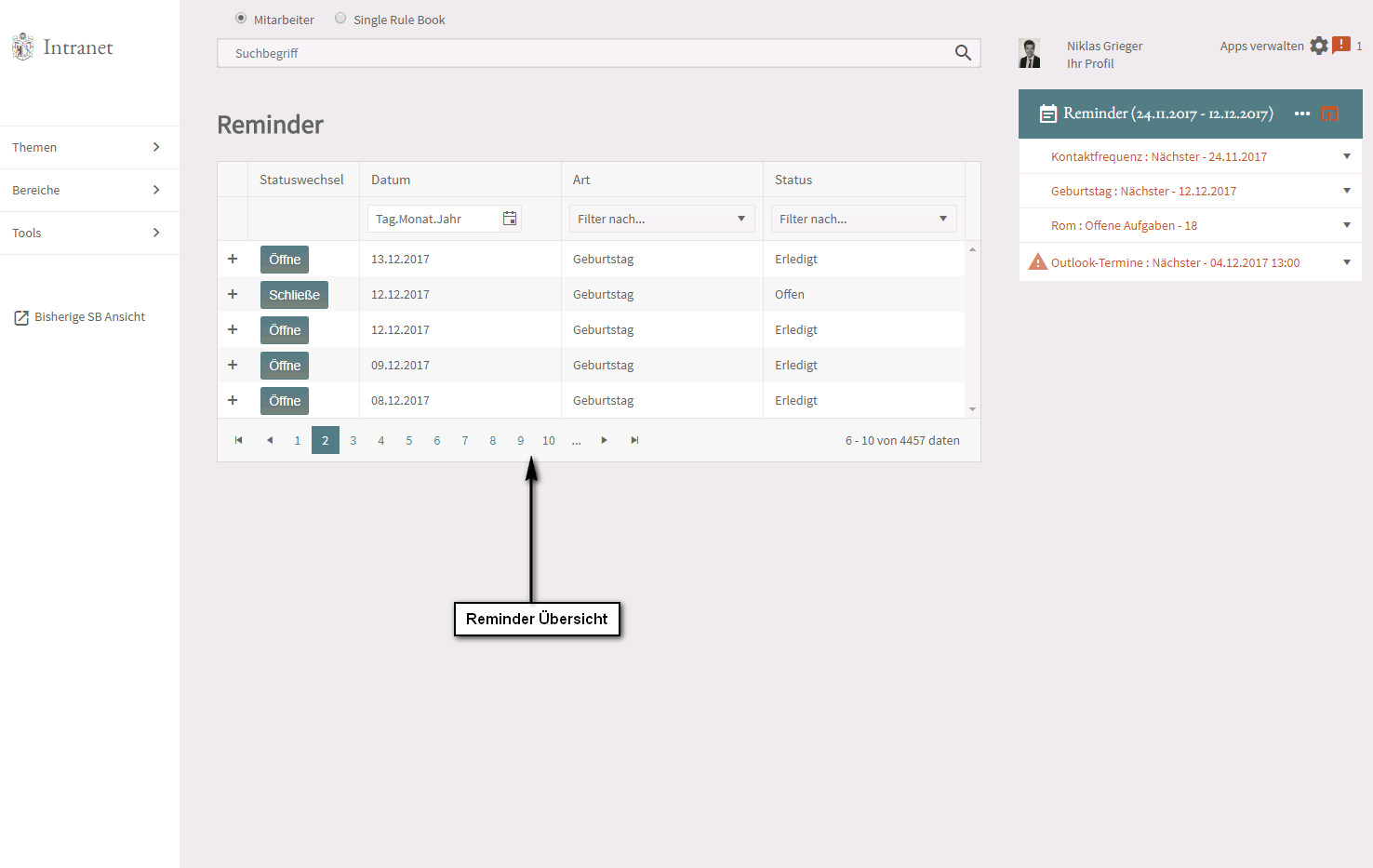
U:\Abschlussprojekt v2\Dokumentation\img\ic_open_in_browser_black_24dp\web\ic_open_in_browser_black_24dp_1x.png

Sie können über das Icon rechts in dem Titel der „Reminder App“ zu der Reminder-Übersicht navigieren.

Dort sehen Sie alle Reminder, die Ihnen zugeteilt sind.

Sie können die Reminder nach Datum, Art oder Status filtern und sortieren.

Diese Übersicht sieht wie folgt aus:



Ansprechpartner

|  |
| --- |
| Niklas Grieger  Auszubildender  niklas.grieger@berenberg.de |

|  |  |  |
| --- | --- | --- |
| Klicken Sie hier, um Text einzugeben. |  | Joh. Berenberg, Gossler & Co. KG Neuer Jungfernstieg 20 20354 Hamburg Telefon +49 40 350 60-0 Telefax +49 40 350 60-900 www.berenberg.de info@berenberg.de |
|  |  |  |