WDB  
Konzeptbericht

| Auftraggeber | Ninck Georg |
| --- | --- |
| Projektleiter | Häberli Joel |
| Autor | Albrecht Miro, Häberli Joel, Schor David, Keller Anuraly |
| Klassifizierung | *Nicht klassifiziert, ~~Intern, Vertraulich, GEHEIM~~* |
| Status | *In Arbeit, ~~Genehmigt~~* |
|  |  |

Änderungsverzeichnis

| Datum | Version | Änderung | Autor |
| --- | --- | --- | --- |
| 31.10.2017 | 0.1 | Kapitel 2.1, 3 | Albrecht Miro, Häberli Joel |
| 05.11.2017 | 0.2 | Kaptiel 4 | Albrecht Miro, Häberli Joel |
| 07.11.2017 | 1.0 | Restliche Kapitel, Qualitätskontrolle | Albrecht Miro, Häberli Joel, Schor David, Keller Anuraly |

**Inhaltsverzeichnis**

1 Zusammenfassung 3

2 Systemanforderungen 4

2.1 Fachliche Entitätstypen 4

2.1.1 User 4

2.1.2 Page 4

2.1.3 Tag 4

2.2 Use Case Diagram 5

2.3 Anwendungsfall «Neues Benutzerkonto erstellen» 5

2.4 Anwendungsfall «Anmelden» 6

2.5 Anwendungsfall «Page erstellen» 6

2.6 Anwendungsfall «Tags hinzufügen» 6

2.7 Anwendungsfall «Bearbeiten von Tags» 7

2.8 Anwendungsfall «Page bearbeiten» 7

2.9 Anwendungsfall «Page suchen» 7

2.10 Anwendungsfall «Page löschen» 8

2.11 Anwendungsfall «Page Exportieren» 8

2.12 Anwendungsfall «Passwort ändern» 8

2.13 Anwendungsfall «Benutzer deaktivieren (löschen)» 9

2.14 Anwendungsfall «Benutzer reaktivieren» 9

3 Benutzerschnittstelle 11

4 Systemarchitektur 13

4.1 Gliederung der Lösung 13

1.1.1 Application-Layers 13

1.1.2 Pakete 13

4.2 Schnittstellen 15

4.2.1 Datenformat 15

4.2.2 Beispiel 15

4.2.3 Entitäten 15

4.2.4 Aufrufe 15

5 Qualitätssicherung 17

6 Projektplanung 17

# Zusammenfassung

Benutzer der Wissensdatenbank sollen Pages erstellen und suchen können. Um das Suchen zu erleichtern, kann jede Page mit Tags kategorisiert werden. Aus diesen Anforderungen haben wir drei Entitäten definiert: User, Page und Tag. Die Schnittstelle der Wissensdatenbank hat mit Attachement noch eine vierte Entität.

Unsere Applikation hat keine grafische Benutzerschnittstelle. Wir werden aber einen Beispielclient bauen. Wir haben also trotzdem eine grafische Präsentation skizziert.

Der wichtigste Teil ist unsere REST-basierte Web-API. Deshalb werden wir die Schnittstelle auch am umfangreichsten testen. Das wird durch die Entwicklung des Beispielclients sichergestellt.

Wir strukturieren die Wissensdatenbank nach dem Layer-Entwurfsmuster.

# Systemanforderungen

## 2.1 Fachliche Entitätstypen

### User

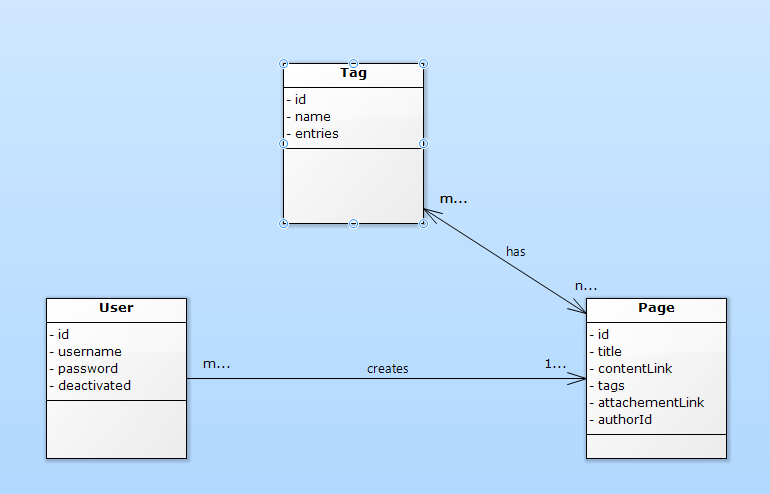
Ein User hat einen Usernamen und ein Passwort, mit dem er sich anmelden kann. Zudem hat er einen Status, ob der Account aktiv ist oder nicht. Eindeutig identifiziert wird er durch eine ID. Ein User kann keine oder mehrere Pages haben.

### Page

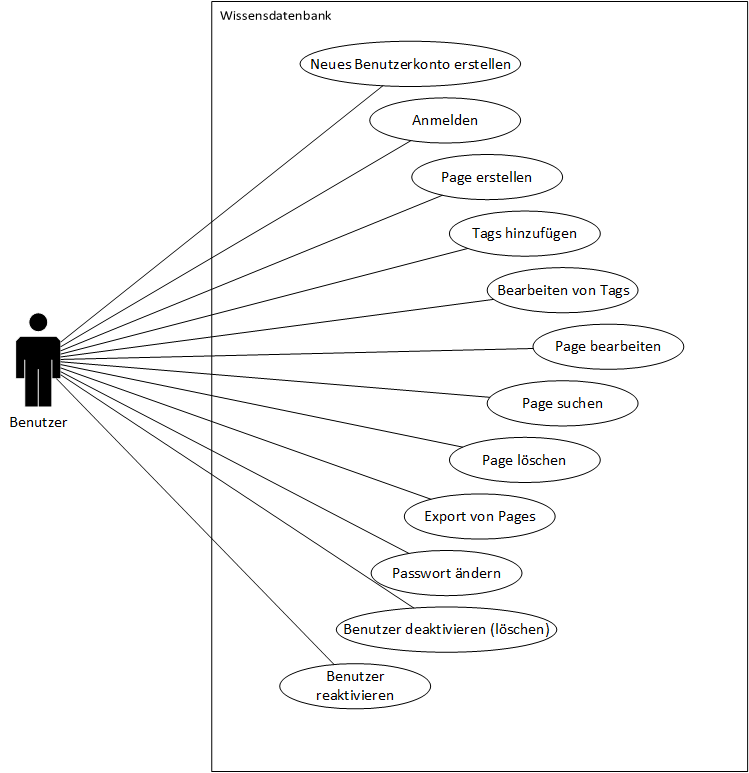
Die Page ist für das Speichern der Inhalte zuständig. Sie beinhaltet Titel und Inhalt, so wie die Möglichkeit, Dateien als Anhang zu speichern. Jede Page hat einen Autor. Auch die Page wird durch eine ID eindeutig identifizierbar. Damit man Inhalte leichter finden kann, kann man Tags vergeben.

### Tag

Tags haben einen Namen und eine ID, durch welche sie eindeutig identifizierbar werden. Jeder Tag kann beliebig viele Verknüpfungen mit Pages haben.



## Use Case Diagram



## Anwendungsfall «Neues Benutzerkonto erstellen»

|  |  |
| --- | --- |
| **Anwendungsfall** **„Neues Benutzerkonto erstellen“** | |
| Kurzbeschreibung | Ein Benutzer registriert sich und erstellt dabei einen neuen Benutzeraccount |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat noch kein Benutzerkonto  Der Benutzer hat die Kommandozeile geöffnet |
| Ablauf | 1. Der Benutzer gibt «register –name "Name" –passwort "Passwort"» ein 2. Das System überprüft, ob bereits dieser Username existiert 3. Das System verlangt die Wiederholung des Passworts 4. Der Benutzer gibt sein Passwort nochmals ein 5. Das System erstellt den neuen Account |
| Resultat | Ein neuer Benutzeraccount wurde erstellt |
| Ausnahmen | * Der Benutzername existiert bereits (Schritt 2) * Der Benutzer gibt sein Passwort falsch ein (Schritt 4) * Es gab einen Fehler bei der Internet-Verbindung * Der Command wurde falsch eingegeben (Schritt 1) |

## Anwendungsfall «Anmelden»

|  |  |
| --- | --- |
| **Anwendungsfall „Anmelden“** | |
| Kurzbeschreibung | Ein Benutzer kann sich mit einem vorhandenen Konto anmelden. |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat ein Benutzerkonto  Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt «login –name "Name" –passwort "Passwort"» ein 2. Das System überprüft die Eingaben 3. Der Benutzer wird angemeldet |
| Resultat | Der Benutzer wird angemeldet |
| Ausnahmen | * Der Name oder das Passwort wurde falsch eingegeben (Schritt 1) * Es gab einen Fehler bei der Internet-Verbindung * Der Command wurde falsch eingegeben (Schritt 1) |

## Anwendungsfall «Page erstellen»

|  |  |
| --- | --- |
| **Anwendungsfall „neue Page erstellen“** | |
| Kurzbeschreibung | Der Benutzer kann eine neue Page erstellen. |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt «new-page –title "Website" –tags "PHP"» ein 2. Das System erstellt die neue Page 3. Der User kann seinen Inhalt eingeben und mit "Enter" bestätigen 4. Das System speichert die Page und gibt eine Rückmeldung |
| Resultat | Der Benutzer hat eine Page hinzugefügt. |
| Ausnahmen | * Es gab einen Fehler bei der Internet-Verbindung * Der Command wurde falsch eingegeben (Schritt 1) |

## Anwendungsfall «Tags hinzufügen»

|  |  |
| --- | --- |
| **Anwendungsfall „Tags hinzufügen“** | |
| Kurzbeschreibung | Ein Benutzer fügt Tags zu einer Page hinzu. |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt «add-tag "XML-Datei, HTML-Datei, JS-Datei" –page "ID-Nummer"» ein 2. Das System durchsucht die Datenbank nach der angegebenen ID-Nummer 3. Das System fügt die Tags hinzu. 4. Dass System sendet eine Bestätigung |
| Resultat | Die Page hat Tags, welche man mit der Such-Funktion finden kann. |
| Ausnahmen | * Die Angegebene Page existiert nicht (Schritt 2) * Es gaben ein Fehler bei der Internet-Verbindung * Der Command wurde falsch eingegeben (Schritt 1) |

## Anwendungsfall «Bearbeiten von Tags»

|  |  |
| --- | --- |
| **Anwendungsfall „Bearbeiten von Tags“** | |
| Kurzbeschreibung | Der Benutzer bearbeitet die Tags einer Page |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt «edit-tags –page "ID-Nummer"» ein 2. Das System durchsucht die Datenbank nach der angegebenen Page 3. Das System gibt alle Tags dieser Page zurück 4. Der Benutzer gibt «add "PHP"» ein 5. Das System fügt die angegebenen Tags hinzu und gibt eine Bestätigung. 6. Der Benutzer gibt «del "XML"» ein 7. Das System löscht die angegebenen Tags und gibt eine Bestätigung. |
| Resultat | Der Benutzer hat die Tags einer Page bearbeitet |
| Ausnahmen | * Die Angegebene Page existiert nicht (Schritt 2) * Es gab einen Fehler bei der Internet-Verbindung * Der Command wurde falsch eingegeben (Schritt 1, 4, 6) * Der Tag ist nicht vorhanden (Schritt 6) |

## Anwendungsfall «Page bearbeiten»

|  |  |
| --- | --- |
| **Anwendungsfall „Page bearbeiten“** | |
| Kurzbeschreibung | Benutzer kann eine Page bearbeiten |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt «edit-page –page "ID-Nummer"» ein 2. Das System zeigt die Page an 3. Der Benutzer kann den Titel bearbeiten und mit "Enter" bestätigen 4. Der Benutzer kann den Inhalt bearbeiten und mit "Enter" bestätigen 5. Das System speichert die Änderungen und gibt eine Rückmeldung |
| Resultat | Die ausgewählte Page wurde bearbeitet und abgespeichert |
| Ausnahmen | * Es gab einen Fehler bei der Internet-Verbindung * Der Command wurde falsch eingegeben (Schritt 1) |

## Anwendungsfall «Page suchen»

|  |  |
| --- | --- |
| **Anwendungsfall „Page suchen“** | |
| Kurzbeschreibung | Benutzer kann eine Page suchen. |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt «search –title "Website" –tags "PHP"» ein 2. Das System durchsucht die Datenbank nach dem Title und dem Tag 3. Das System zeigt die gefunden Pages mit diesen Angaben an |
| Resultat | Wenn die Suche beendet ist, zeigt das System die Ergebnisse. |
| Ausnahmen | * Es gab einen Fehler bei der Internet-Verbindung * Der Command wurde falsch eingegeben (Schritt 1) * Es existiert keine Page mit diesen Angaben (Schritt 2) |

## Anwendungsfall «Page löschen»

|  |  |
| --- | --- |
| **Anwendungsfall „Page löschen“** | |
| Kurzbeschreibung | Benutzer kann seine Pages löschen. |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt „delete-page –page "ID-Nummer"“ ein 2. Das System fragt nochmals, ob der Benutzer diese Page löschen möchte 3. Der Benutzer bestätigt und das System löscht die angegebene Page |
| Resultat | Die Page wurde gelöscht |
| Ausnahmen | * Es gab einen Fehler bei der Internet-Verbindung * Der Command wurde falsch eingegeben (Schritt 1) * Der Benutzer bricht bei der Nachfrage ab (Schritt 2) |

## Anwendungsfall «Page Exportieren»

|  |  |
| --- | --- |
| **Anwendungsfall „** **Page Exportieren“** | |
| Kurzbeschreibung | Ein Benutzer exportiert eine Page. |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt «export "ID-Nummer"» ein 2. Das System durchsucht die Datenbank nach der angegebenen Page 3. Das System sendet den Inhalt der Page als Datei zurück |
| Resultat | Der Benutzer hat eine exportierte Page. |
| Ausnahmen | * Die Angegebene Page existiert nicht (Schritt 2) * Es gab einen Fehler bei der Internet-Verbindung * Der Command wurde falsch eingegeben (Schritt 1) * Es gab ein Fehler bei der Zusammenstellung der Datei (Schritt 3) |

## Anwendungsfall «Passwort ändern»

|  |  |
| --- | --- |
| **Anwendungsfall „Passwort ändern“** | |
| Kurzbeschreibung | Der Benutzer ändert sein Passwort |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt «change-password» ein 2. Das System fordert das alte Passwort 3. Der Benutzer gibt sein altes Passwort ein 4. Das System bestätig, dass das Passwort korrekt war und fordert das neue Passwort 5. Der Benutzer gibt sein neues Passwort an 6. Das System speichert das neue Passwort |
| Resultat | Das Passwort des Benutzers wurde geändert |
| Ausnahmen | * Der Command wurde falsch eingegeben (Schritt 1) * Es gab einen Fehler bei der Internet-Verbindung * Das alte Passwort ist falsch (Schritt 3) |

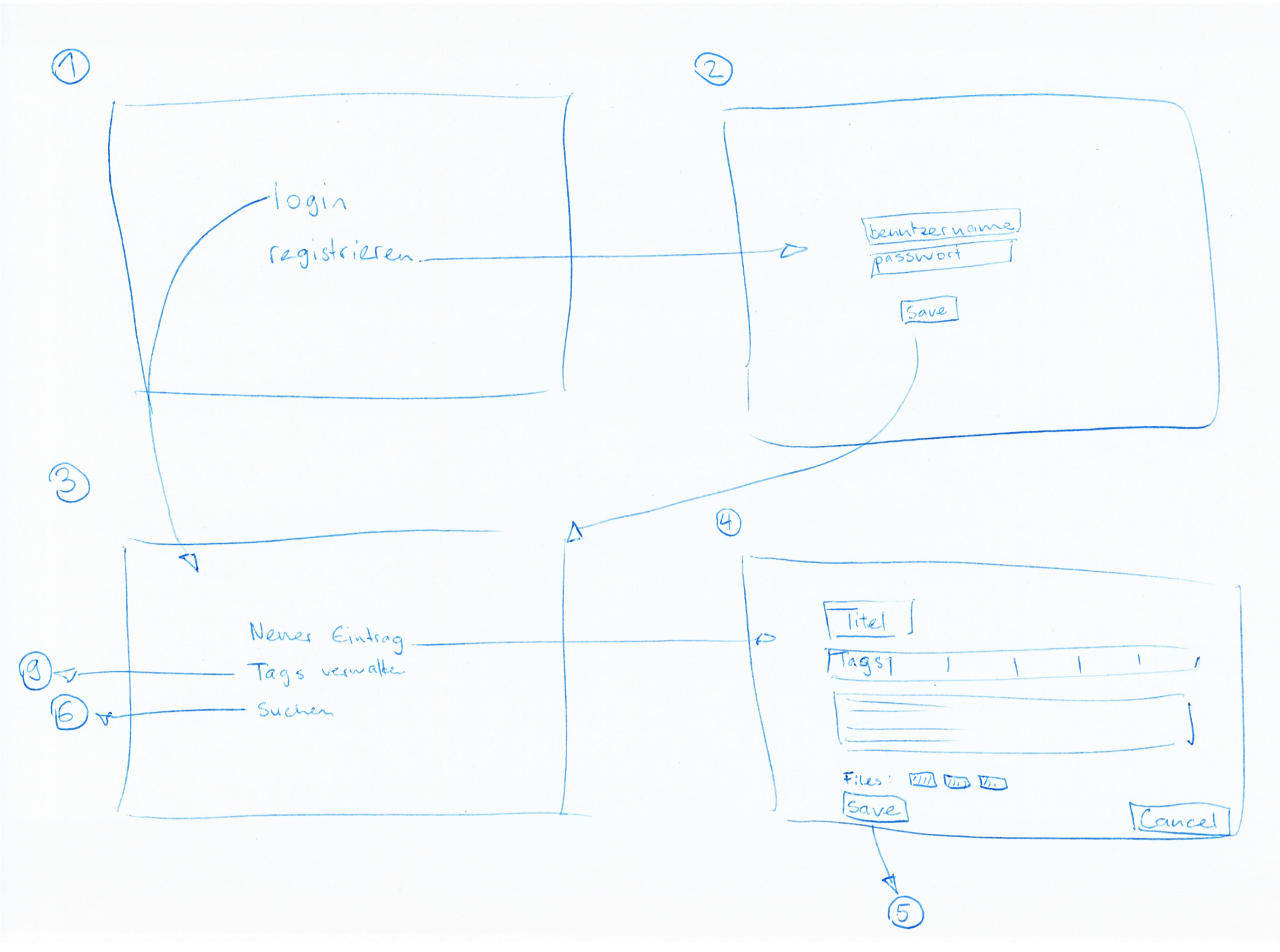
## Anwendungsfall «Benutzer deaktivieren (löschen)»

|  |  |
| --- | --- |
| **Anwendungsfall „Benutzer deaktivieren (löschen)“** | |
| Kurzbeschreibung | Der Account des Benutzers wird deaktiviert (gelöscht) |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. |
| Ablauf | 1. Der Benutzer gibt «del-account» ein 2. Das System fordert das Passwort 3. Der Benutzer gibt sein Passwort ein 4. Das System deaktiviert den Account und sendet eine Bestätigung |
| Resultat | Der Account des Benutzers wurde deaktiviert (gelöscht) |
| Ausnahmen | * Der Command wurde falsch eingegeben (Schritt 1) * Es gab einen Fehler bei der Internet-Verbindung * Das Passwort ist falsch (Schritt 3) |

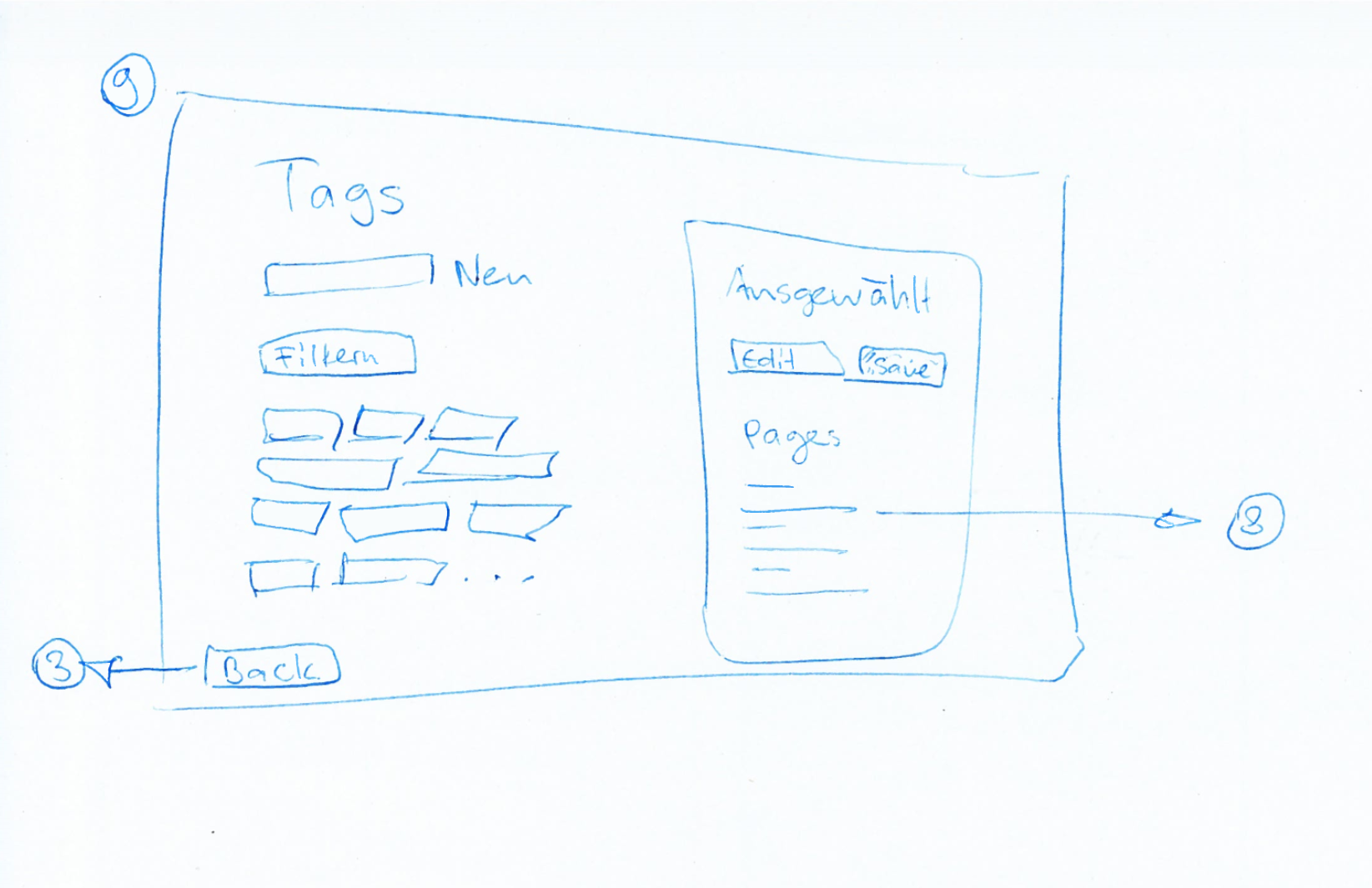
## Anwendungsfall «Benutzer reaktivieren»

|  |  |
| --- | --- |
| **Anwendungsfall „Benutzer reaktivieren“** | |
| Kurzbeschreibung | Der Benutzer reaktiviert seinen Account |
| Akteure | Benutzer |
| Vorbedingungen | Der Benutzer hat die Kommandozeile geöffnet. Der Benutzer hat sein Account deaktiviert (gelöscht). Siehe Anwendungsfall „Benutzer deaktivieren (löschen)“ |
| Ablauf | 1. Der Benutzer gibt «activate-account» 2. Das System durchsucht die Datenbank nach dem Benutzer und überprüft, ob dieser länger als 2 Wochen deaktiviert (gelöscht) ist 3. Das System fordert das Passwort 4. Der Benutzer gibt sein Passwort ein 5. Das System reaktiviert den Account und sendet eine Bestätigung |
| Resultat | Der Account des Benutzers wurde reaktiviert |
| Ausnahmen | * Der Command wurde falsch eingegeben (Schritt 1) * Es gab einen Fehler bei der Internet-Verbindung * Das Passwort ist falsch (Schritt 4) * Der Benutzer wurde nicht deaktiviert (gelöscht) * Der Account ist länger als 2 Wochen deaktiviert (gelöscht) (Schritt 2) |

# Benutzerschnittstelle







# Systemarchitektur

## Gliederung der Lösung

### Application-Layers

Unsere Applikation ist in vier Schichten unterteilt. Zu Oberst ist die Darstellungsschicht. Diese wird jedoch nur zu Test- und Präsentationszwecken erstellt. Dann kommt der API-Controller welcher die Schnittstelle zwischen UI und Backend bildet. Der API-Controller kommuniziert mit der Businesslogik-Schicht. Diese wiederum greift auf die Datenbank-Schicht. Die Datenbank-Schicht ist die unterste Schicht.

### Pakete

Die Pakete sind stark an den Schichten orientiert. Zusätzlich gibt es Hilfspakete, welche die Arbeiten auf den verschiedenen Schichten unterstützen. Sie wurden abgespalten, da es technische Aufgaben sind, welche man in Zukunft eventuell austauschen möchte. Folgend sind die Pakete aufegelistet und die jeweiligen Aufgaben kurz beschrieben.

Ui

Das Paket „ui“ ist zuständig, die Darstellungsschicht zu bedienen.

Api

Im Paket „api“ ist nur die Schnittstelle zwischen UI und Backend zu finden. Sie bildet das Herzstück der Applikation. Wenn diese nicht funktioniert, dann geht nichts.

Controller

Im Paket „controller“ sind alle Controller zu den fachlichen Entitäten zu finden. Sie beinhalten die Businesslogik der Anwendung

Parse

Das Paket „parse“ ist eines der oben genannten technischen Pakete. Es formatiert Daten in versandfertige Pakete und verwandelt Daten, die als Input reinkommen, in Objekte, welche das Backend versteht.

Connection

Das Paket „connection“ ist ebenfalls eine technische Klasse und betreut die Verbindung zwischen Backend und Datenbank. Mit diesem Zwischenteil zwischen Backend und Datenbank wird die Datenbank austauschbar.

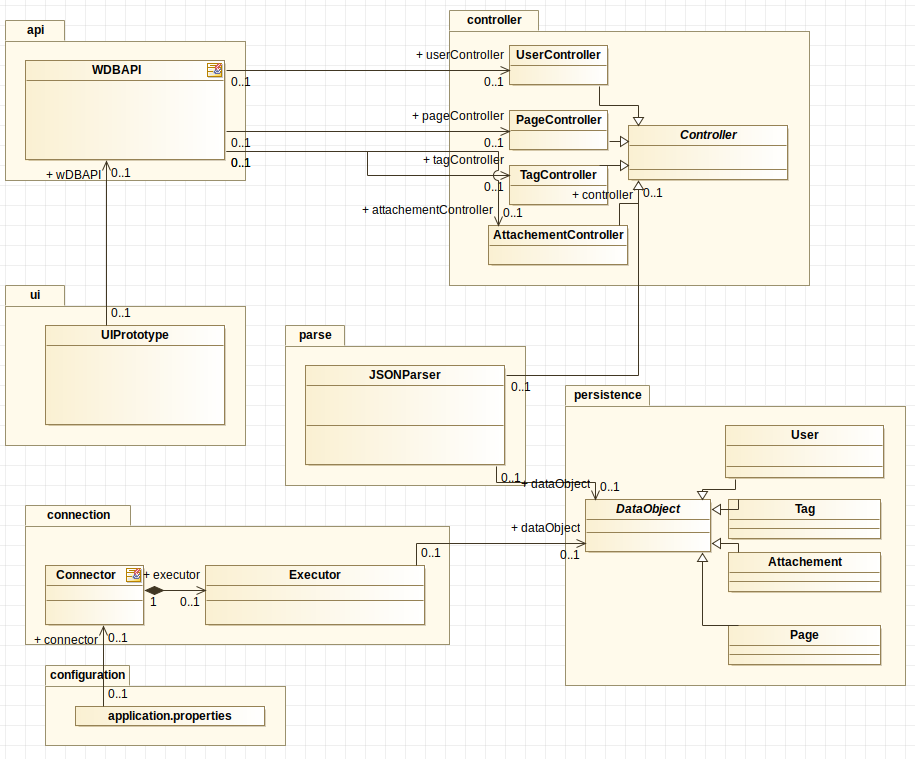
Persistence

Das Paket „persistence“ bildet die Datenbank in der Applikation ab. Es ist für die richtige Speicherung der Daten während der Laufzeit zuständig.

Configuration

Im Paket „configuration“ werden globale Konfigurationen festgehalten. Auch dieses Paket ist technisch. Anders als die anderen Pakete, welche klare Beziehungen zueinander haben, kann jedes Paket aus dem Paket „configuration“ Informationen beziehen.

Auf folgendem Diagramm sind die Packages und ihre Klassen und Beziehungen visualisiert:



## Schnittstellen

Unsere Applikation bietet eine REST-basierte HTTP-Api an. Die Schnittstelle wird also durch eine HTTP-Request aufgerufen. Wir folgen dem üblichen Schema. Unsere Api wird unter dem (relativen) Pfad /api/v1 verfügbar sein.

Für die Version 1.0 haben wir nur die Entitäten User, Page und Tag definiert. Dadurch bleibt auch die Schnittstelle überschaubar.

### Datenformat

Der Datenaustausch zwischen der Wissensdatenbank und einem Client findet immer mit dem JSON-Format statt. Das ist einheitlich und es gibt für fast alle Plattformen JSON-Bibliotheken.

### Beispiel

REST verwendet den URL um das Ziel zu bestimmen. Wie die Applikation genau reagiert, bestimmen die HTTP-Verben (GET, POST, PUT, DELETE).

Um den Inhalt einer Page zu laden, muss die Id (zum Beispiel 01293) der Page bekannt sein.

GET /api/v1/page/01293/

Die Antwort wird als JSON zurückgeschickt.

### Entitäten

Die Schnittstelle kennt folgende Entitäten:

|  |  |
| --- | --- |
| **Entität** | **Attribute** |
| Tag | Id, Label |
| Page | Id, Title, Content, Attachement[], Author, Tag[] |
| User | Id, Name, Active, PageId[] |
| Attachement | Id, FileName, Size (, Date) |

Attribute mit zwei eckigen Klammern am Schluss sind Arrays.

### Aufrufe

Die API-Aufrufe sollten möglichst wenig verschachtelt werden, damit die Schnittstelle nicht zu komplex wird.

Wir ziehen eine praktische Anwendung einer exakten implementierung der REST-Standards vor.

Die geschweiften Klammern stellen den Http-Body dar. Id steht immer für eine Identifikationsnummer der Entität, nicht für die Buchstaben id.

|  |  |
| --- | --- |
| GET (Lesen)  Alle Get-Aufrufe können mit Paramter verfeinert werden. Beispiel (sucht nach Pages mit «HTML» im Titel): page?title=html  Alle Get-Aufrufe können in «Seiten» aufgeteilt werden. Dazu dienen die Parameter offset und count. Beispiel (liefert Page 50 - 100): page?offset=50&count=50 | |
| tag | Alle Tags |
| tag/id | Tag mit Id |
| tag/id/page | Alle Pages mit Tag |
| page | Alle Pages |
| page/id | Page mit Id |
| attachement/id | Attachement mit Id |
| attachement/id/file | Die Datei |
| attachement/id/page | Page des Attachement |
| user | Alle User |
| user/id | User mit Id |
| user/id/page | Alle Pages des Users |

|  |  |
| --- | --- |
| POST (Erstellen) | |
| tag { Label } | Erstelle Tag |
| page { Title, Content, AuthorId, TagIds[] } | Erstelle Page |
| user { Name, Password } | Erstelle User |
| page/id/tag { TagId } | Tag einer Page hinzufügen |
| page/id/attachement { Attachement } | Attachement einer Page hinzufügen |
| login { Name, Password } | Erstellt eine Session für einen User |
| logout | Löscht die Session |

|  |  |
| --- | --- |
| PUT (Ändern) | |
| tag/id { Label } | Ändert Label des Tags |
| page/id { Title, Content } | Ändert Titel/Content der Page |
| user/id { Old Password, New Password } | Ändert das Passwort |

|  |  |
| --- | --- |
| DELETE (Löschen) | |
| tag/id | Löscht den Tag (wenn keine Page mit diesem Tag existiert) |
| page/id | Löscht die Page |
| user { Password } | Löscht den eigenen Account |
| page/id/tag { TagId } | Entfernt einen Tag von einer Page |
| page/id/attachement { AttachementId } | Entfernt Attachement von einer Page |
| attachement/id | Löscht Attachement |

# Qualitätssicherung

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Abgedeckter Anwendungsfall** | **Beschreibung** |
| 1 | Neues Benutzerkonto erstellen | Der Benutzer kann ein Account erstellen, sofern er noch keines besitzt und die Daten valide sind. Anschliessend wird er eingeloggt. |
| 2 | Anmelden | Der Benutzer kann sich mit seinen Login Daten anmelden, er wird dann auf die nächste Page weitergeführt. |
| 3 | Page erstellen | Der Benutzer kann eine neue Page mit Inhalten erstellen. Dies wird beim Speichern mit einer Meldung quittiert. |
| 4 | Tags hinzufügen | Der Benutzer gibt die Page und die gewünschten Tags an. Die Tags werden der Page hinzugefügt. Die Page ist nun durch die Tags besser auffindbar. |
| 5 | Bearbeiten von Tags | Der Benutzer bearbeitet die Tags. Der Benutzer kann anschliessend die Page durch die bearbeiteten Tags wiederfinden. |
| 6 | Page bearbeiten | Der Benutzer kann seine Page bearbeiten. Nach dem Speichern wird die neue Version abgespeichert, der Benutzer bekommt eine entsprechende Meldung. |
| 7 | Page suchen | Der Benutzer kann eine Page nach Titel, Tag und Inhalt suchen. |
| 8 | Page löschen | Der Benutzer kann seine Page löschen. Nach einer Meldung wird dies ihm bestätigt und die Page ist gelöscht. |
| 9 | Export von Pages | Der Benutzer kann seine eigenen Pages exportieren und anderen zur Verfügung stellen, indem er den Export weiterversendet. |
| 10 | Passwort ändern | Der Benutzer ändern sein Passwort und wird anschliessend ausgeloggt und muss sich mit dem neuen Passwort anmelden. |
| 11 | Benutzer deaktivieren (löschen) | Der Benutzer kann sein Account deaktivieren. Anschliessend wird er abgemeldet. Die Pages werden nicht gelöscht. |
| 12 | Benutzer reaktivieren | Der Benutzer kann sein Account, wenn dieser nicht mehr als vor 2 Wochen deaktiviert wurde, wieder aktivieren. Dazu muss er sich ganz normal anmelden und die Deaktivierung wird aufgehoben. Wenn der Account länger als 2 Wochen deaktiviert ist, ist es nicht mehr möglich diesen zu reaktivieren. |

# Projektplanung

Siehe Projektplan, Version 2.0 vom 07.11.2017