### API-Spezifikationen

| **Feld** | **Beschreibung** |
| --- | --- |
| **Name des Artefakts** | API-Spezifikationen |
| **Verantwortlich** | IT-Systemverantwortlicher |
| **Ziel des Dokuments** | Beschreibung der Schnittstellen zu anderen Systemen zur Gewährleistung einer nahtlosen Integration. |
| **Wesentliche Inhalte** | - Endpunkte und Routen- Anfrage- und Antwortformate- Authentifizierungs- und Autorisierungsmethoden |
| **Erstellungszeitpunkt** | Planungsphase des Projekts, nach Erstellung der technischen Spezifikationen |
| **Nutzung im Projektverlauf** | Grundlage für die Entwicklung und Integration der Schnittstellen |
| **Abnahme** | Durch den IT-Systemverantwortlichen und die Entwickler; Überprüfung der Vollständigkeit und Korrektheit der Spezifikationen |
| **Abhängigkeiten** | - Technische Spezifikationen- Systemarchitektur-Diagramme |
| **Versionierung** | Versionskontrolle mittels Versionsmanagement-Tools; regelmäßige Aktualisierung bei Änderungen der API-Spezifikationen |
| **Vorlagen und Tools** | API-Dokumentationstools (z.B. Swagger, Postman) |
| **Zusätzliche Anmerkungen** | Enge Zusammenarbeit zwischen den Entwicklern erforderlich, um sicherzustellen, dass die API-Spezifikationen korrekt und vollständig umgesetzt werden |

## API-Spezifikationen für die Erweiterung des CRM-Systems

### 1. Überblick

| **Feld** | **Beschreibung** |
| --- | --- |
| **Projektname** | Erweiterung des CRM-Systems um Scoring-Daten |
| **Projektleiter** | Max Mustermann |
| **IT-Systemverantwortlicher** | Maria Musterfrau |
| **Fachbereich** | Vertrieb |
| **Erstellungsdatum** | 19. Mai 2024 |
| **Version** | 1.0 |

### 2. Ziel

Das Ziel dieses Dokuments ist es, die Spezifikationen der RESTful API zu beschreiben, die für die Erweiterung des CRM-Systems um die Befüllung mit Scoring-Daten benötigt wird.

### 3. API-Übersicht

Die API ermöglicht es Benutzern, Scoring-Daten per Excel-Datei hochzuladen und diese Daten automatisch den entsprechenden Kunden im CRM-System zuzuordnen und zu speichern.

### 4. Endpunkte

#### 4.1 POST /upload

**Beschreibung**: Dieser Endpunkt empfängt eine Excel-Datei mit Scoring-Daten und validiert die Daten.

| **Feld** | **Beschreibung** |
| --- | --- |
| **URL** | /upload |
| **Methode** | POST |
| **Content-Type** | multipart/form-data |
| **Authentifizierung** | Erforderlich (Bearer Token) |

**Anfrage-Header**:

plaintext

Authorization: Bearer <token>

Content-Type: multipart/form-data

**Anfrage-Parameter**:

* **file**: Die hochzuladende Excel-Datei (erforderlich)

**Beispiel-Anfrage**:

plaintext

POST /upload HTTP/1.1

Host: api.crm-system.com

Authorization: Bearer <token>

Content-Type: multipart/form-data

--boundary

Content-Disposition: form-data; name="file"; filename="scoring\_data.xlsx"

Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet

<Binary Data>

--boundary--

**Antwort (Erfolg)**:

json

{

 "status": "success",

 "message": "File uploaded and data validated successfully."

}

**Antwort (Fehler)**:

json

{

 "status": "error",

 "message": "File validation failed.",

 "errors": [

 {

 "row": 5,

 "column": "Kunden\_ID",

 "error": "Kunden\_ID not found."

 }

 ]

}

#### 4.2 GET /customers/{id}/score

**Beschreibung**: Dieser Endpunkt liefert den Scoring-Wert für einen bestimmten Kunden.

| **Feld** | **Beschreibung** |
| --- | --- |
| **URL** | /customers/{id}/score |
| **Methode** | GET |
| **Content-Type** | application/json |
| **Authentifizierung** | Erforderlich (Bearer Token) |

**Anfrage-Header**:

plaintext

Authorization: Bearer <token>

Content-Type: application/json

**Pfad-Parameter**:

* **id**: Die ID des Kunden (erforderlich)

**Beispiel-Anfrage**:

plaintext

GET /customers/123/score HTTP/1.1

Host: api.crm-system.com

Authorization: Bearer <token>

Content-Type: application/json

**Antwort (Erfolg)**:

json

{

 "customer\_id": 123,

 "scoring\_value": 85.5,

 "last\_updated": "2024-05-18T14:52:00Z"

}

**Antwort (Fehler)**:

json

{

 "status": "error",

 "message": "Customer not found."

}

#### 4.3 PUT /customers/{id}/score

**Beschreibung**: Dieser Endpunkt ermöglicht das manuelle Aktualisieren des Scoring-Werts eines Kunden.

| **Feld** | **Beschreibung** |
| --- | --- |
| **URL** | /customers/{id}/score |
| **Methode** | PUT |
| **Content-Type** | application/json |
| **Authentifizierung** | Erforderlich (Bearer Token) |

**Anfrage-Header**:

plaintext

Authorization: Bearer <token>

Content-Type: application/json

**Pfad-Parameter**:

* **id**: Die ID des Kunden (erforderlich)

**Anfrage-Body**:

json

{

 "scoring\_value": 90.0

}

**Beispiel-Anfrage**:

plaintext

PUT /customers/123/score HTTP/1.1

Host: api.crm-system.com

Authorization: Bearer <token>

Content-Type: application/json

{

 "scoring\_value": 90.0

}

**Antwort (Erfolg)**:

json

{

 "status": "success",

 "message": "Scoring value updated successfully."

}

**Antwort (Fehler)**:

json

{

 "status": "error",

 "message": "Update failed. Customer not found."

}

### 5. Authentifizierung

Die API verwendet Bearer Tokens zur Authentifizierung. Jeder Anfrage muss ein gültiges Token im Authorization-Header mitgeführt werden.

**Beispiel-Header**:

plaintext

Authorization: Bearer <token>

### 6. Fehlerbehandlung

Die API gibt Fehler in einem standardisierten Format zurück.

#### Beispiel-Fehlerantwort (Validierungsfehler):

json

{

 "status": "error",

 "message": "Validation failed.",

 "errors": [

 {

 "field": "Kunden\_ID",

 "error": "Kunden\_ID not found."

 }

 ]

}

#### Beispiel-Fehlerantwort (Authentifizierungsfehler):

json

{

 "status": "error",

 "message": "Unauthorized."

}

### 7. Sicherheitsmaßnahmen

* **HTTPS**: Alle API-Aufrufe müssen über HTTPS erfolgen.
* **Rate Limiting**: Begrenzung der Anzahl der API-Aufrufe pro Benutzer, um Missbrauch zu verhindern.
* **Input Validierung**: Validierung aller Eingabedaten, um SQL-Injection und andere Angriffe zu verhindern.

### 8. Abnahmekriterien

* **Funktionalität**: Alle Endpunkte müssen wie beschrieben funktionieren.
* **Sicherheit**: Alle Sicherheitsmaßnahmen müssen implementiert und wirksam sein.
* **Fehlerbehandlung**: Die API muss alle Fehler korrekt behandeln und die entsprechenden Fehlermeldungen zurückgeben.

### 9. Risiken und Maßnahmen

| **Risiko** | **Maßnahme** |
| --- | --- |
| Sicherheitslücken | Regelmäßige Sicherheitsüberprüfungen und Penetrationstests |
| Dateninkonsistenz | Implementierung von Validierungsregeln und regelmäßige Datenüberprüfung |
| Überlastung der API | Implementierung von Rate Limiting und Skalierungsstrategien |

### 10. Beteiligte Stakeholder

| **Stakeholder** | **Rolle** | **Verantwortlichkeiten** |
| --- | --- | --- |
| API-Entwickler | Entwicklung | Implementierung und Wartung der API |
| IT-Sicherheitsverantwortlicher | Sicherheitsmanagement | Implementierung und Überwachung der Sicherheitsmaßnahmen |
| QA-Tester | Qualitätssicherung | Durchführung von Funktionalitäts- und Sicherheitstests |
| Projektleiter | Projektmanagement | Koordination und Überwachung des Projekts |

### 11. Dokumentation und Versionierung

| **Version** | **Datum** | **Beschreibung** |
| --- | --- | --- |
| 1.0 | 19. Mai 2024 | Initiale Version der API-Spezifikationen |
| 1.1 | XX.XXX.XXXX | Anpassungen basierend auf Feedback und neuen Erkenntnissen |

### 12. Vorlagen und Tools

* **API-Dokumentations-Tools**: Swagger, Postman
* **Entwicklungs-Tools**: Visual Studio Code, WebStorm
* **Test-Tools**: JUnit, Postman