ALTE OLDENBURGER
Private Krankenversicherung

Projektname: ENTON

Prüfling: Christian Eirich

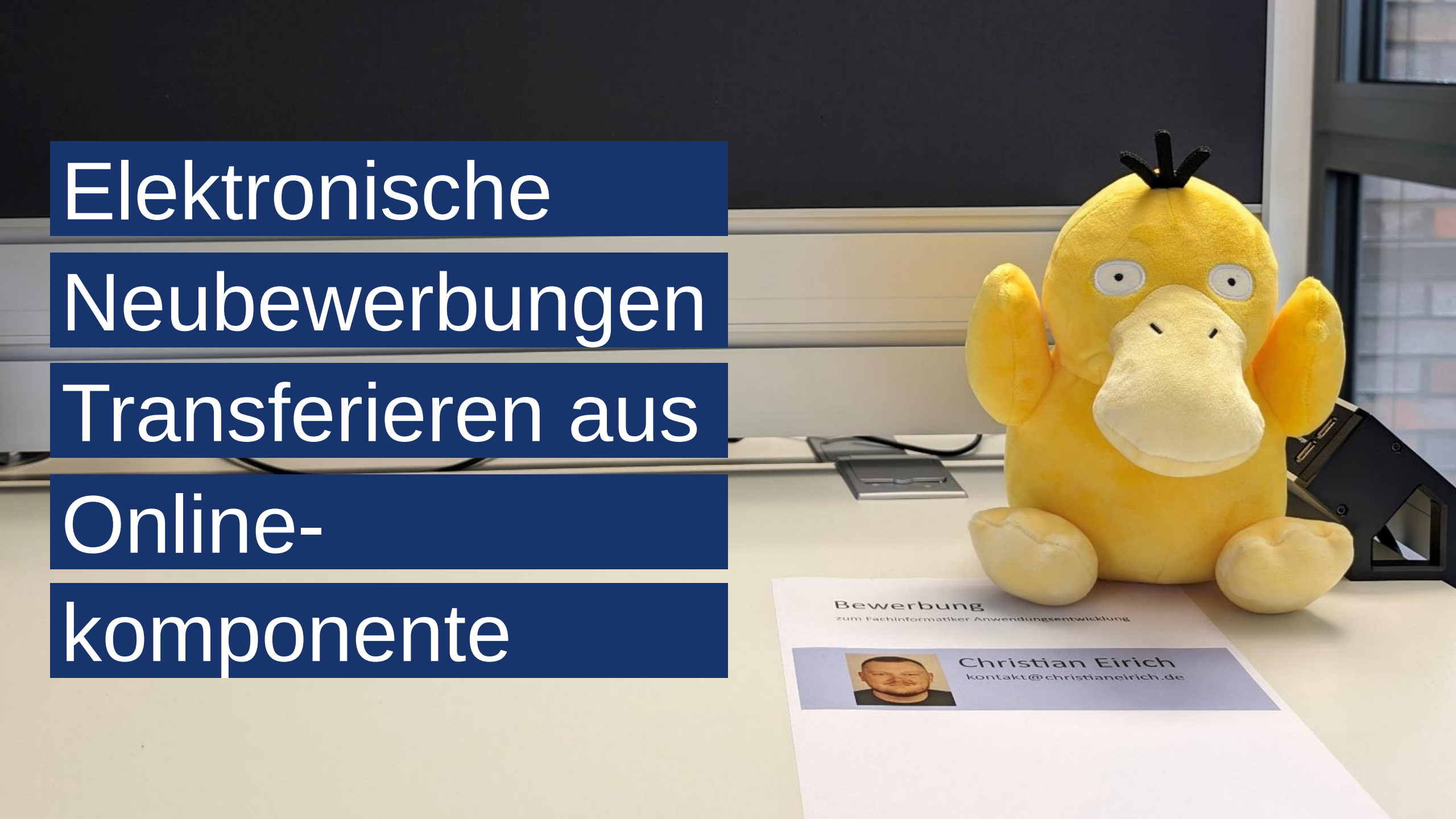

BEWERBUNG
BEWERBUNG
BEWERBUNG
BEWERBUNG
Bewerbung


ROTER
RECHNER
11:19
Dienstag, 9
FUJITSU
AO-MO-806

CHRISTIAN EIRICH

CHRISTIAN EIRICH

Elektronische Neubewerbungen Transferieren aus Online-komponente



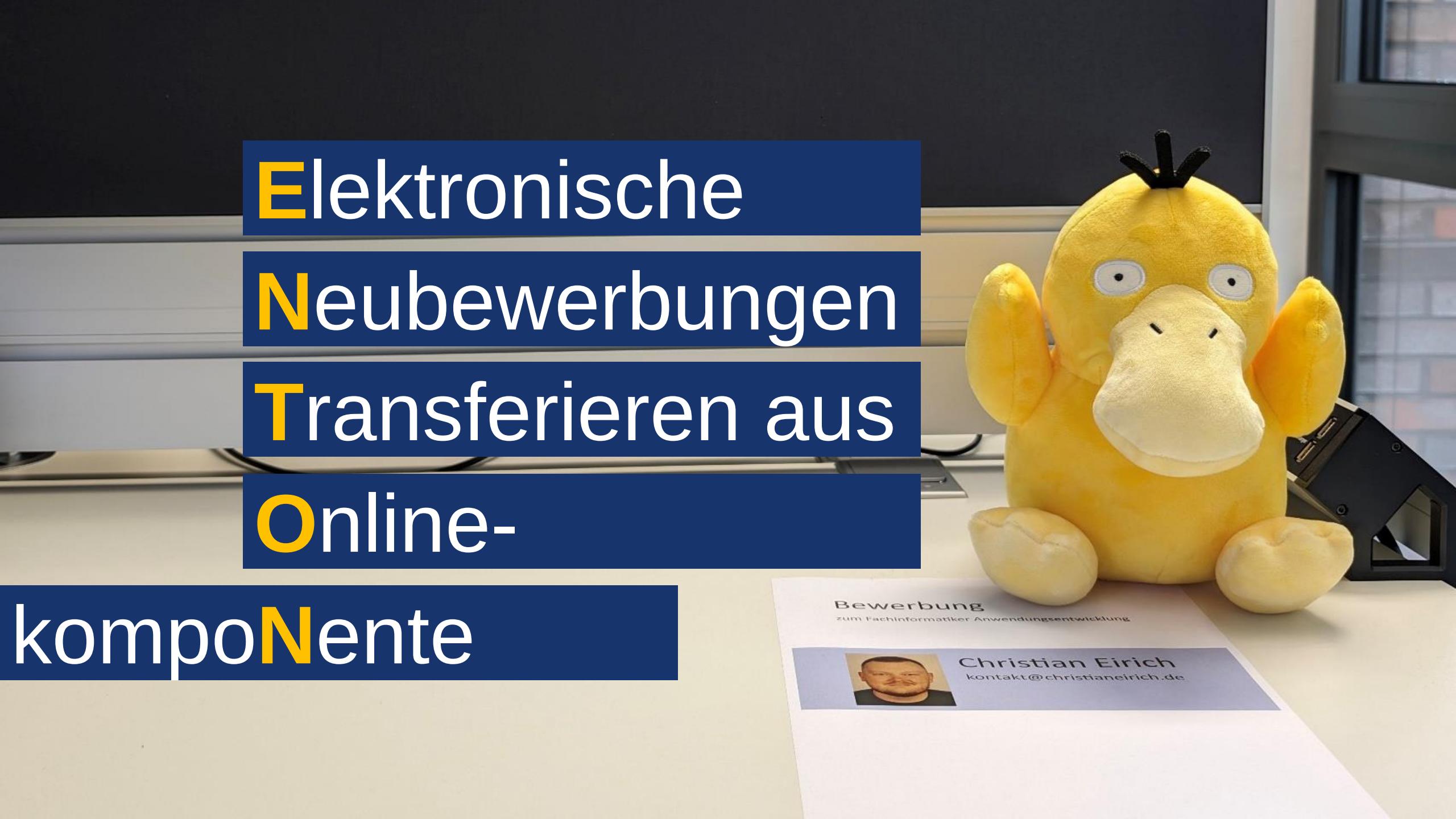
Elektronische

Neubewerbungen

Transferieren aus

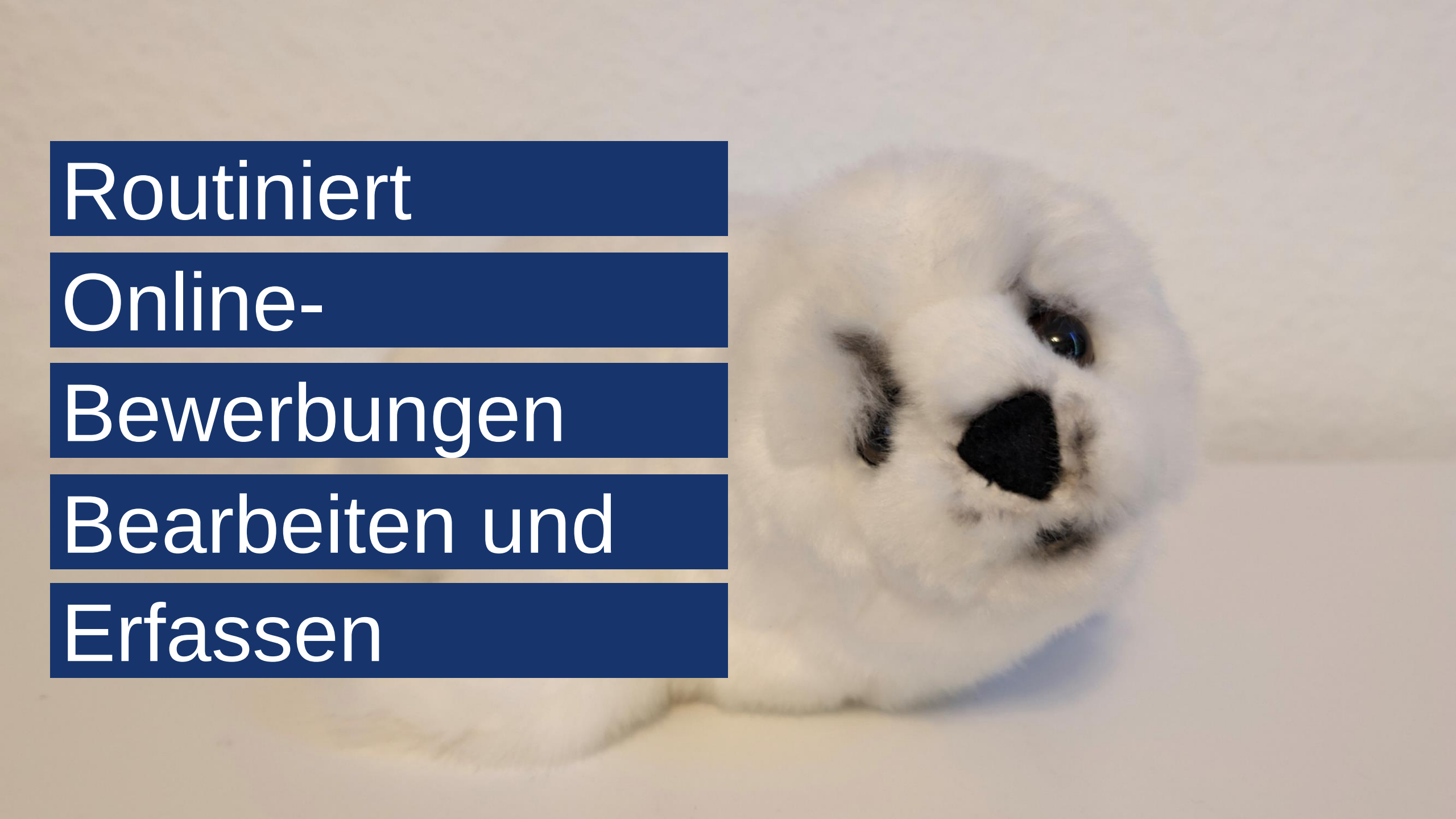
Online-

kompo**N**ente




ALTE OLDENBURGER

Routiniert Online-Bewerbungen Bearbeiten und Erfassen



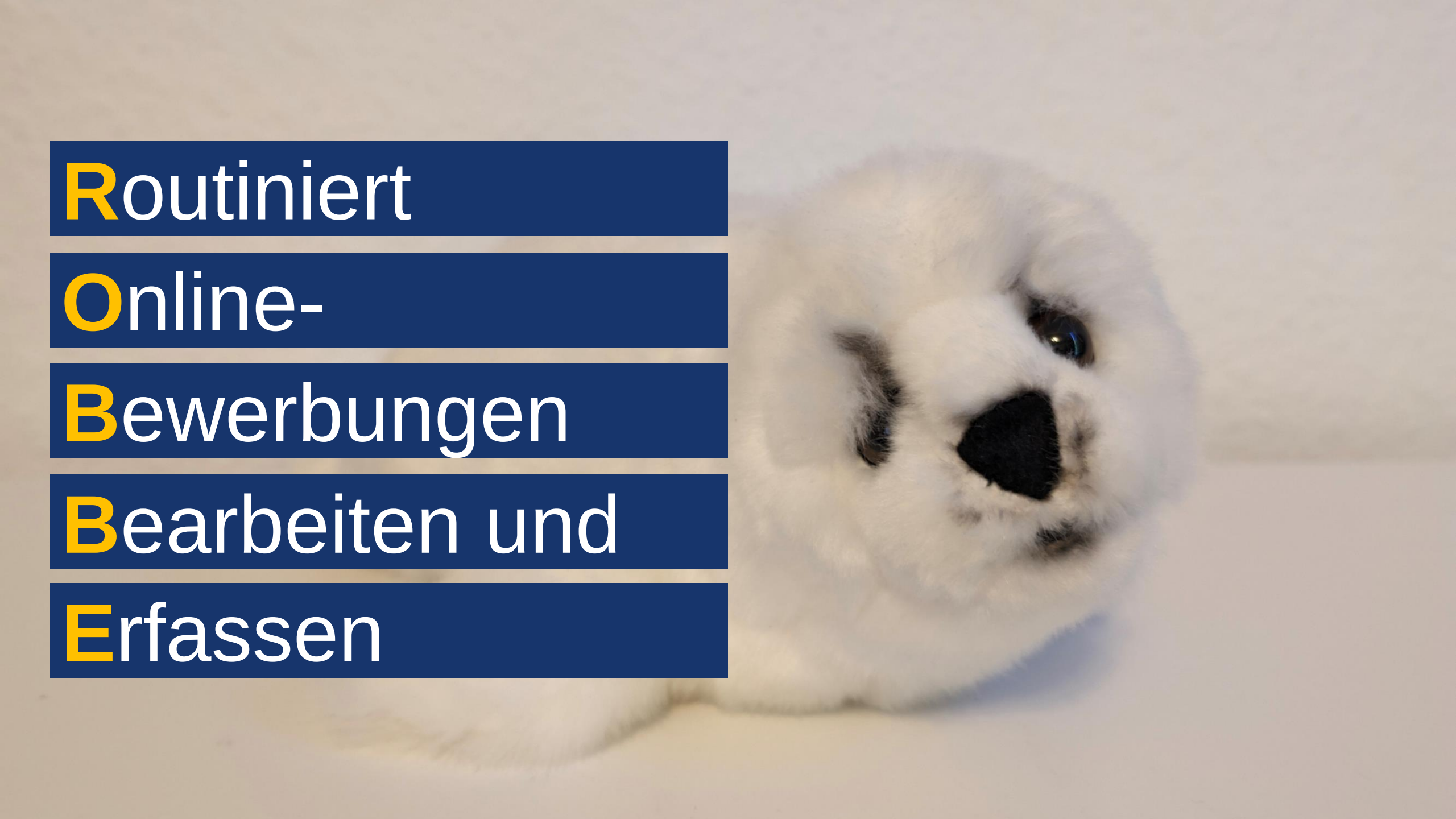
Routiniert

Online-

Bewerbungen

Bearbeiten und

Erfassen



Stellenangebote bei der ALTE OLDENBURGER Krankenversicherung

Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Ausbildungsstart zum 01.08.2024.

Mehr erfahren | Jetzt bewerben

Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Ausbildungsstart zum 01.08.2025.

Mehr erfahren | Jetzt bewerben

Duales Studium Wirtschaftsinformatik 2024 (m/w/d)

Ausbildungsstart zum 01.08.2024.

Mehr erfahren | Jetzt bewerben

Duales Studium Wirtschaftsinformatik 2025 (m/w/d)

Ausbildungsstart zum 01.08.2025.

Mehr erfahren | Jetzt bewerben

Benutzerdokumentation

Laden Sie unsere umfassende Benutzerdokumentation herunter, um mehr über die Funktionalitäten dieser Plattform zu erfahren. Die Dokumentation enthält eine Schritt-für-Schritt-Anleitung für den Bewerbungsprozess. Klicken Sie dazu auf den folgenden Button:

Dokumentation herunterladen

Hilfe und Support

Für Rückfragen oder bei Problemen stehen wir Ihnen gerne zur Verfügung. Sie erreichen uns über die folgenden Kontaktmöglichkeiten:

Email: bewerbung@alte-oldenburger.de

Telefon: 04441 9050


Fazit
Implementierung
Entwurf
Planung
Analyse

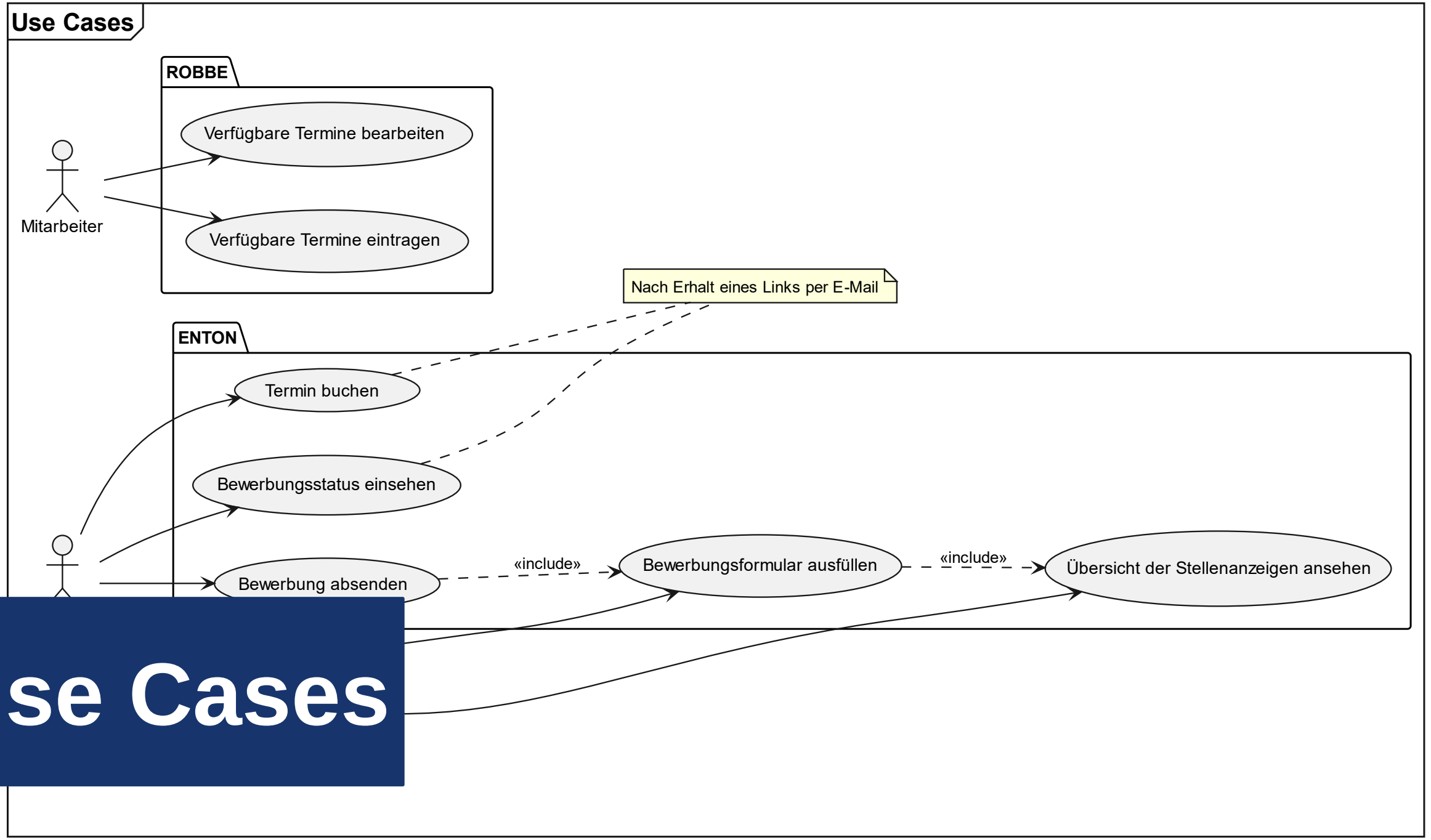
Analyse

Fazit

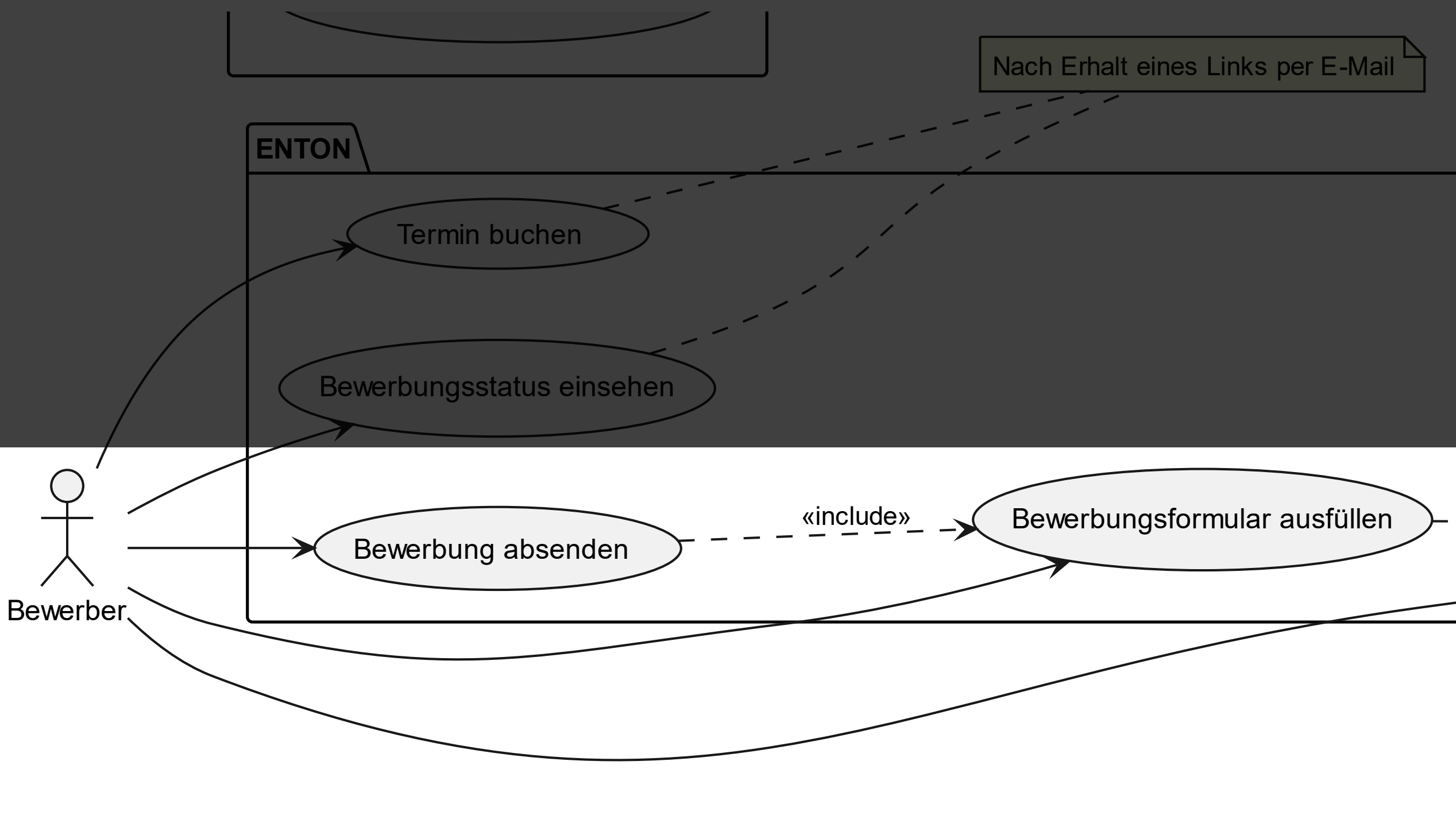
Entw

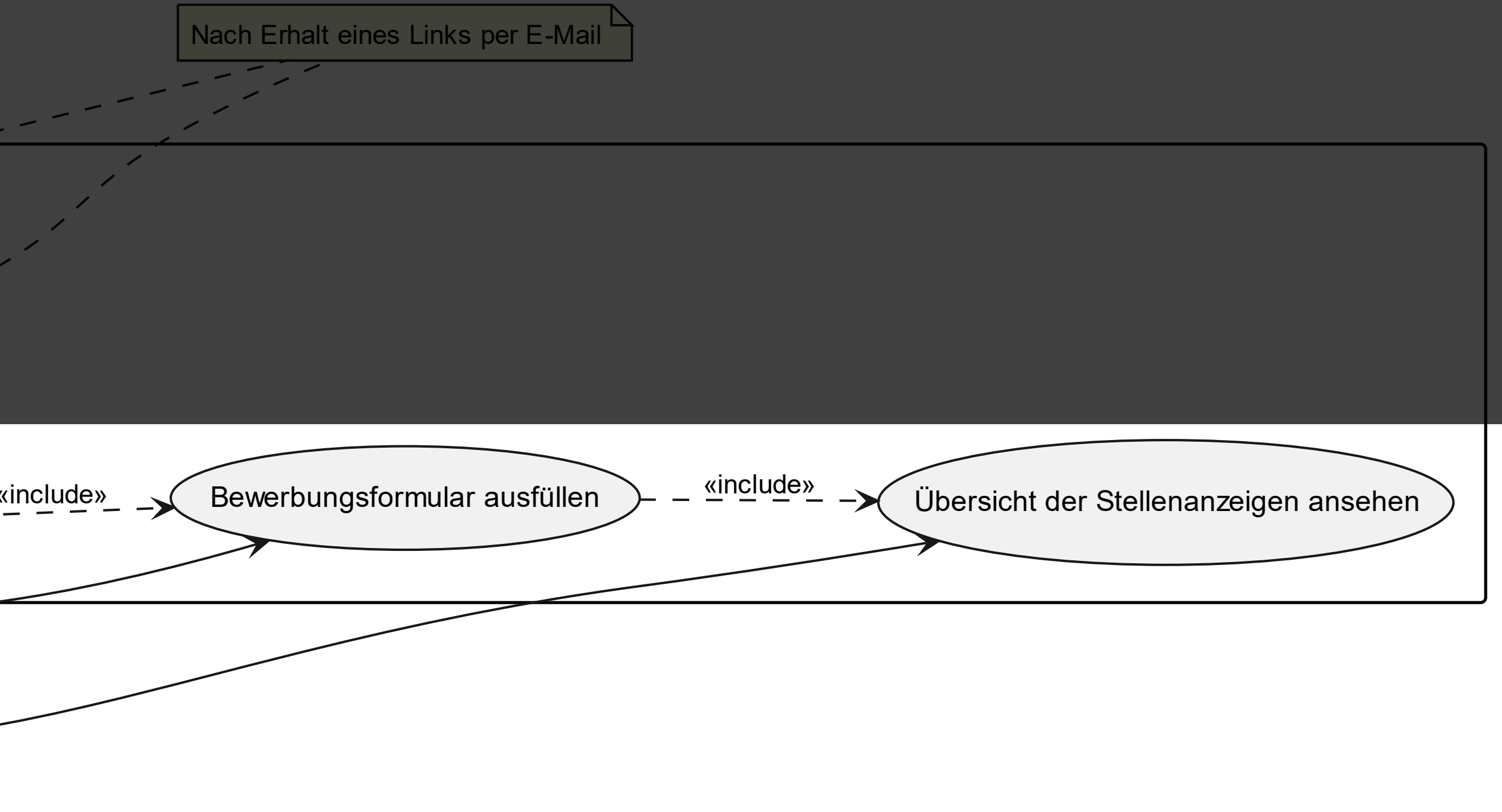
Implementierung

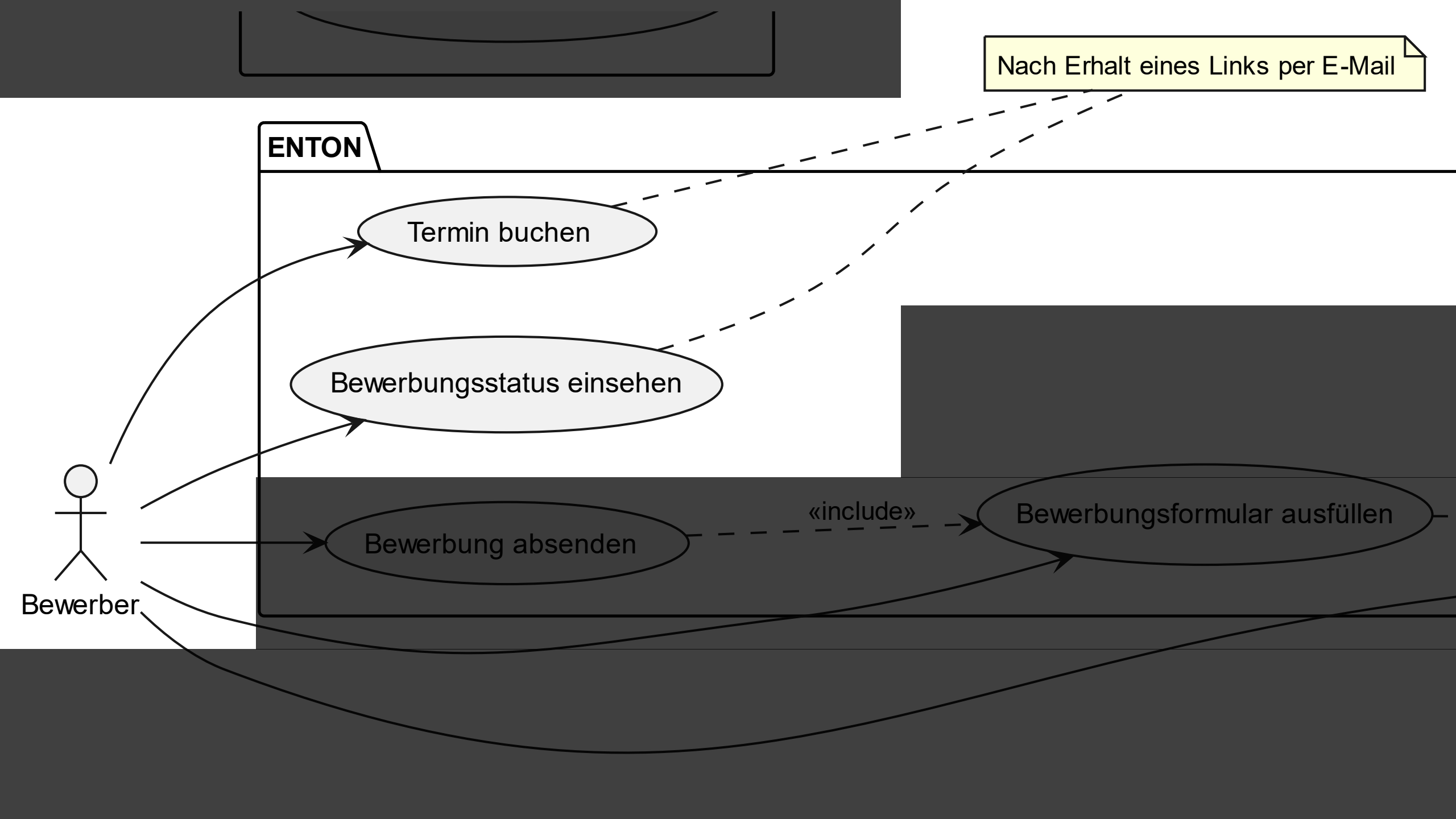
Planung


Use Cases
ROBBE
Verfügbare Termine bearbeiten
Verfügbare Termine eintragen
Mitarbeiter
Nach Erhalt eines Links per E-Mail
ENTON
Termin buchen
Bewerbungsstatus einsehen
Bewerbung absenden
«include»
Bewerbungsformular ausfüllen
«include»
Übersicht der Stellenanzeigen ansehen

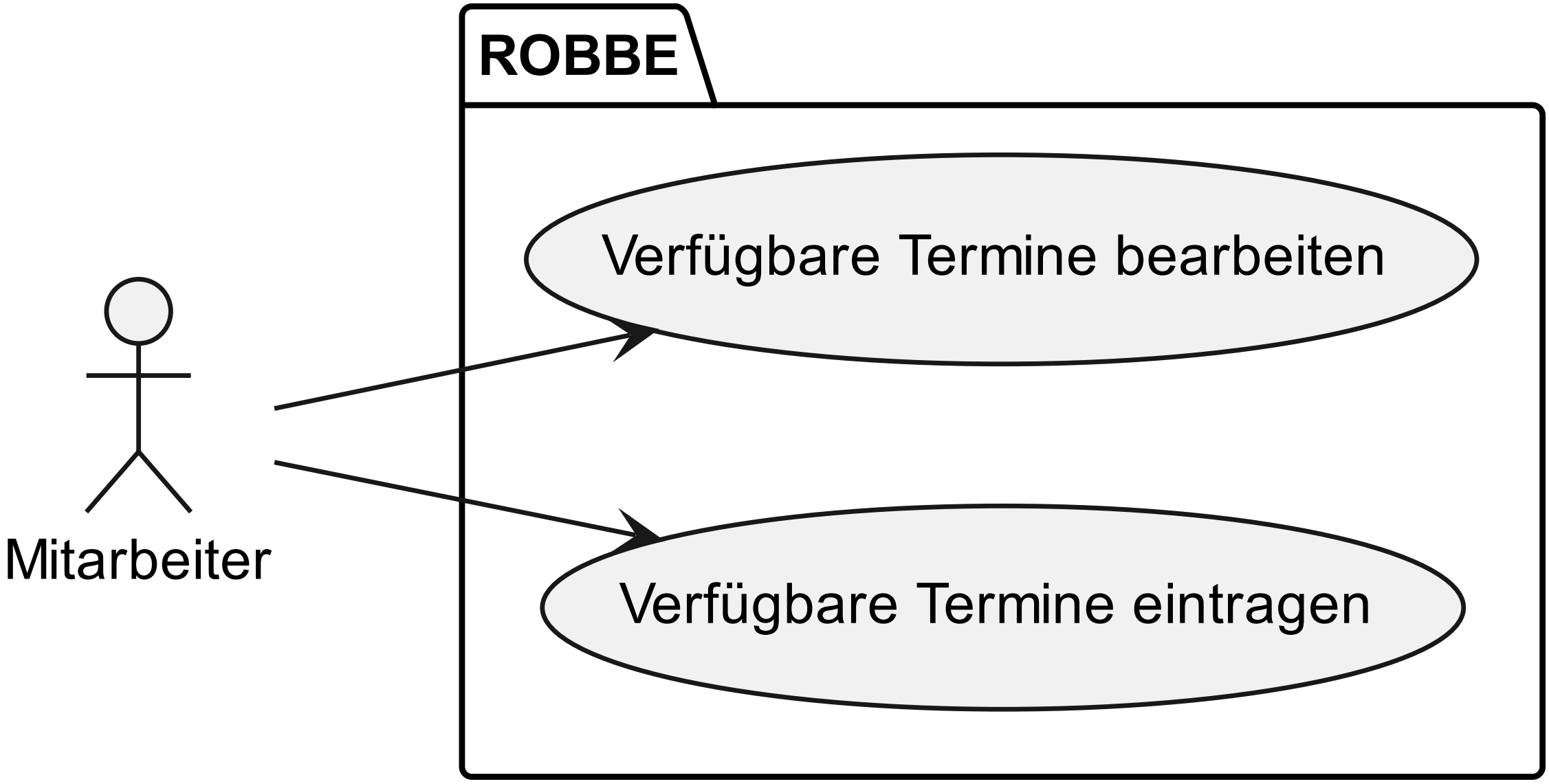
Use Cases


Nach Erhalt eines Links per E-Mail
ENTON
Termin buchen
Bewerbungsstatus einsehen
Bewerber
Bewerbung absenden
«include»
Bewerbungsformular ausfüllen


Nach Erhalt eines Links per E-Mail
«include»
Bewerbungsformular ausfüllen
«include»
Übersicht der Stellenanzeigen ansehen


Nach Erhalt eines Links per E-Mail
ENTON
Termin buchen
Bewerbungsstatus einsehen
Bewerbung absenden
«include»
Bewerbungsformular ausfüllen
Bewerber

Use Cases


ROBBE
Verfügbare Termine bearbeiten
Verfügbare Termine eintragen
Mitarbeiter

Projektkosten

Vorgang	Stunden	Mitarbeiter
Erstellung des Lastenheftes	3 h	5
Abnahme der Mockups	1 h	5
Entwicklungskosten	80 h	1
Abnahme der Pull Requests	3 h	1
Abnahme und Erfolgskontrolle	3 h	5

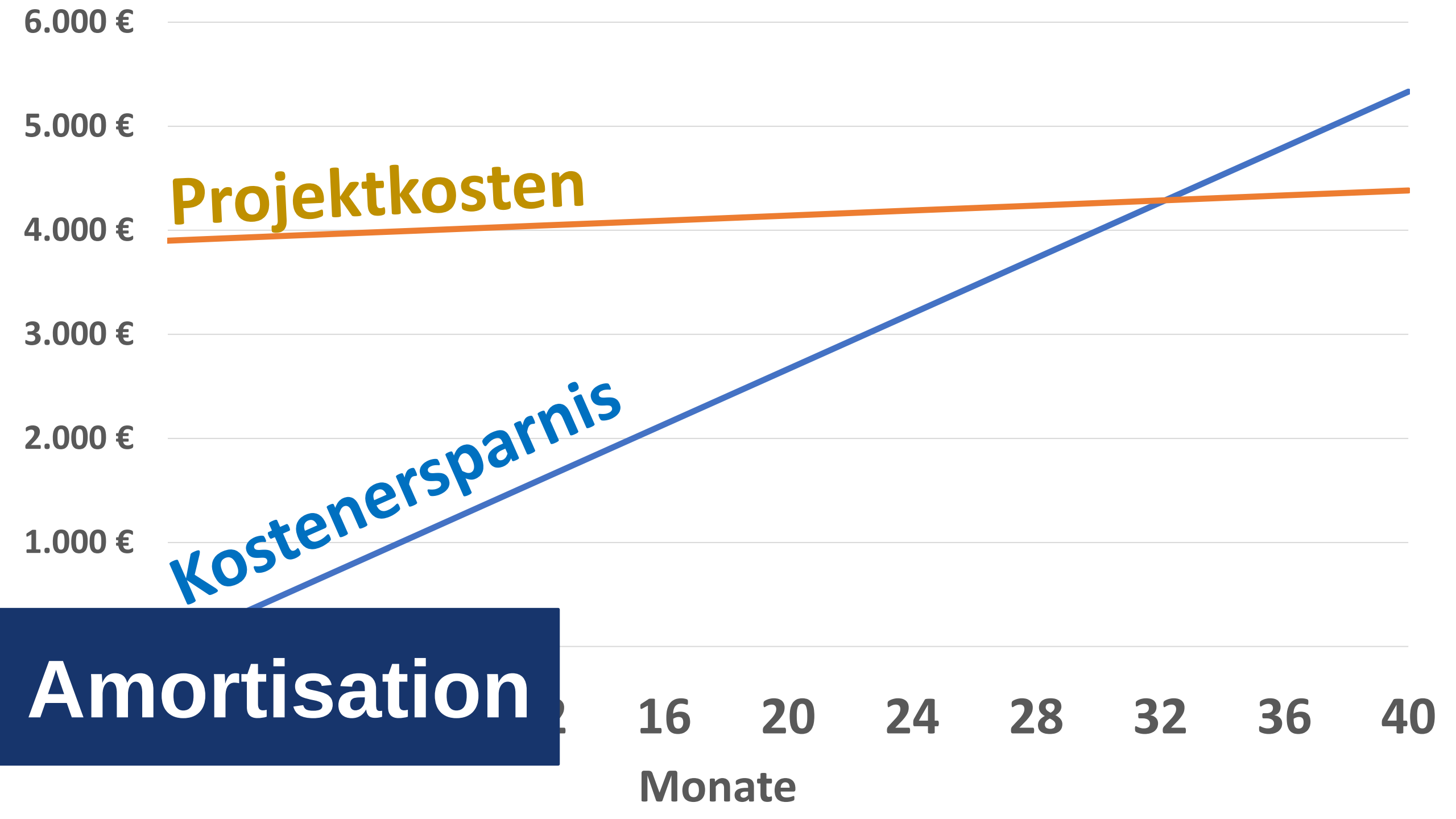
Vorgang	Kosten
Erstellung des Lastenheftes	750,00 €
Abnahme der Mockups	250,00 €
Entwicklungskosten	2.000,00 €
Abnahme der Pull Requests	150,00 €
Abnahme und Erfolgskontrolle	750,00 €
	3.900,00 €

Vorgang	Einsparung je Bewerbung
Download und Virenscan	11 min
Überschneidungen finden	5 min
Endgültige Abstimmung	8 min
	24 min

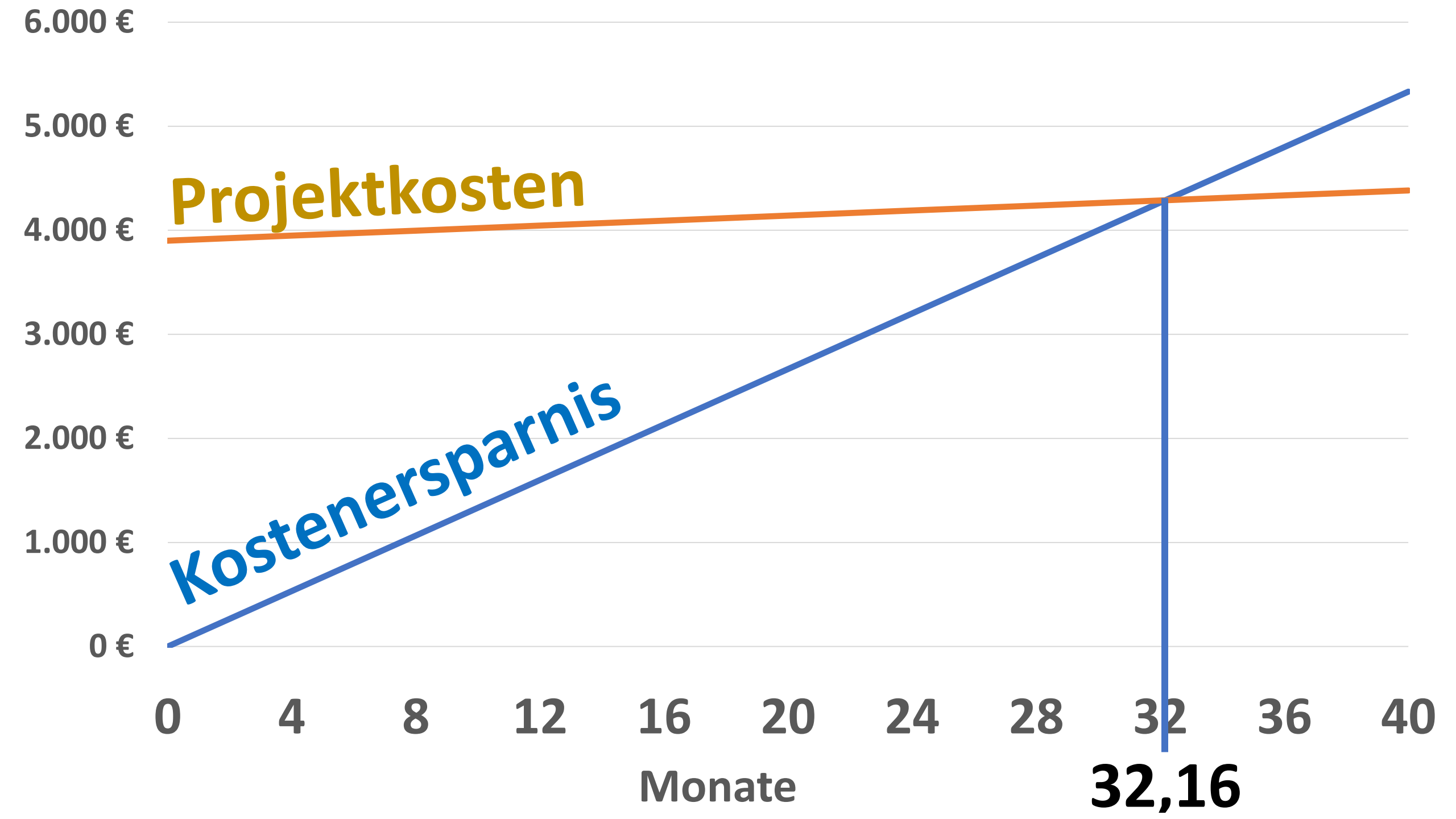
Vorgang	Jährliche Einsparungen
Download und Virenscan	880 min
Überschneidungen finden	400 min
Endgültige Abstimmung	640 min
	1.920 min

Vorgang	Jährliche Einsparungen
Zeiteinsparung	1.600,00 €
Wartung und Betrieb	- 145,00 €
	1.455,00 €

Jährliche Ersparnis



Amortisation


6.000 €
5.000 €
4.000 €
3.000 €
2.000 €
1.000 €
0 €
Projektkosten
Kostenersparnis
0
4
8
12
16
20
24
28
32
36
40
Monate
32,16

LOHNT SICH


Fazit
Entw
Implementierung
Planung
Analyse

Planung

Fazit

Implementierung

Entwurf

Analyse

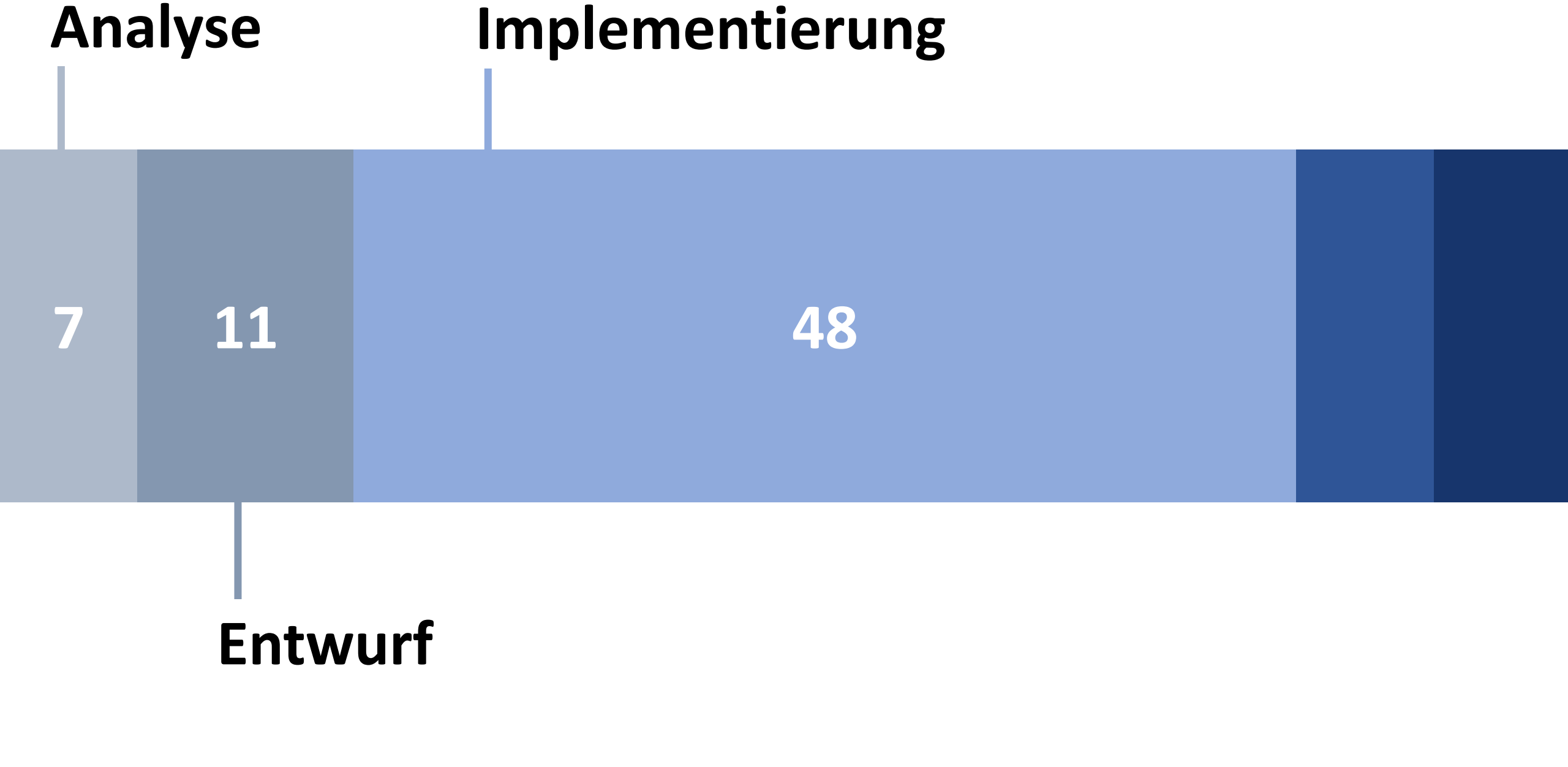
Vorgehensmodell

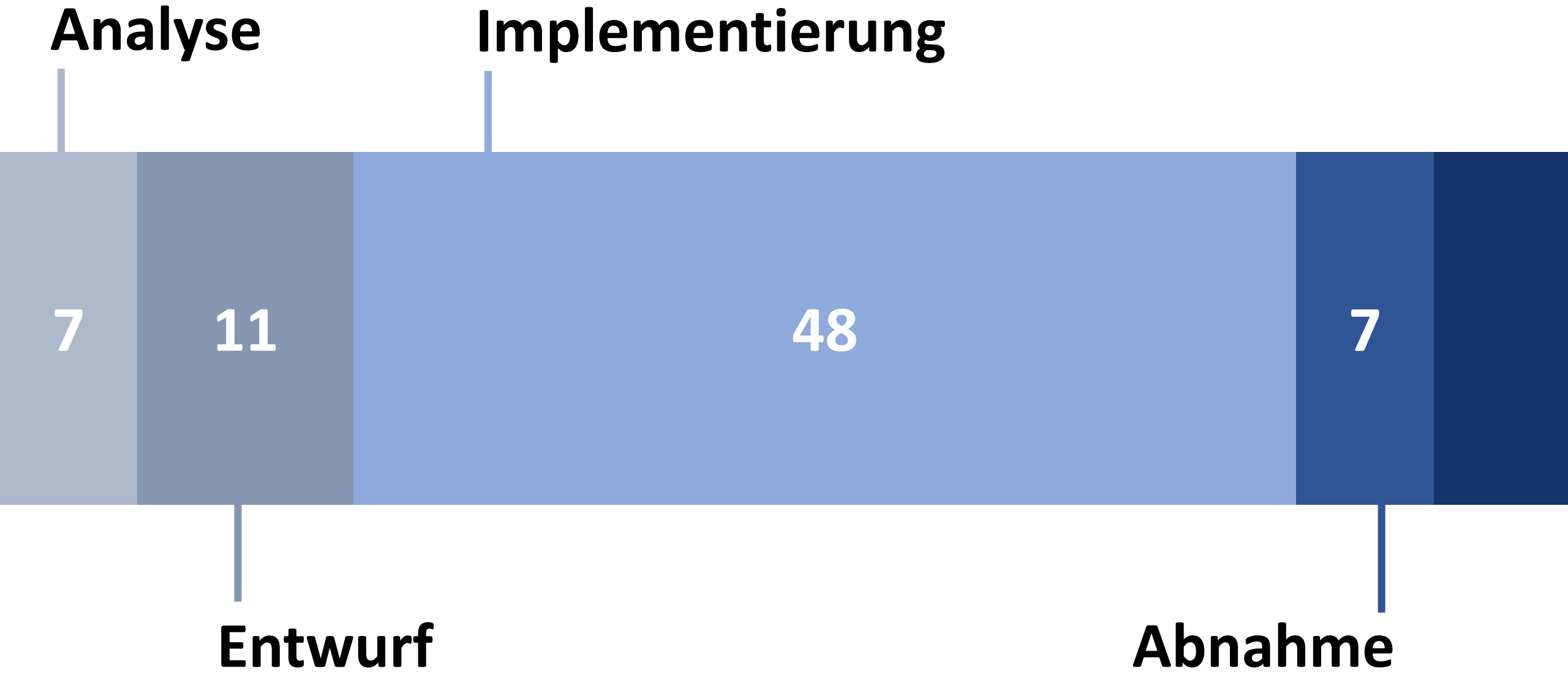
Projektphasen

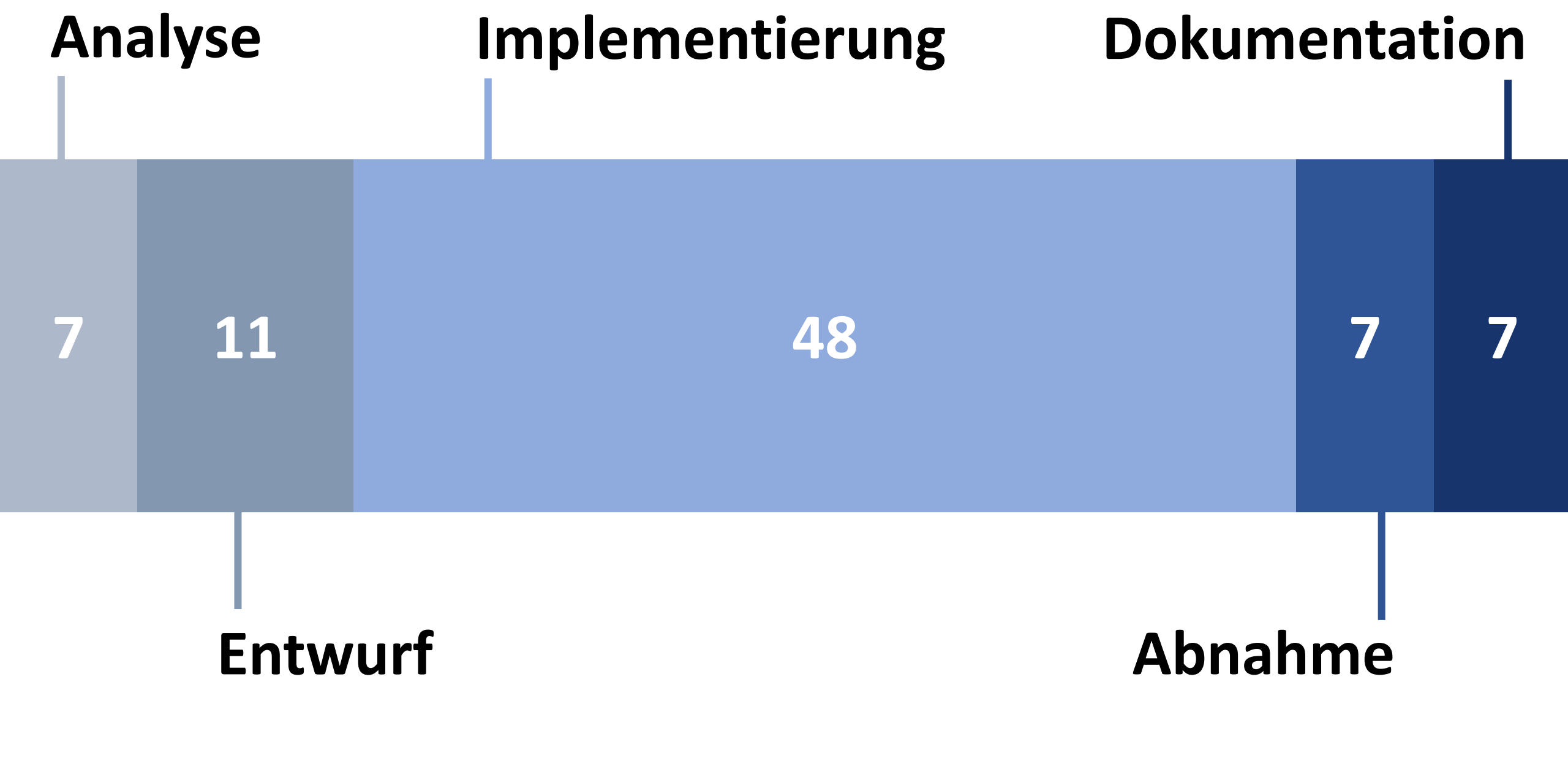
Analyse

7


Analyse
7
11
Entwurf


Analyse
Implementierung
7
11
48
Entwurf


Analyse
Implementierung
7
11
48
7
Entwurf
Abnahme


Analyse
Implementierung
Dokumentation
7
11
48
7
7
Entwurf
Abnahme


Fazit
Implementierung
Entwurf
Planung
Analyse

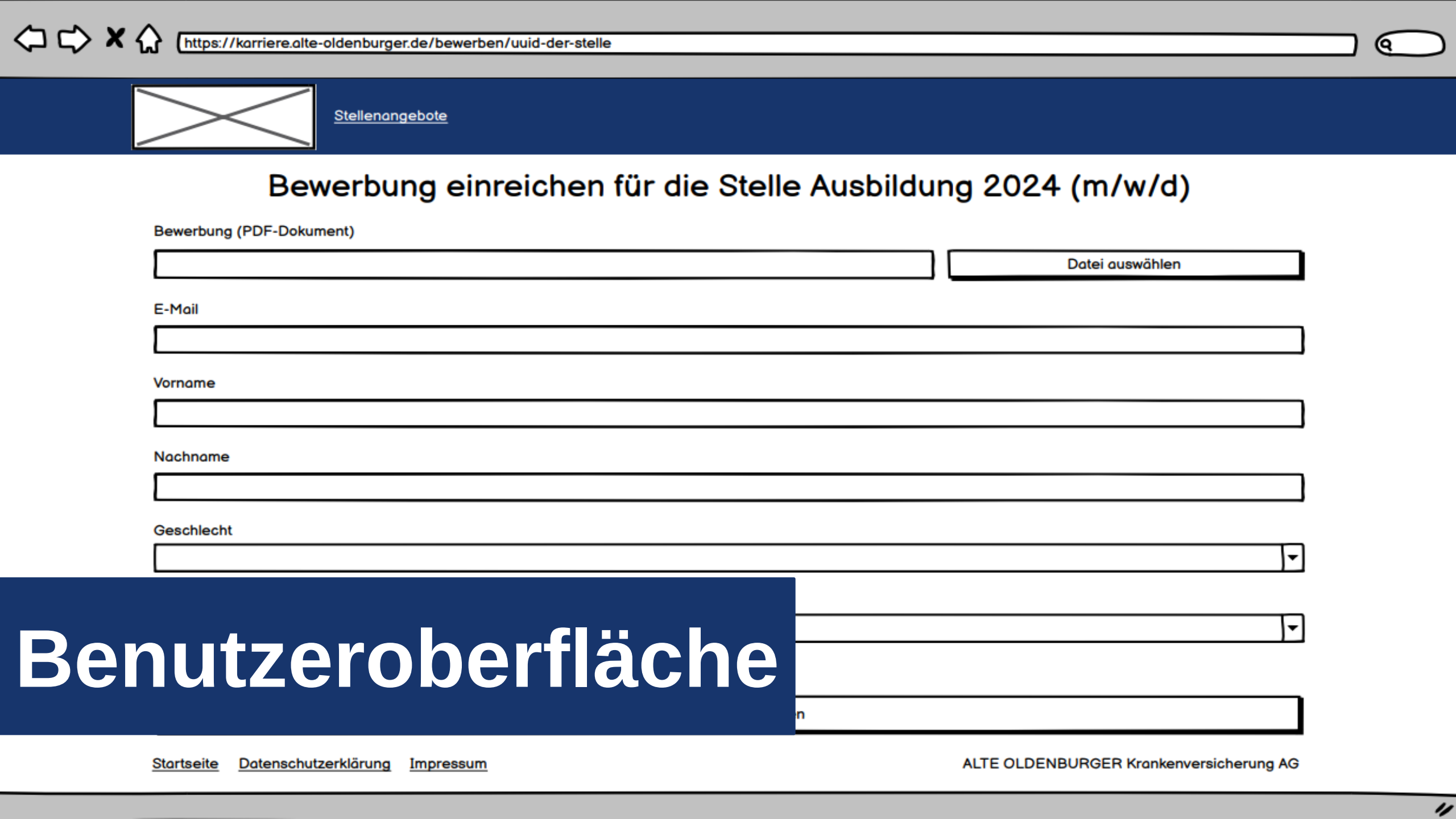
Entwurf

Faz

Implementierung

Planung

Analyse


https://karriere.alte-oldenburger.de/bewerben/uuid-der-stelle
Stellenangebote
Bewerbung einreichen für die Stelle Ausbildung 2024 (m/w/d)
Bewerbung (PDF-Dokument)
Datei auswählen
E-Mail
Vorname
Nachname
Geschlecht
Benutzeroberfläche
Startseite
Datenschutzerklärung
Impressum
ALTE OLDENBURGER Krankenversicherung AG

https://karriere.alte-oldenburger.de/bewerben/uuid-der-stelle

Stellenangebote

Bewerbung einreichen für die Stelle Ausbildung 2024 (m/w/d)

Pflichtfelder

Bewerbung (PDF-Dokument) *

Datei auswählen

Freiweillige Angaben

E-Mail

Vorname

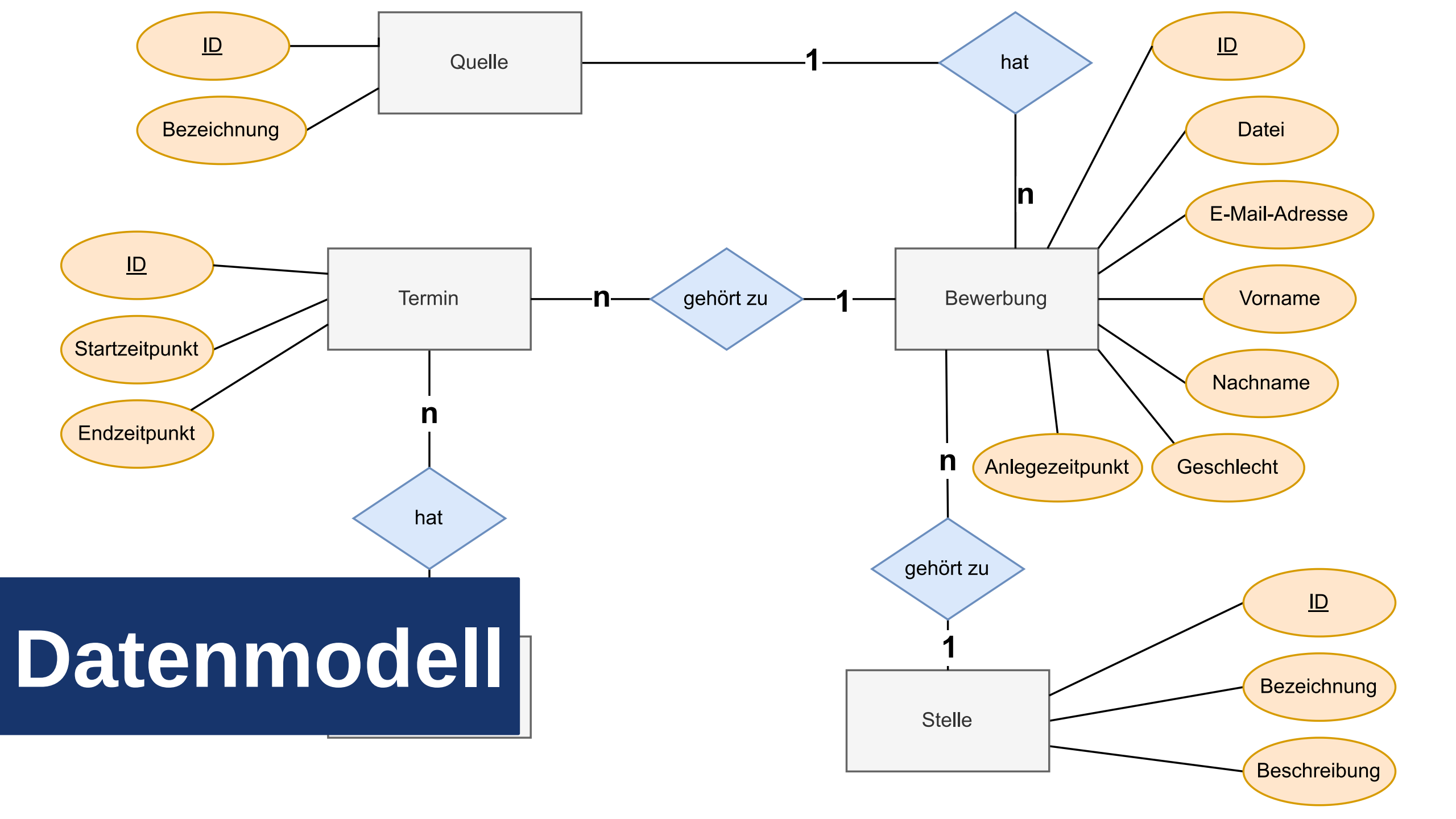
Nachname

Geschlecht

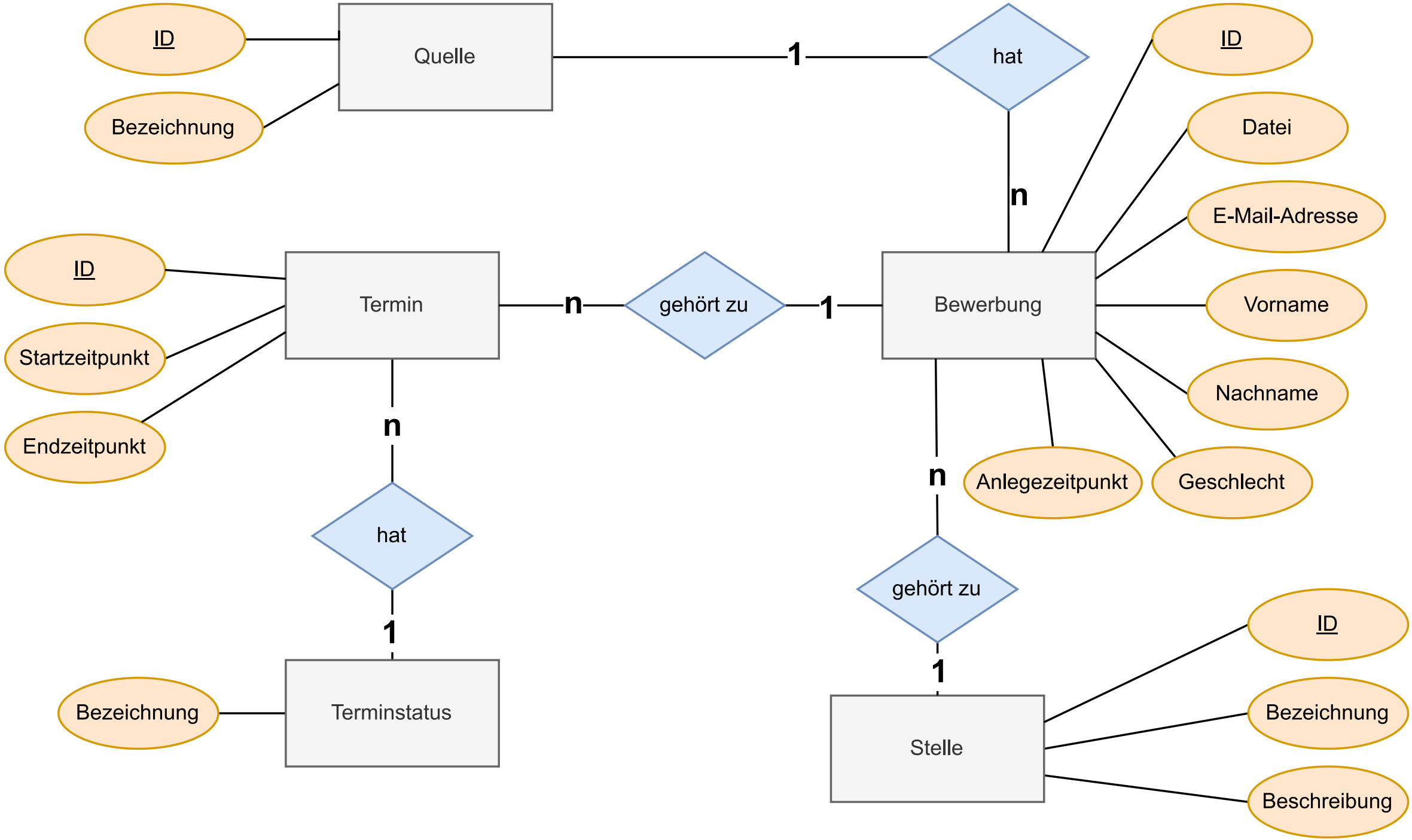
Wie sind Sie auf uns aufmerksam geworden?

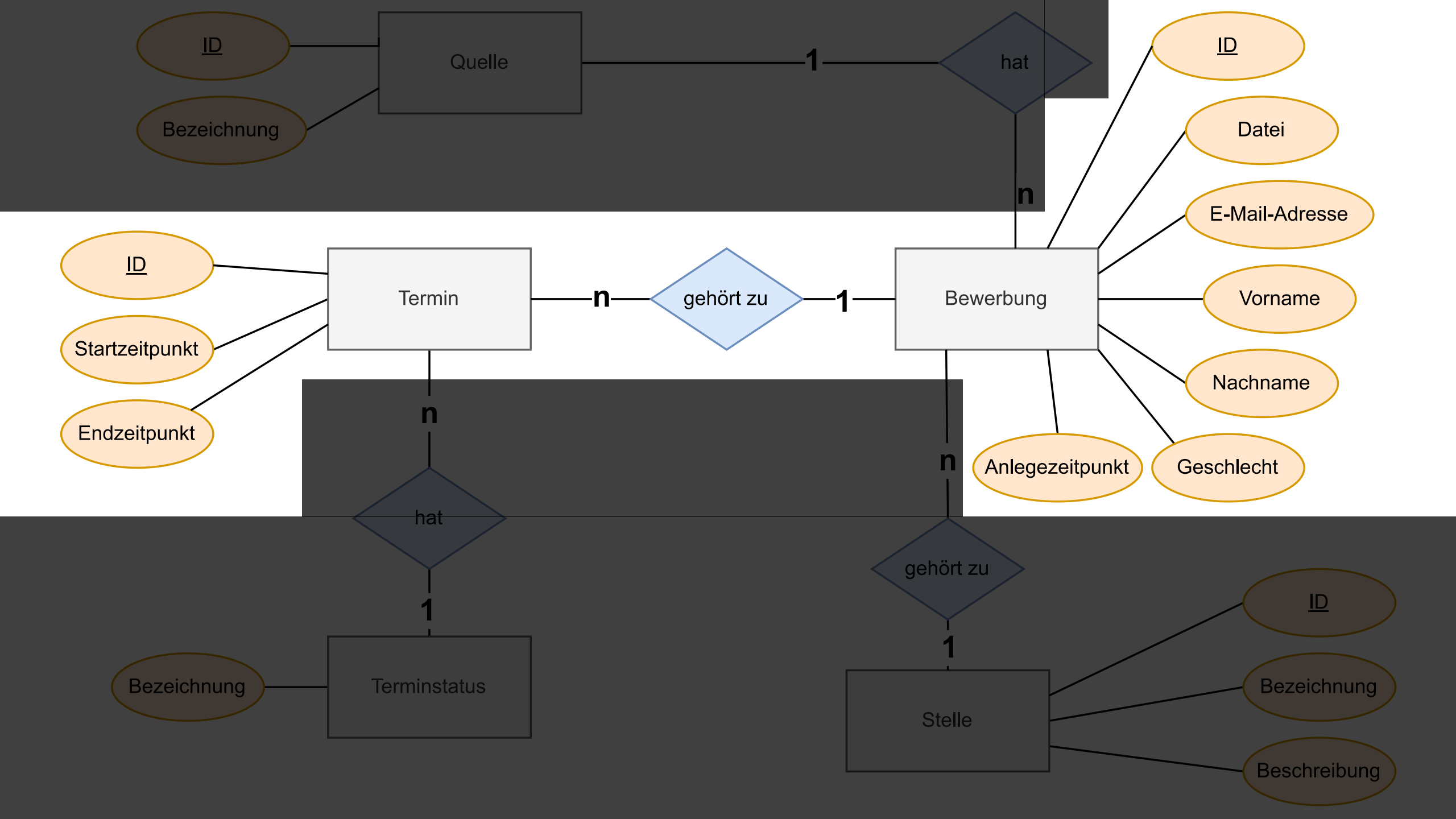
☐ Ich habe die Datenschutzerklärung gelesen und zur Kenntnis genommen.

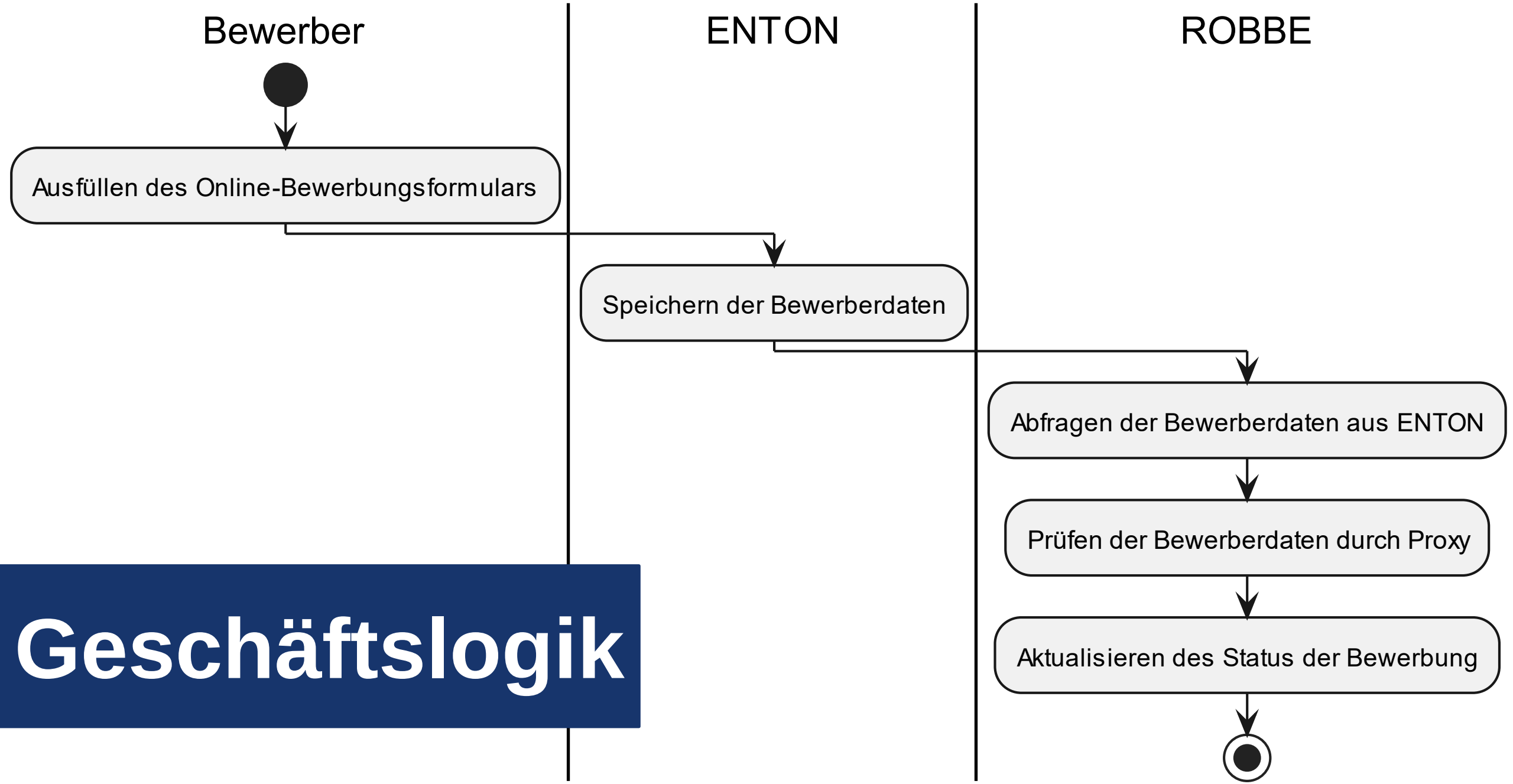
Bewerbung einreichen



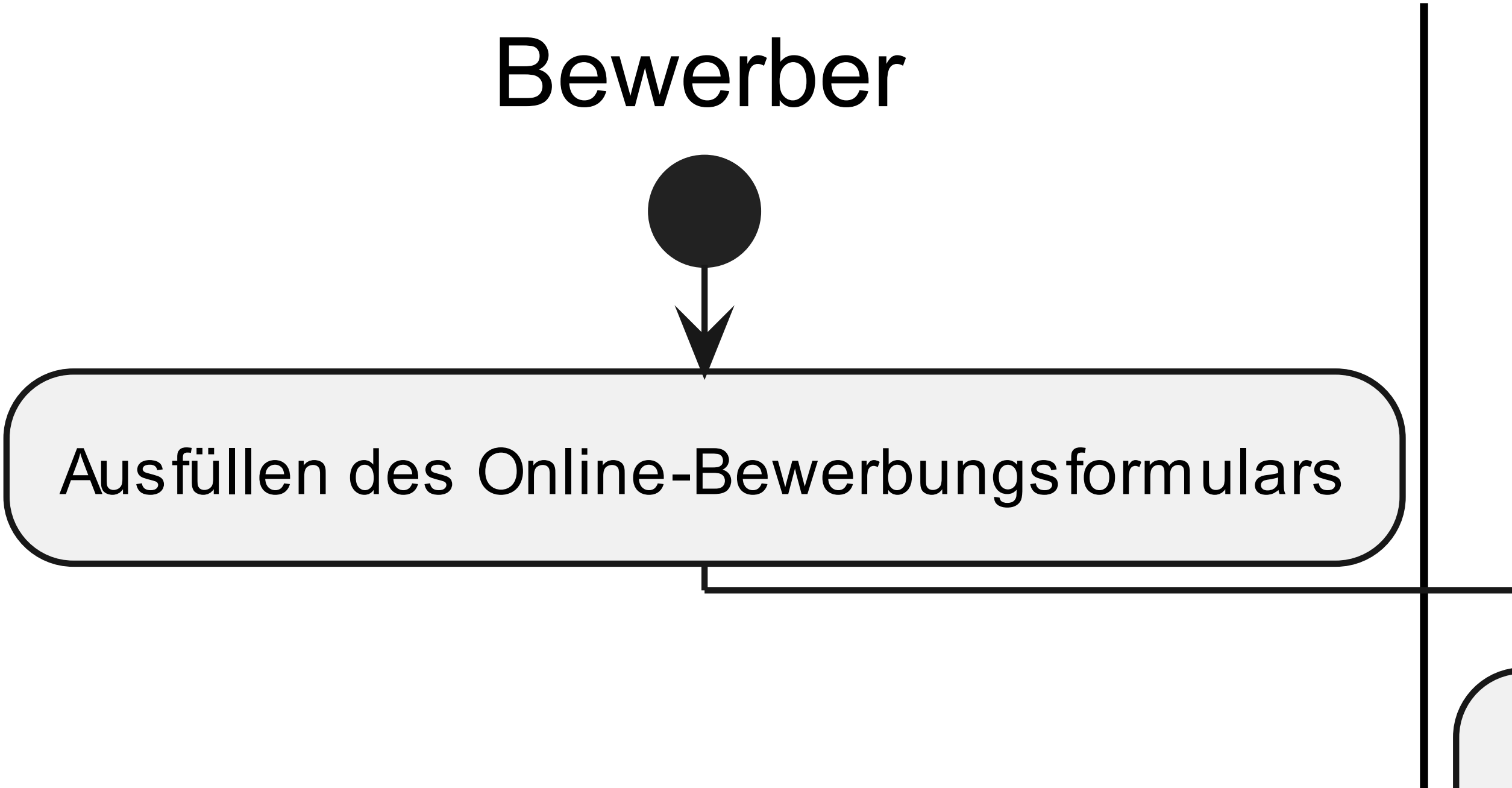
Datenmodell

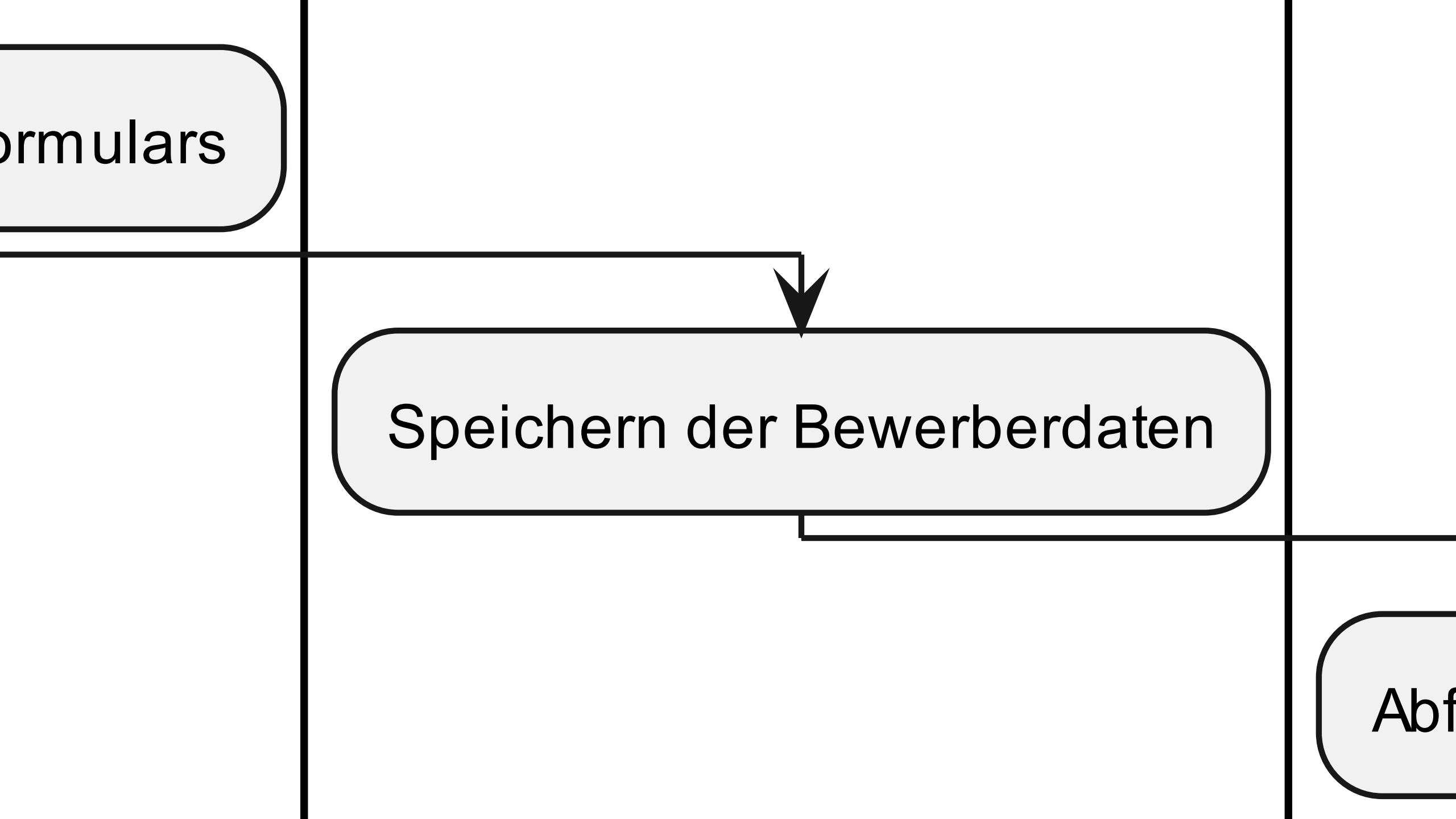

ID
Quelle
Bezeichnung
1
hat
n
ID
Datei
E-Mail-Adresse
Vorname
Nachname
Anlegezeitpunkt
Geschlecht
Bewerbung
ID
Termin
Startzeitpunkt
Endzeitpunkt
n
gehört zu
1
n
hat
1
Bezeichnung
Terminstatus
n
gehört zu
1
Stelle
ID
Bezeichnung
Beschreibung

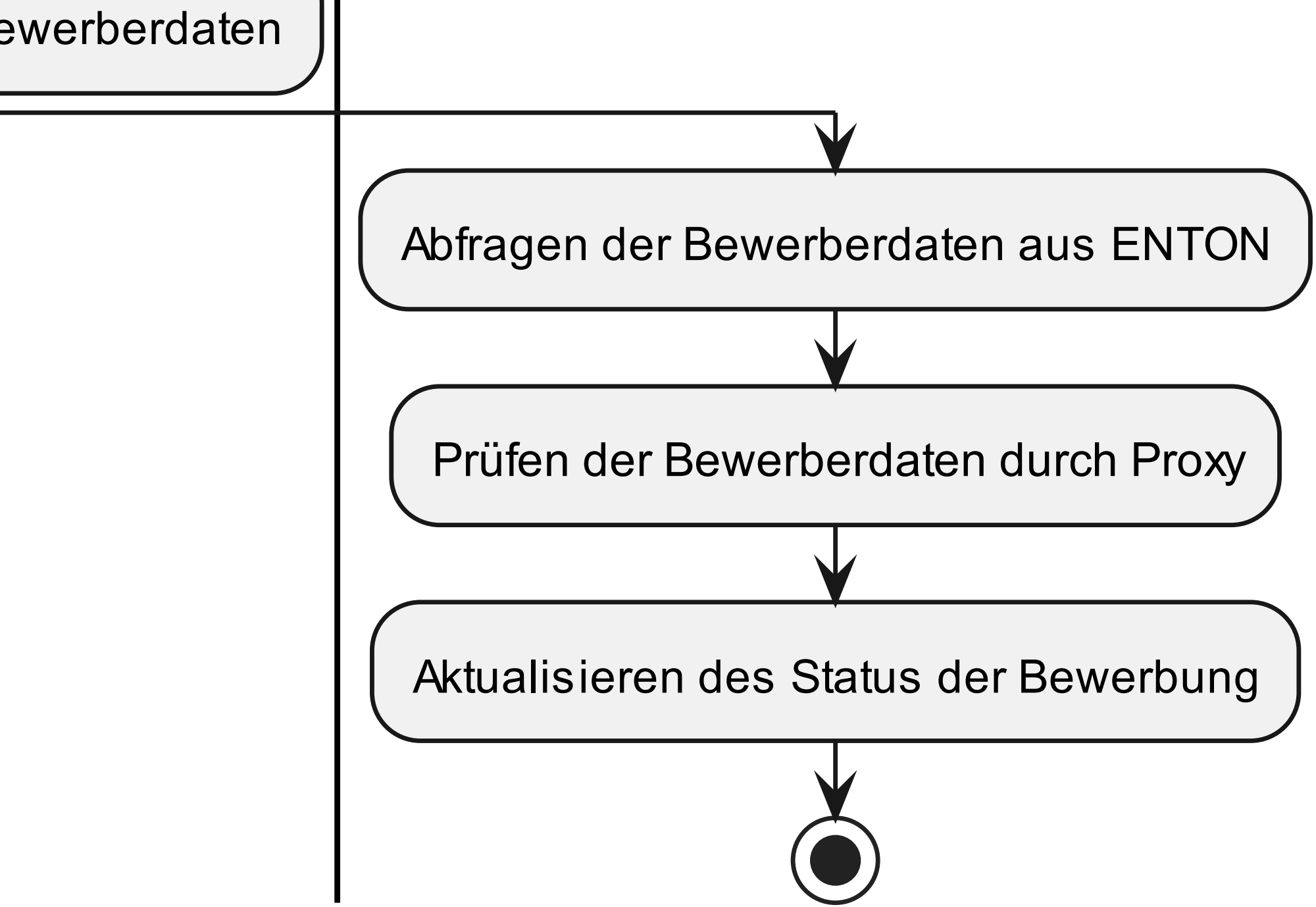

ID
Quelle
Bezeichnung
1
hat
n
ID
Datei
E-Mail-Adresse
Vorname
Nachname
Anlegezeitpunkt
Geschlecht
ID
Termin
Startzeitpunkt
Endzeitpunkt
n
gehört zu
1
Bewerbung
n
hat
1
Bezeichnung
Terminstatus
gehört zu
n
1
Stelle
ID
Bezeichnung
Beschreibung

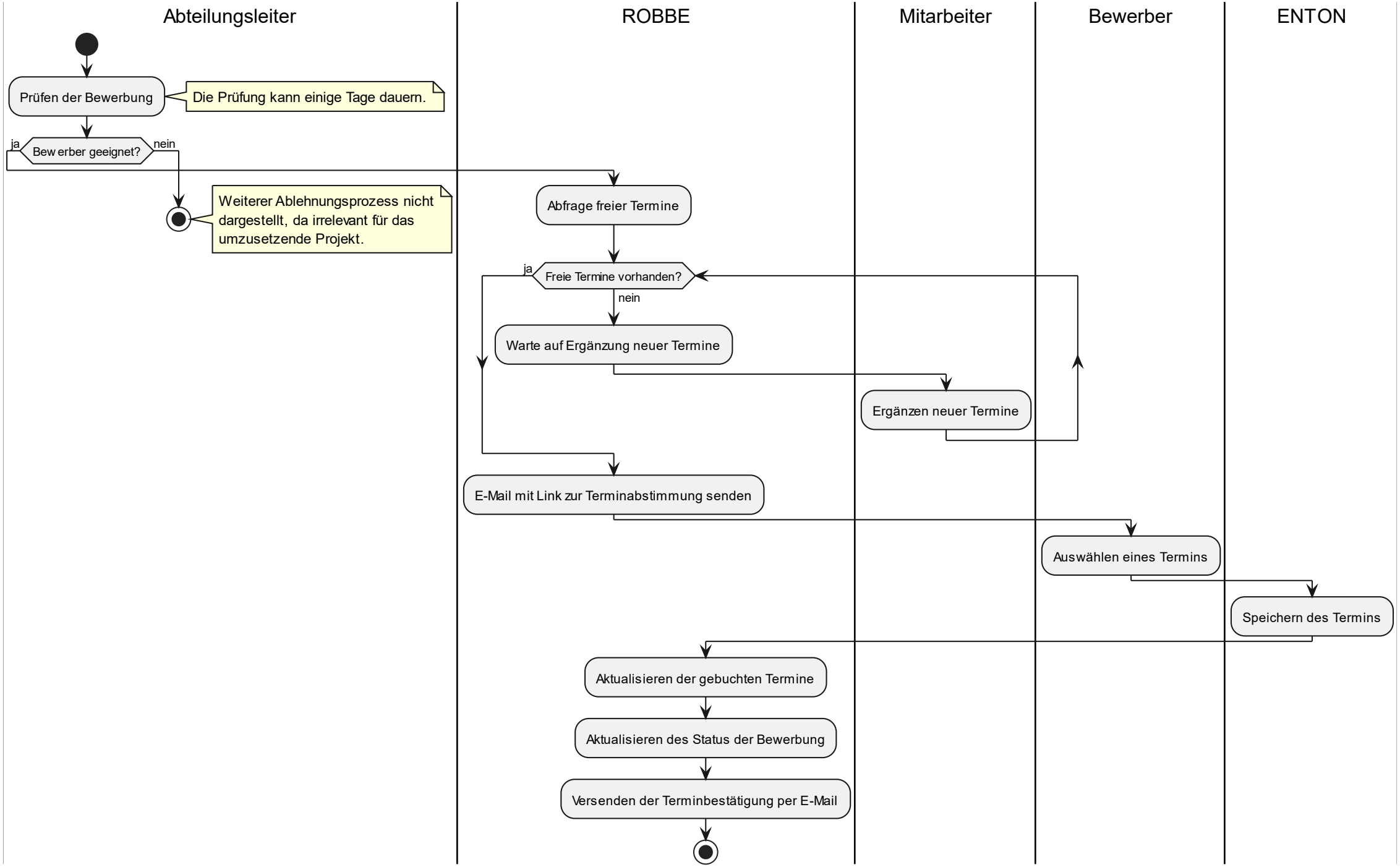


Geschäftslogik

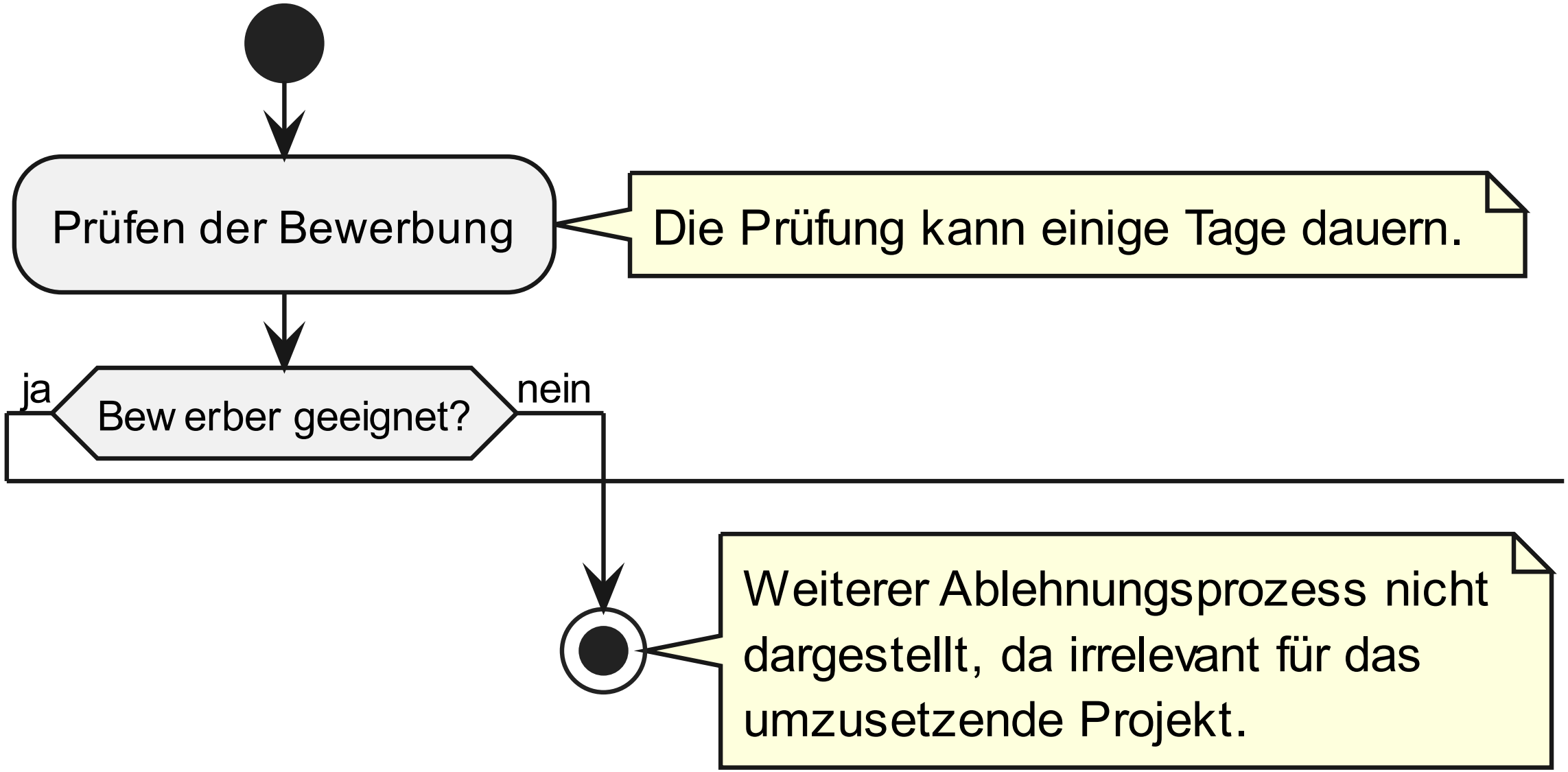

Bewerber
Ausfüllen des Online-Bewerbungsformulars

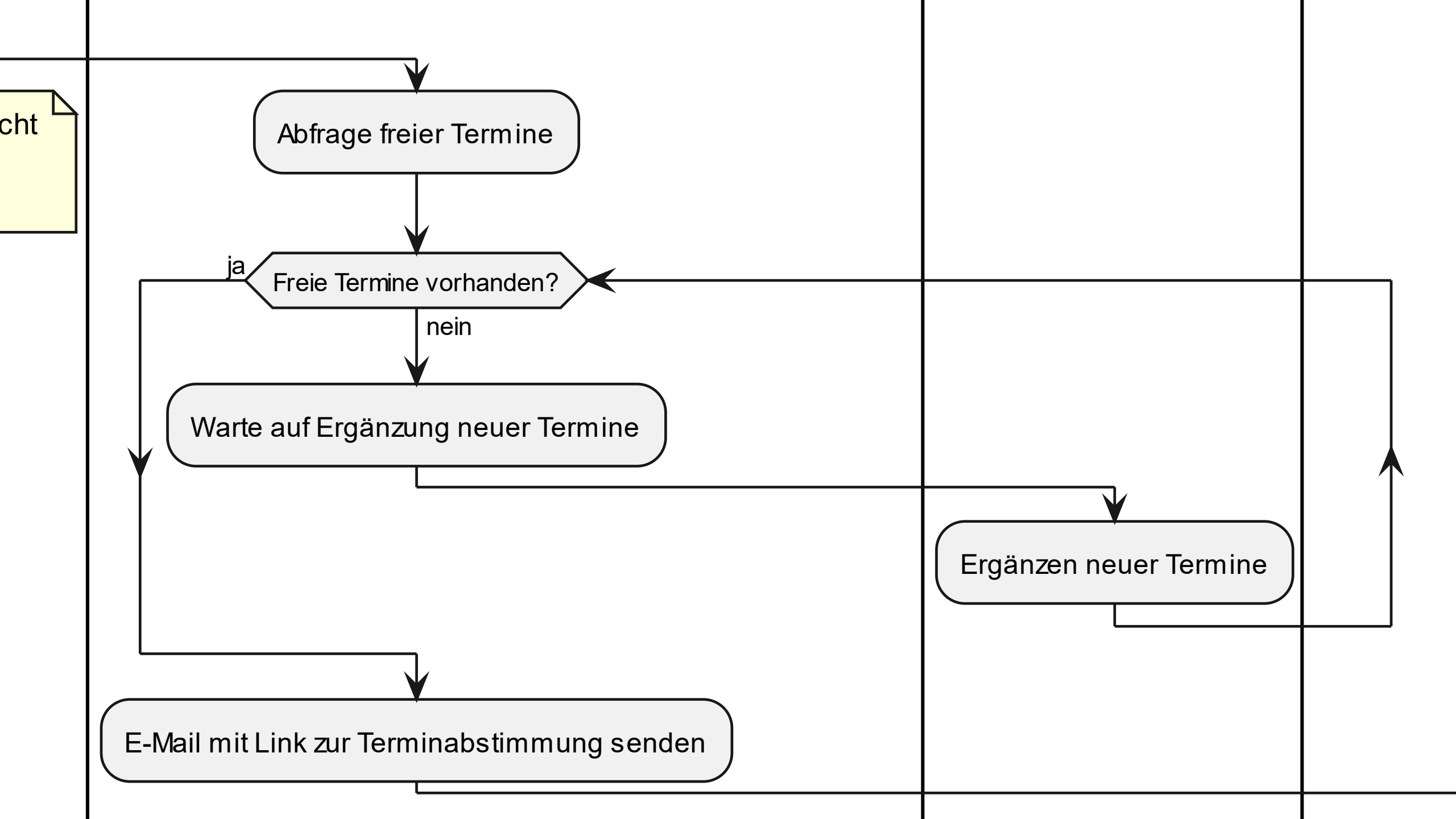

ormulars
Speichern der Bewerberdaten
Abf

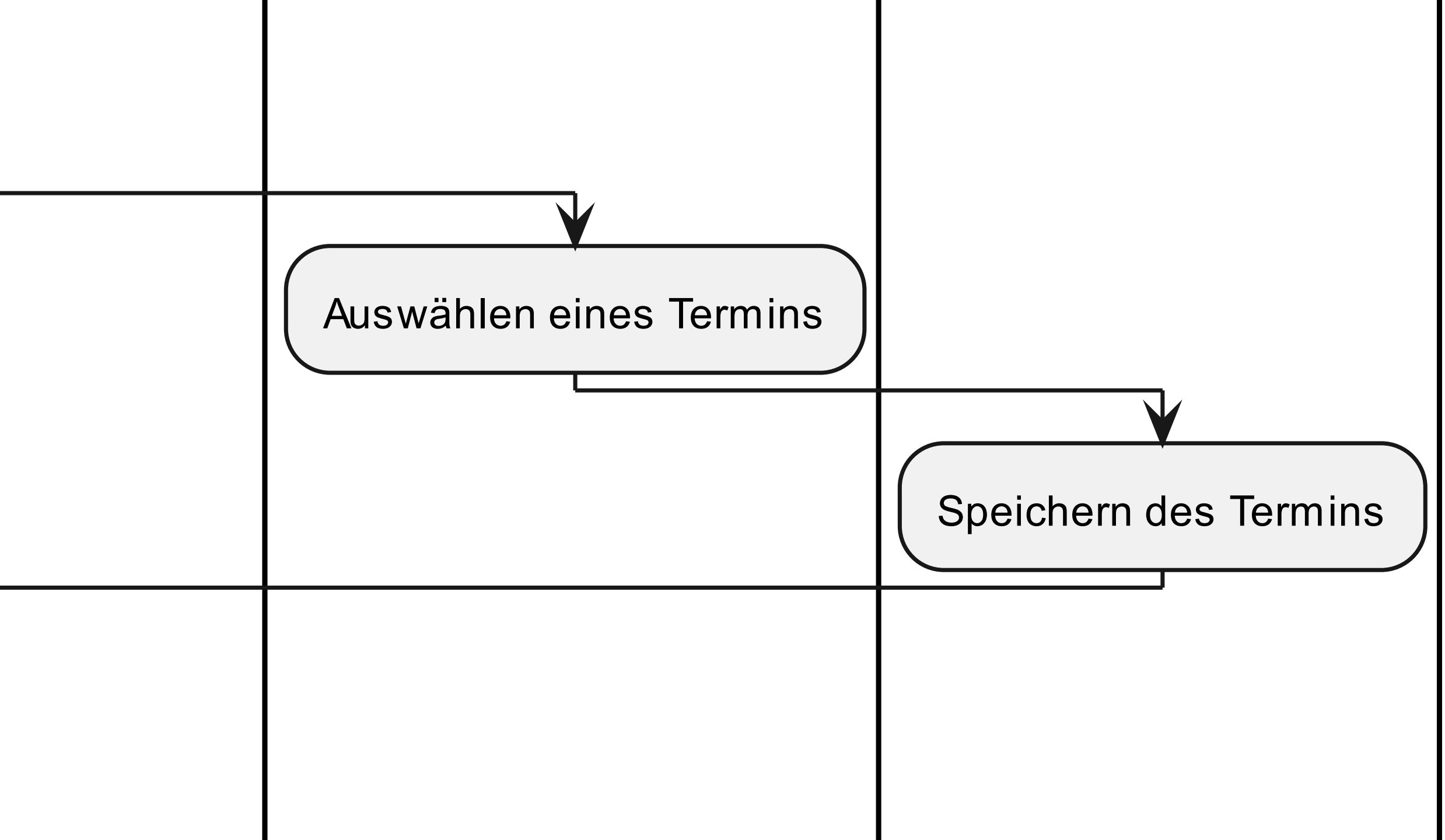

ewerberdaten
Abfragen der Bewerberdaten aus ENTON
Prüfen der Bewerberdaten durch Proxy
Aktualisieren des Status der Bewerbung

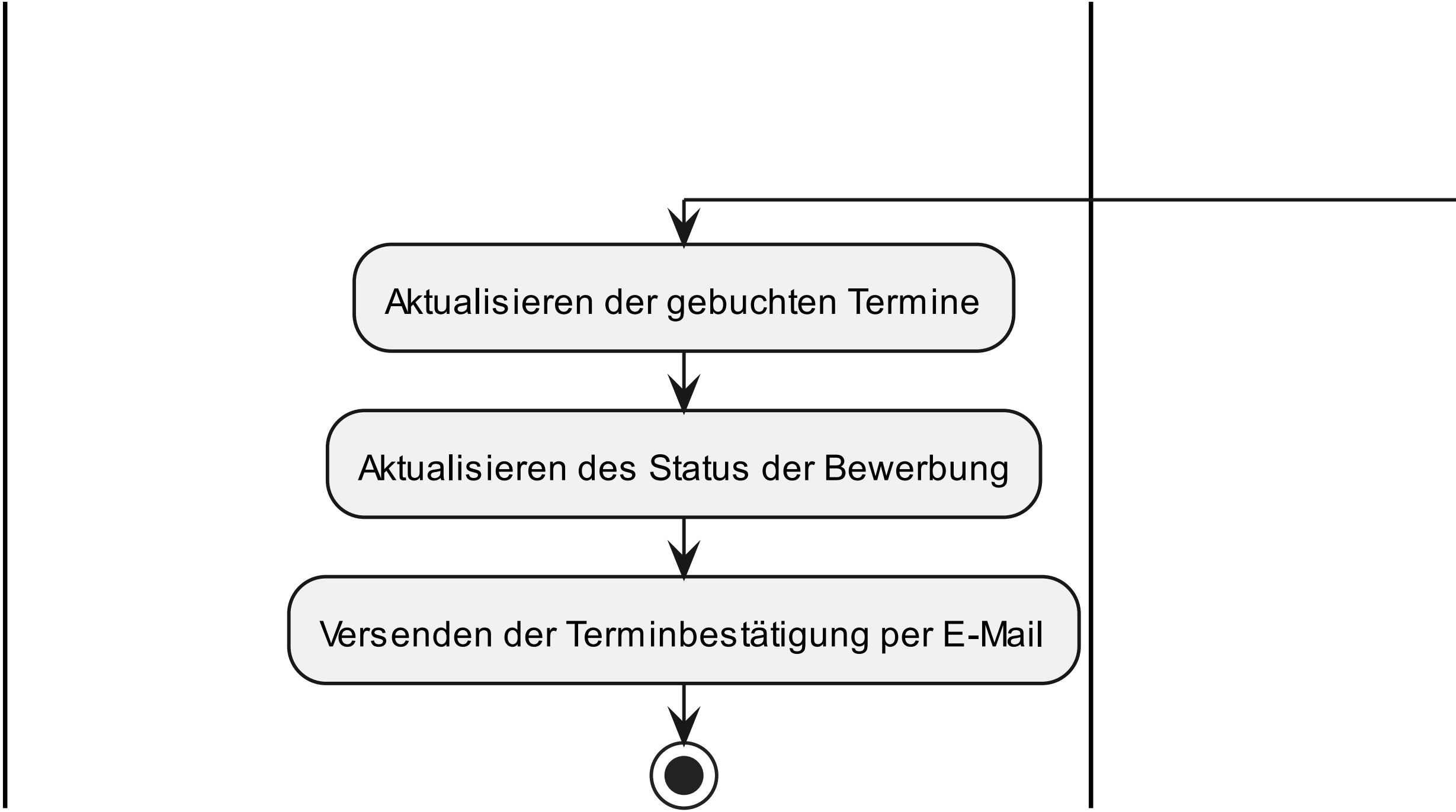

Abteilungsleiter
ROBBE
Mitarbeiter
Bewerber
ENTON
Prüfen der Bewerbung
Die Prüfung kann einige Tage dauern.
ja
Bewerber geeignet?
nein
Weiterer Ablehnungsprozess nicht dargestellt, da irrelevant für das umzusetzende Projekt.
Abfrage freier Termine
ja
Freie Termine vorhanden?
nein
Warte auf Ergänzung neuer Termine
Ergänzen neuer Termine
E-Mail mit Link zur Terminabstimmung senden
Auswählen eines Termins
Speichern des Termins
Aktualisieren der gebuchten Termine
Aktualisieren des Status der Bewerbung
Versenden der Terminbestätigung per E-Mail

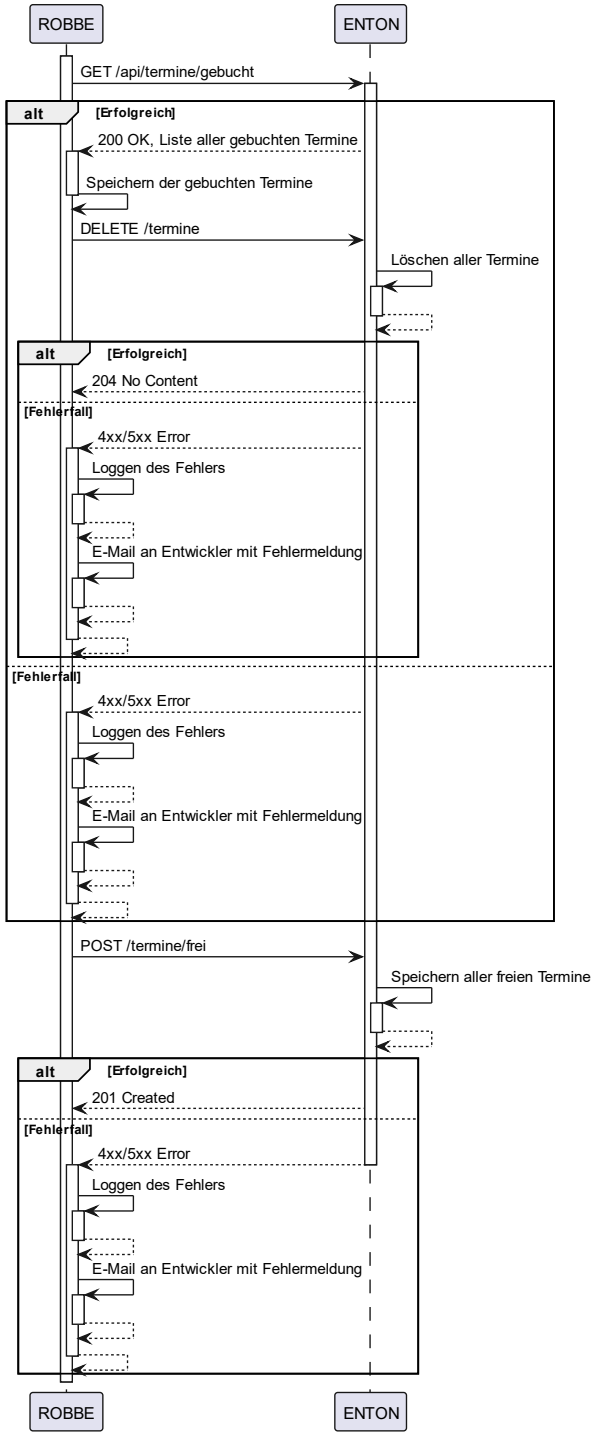
Abteilungsleiter

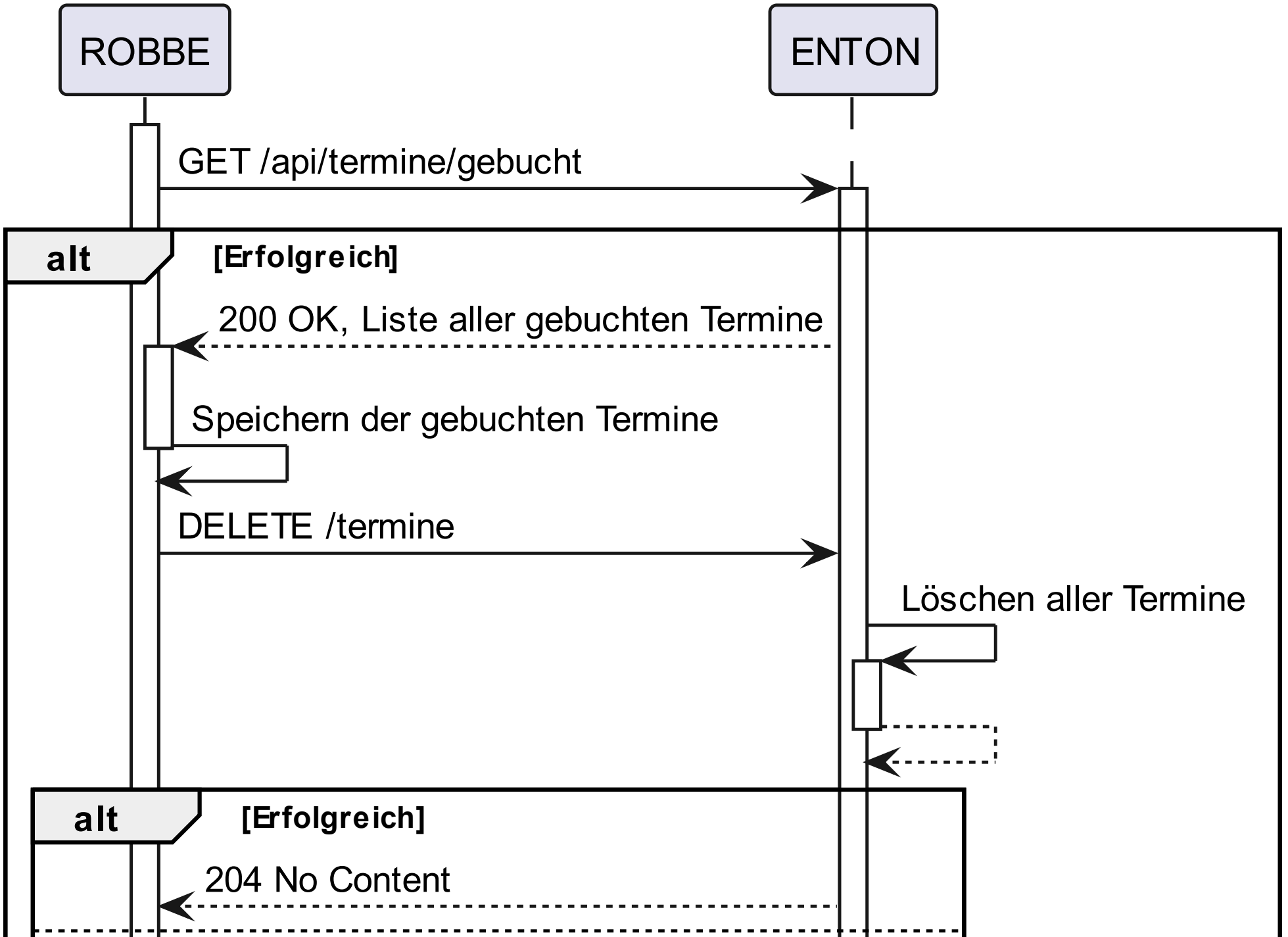


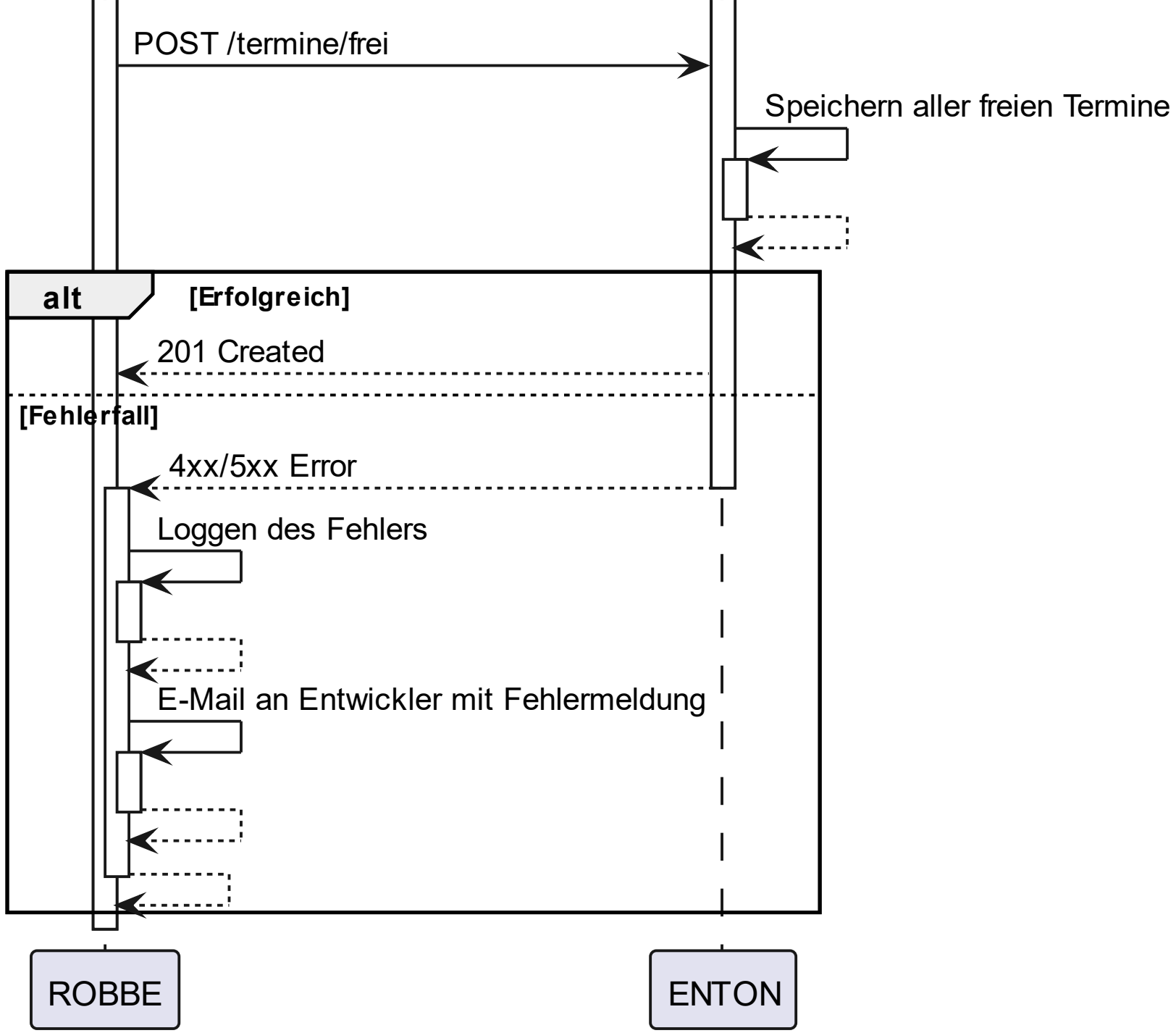

cht
Abfrage freier Termine
ja
Freie Termine vorhanden?
nein
Warte auf Ergänzung neuer Termine
Ergänzen neuer Termine
E-Mail mit Link zur Terminabstimmung senden

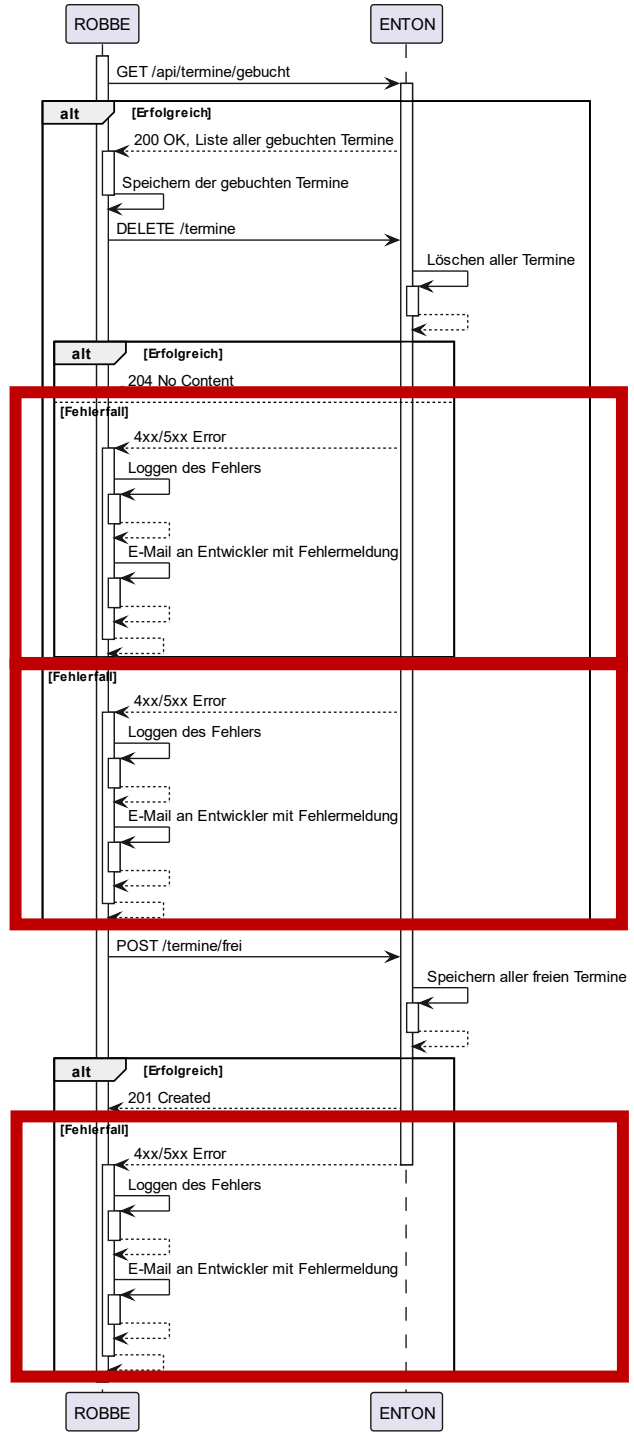

Auswählen eines Termins
Speichern des Termins


Aktualisieren der gebuchten Termine
Aktualisieren des Status der Bewerbung
Versenden der Terminbestätigung per E-Mail


ROBBE
ENTON
GET /api/termine/gebucht
alt
[Erfolgreich]
200 OK, Liste aller gebuchten Termine
Speichern der gebuchten Termine
DELETE /termine
Löschen aller Termine
alt
[Erfolgreich]
204 No Content
[Fehlerfall]
4xx/5xx Error
Loggen des Fehlers
E-Mail an Entwickler mit Fehlermeldung
[Fehlerfall]
4xx/5xx Error
Loggen des Fehlers
E-Mail an Entwickler mit Fehlermeldung
POST /termine/frei
Speichern aller freien Termine
alt
[Erfolgreich]
201 Created
[Fehlerfall]
4xx/5xx Error
Loggen des Fehlers
E-Mail an Entwickler mit Fehlermeldung
ROBBE
ENTON


ROBBE
ENTON
GET /api/termine/gebucht
alt
[Erfolgreich]
200 OK, Liste aller gebuchten Termine
Speichern der gebuchten Termine
DELETE /termine
Löschen aller Termine
alt
[Erfolgreich]
204 No Content


POST /termine/frei
Speichern aller freien Termine
alt
[Erfolgreich]
201 Created
[Fehlerfall]
4xx/5xx Error
Loggen des Fehlers
E-Mail an Entwickler mit Fehlermeldung
ROBBE
ENTON


ROBBE
ENTON
GET /api/termine/gebucht
alt
[Erfolgreich]
200 OK, Liste aller gebuchten Termine
Speichern der gebuchten Termine
DELETE /termine
Löschen aller Termine
alt
[Erfolgreich]
204 No Content
[Fehlerfall]
4xx/5xx Error
Loggen des Fehlers
E-Mail an Entwickler mit Fehlermeldung
[Fehlerfall]
4xx/5xx Error
Loggen des Fehlers
E-Mail an Entwickler mit Fehlermeldung
POST /termine/frei
Speichern aller freien Termine
alt
[Erfolgreich]
201 Created
[Fehlerfall]
4xx/5xx Error
Loggen des Fehlers
E-Mail an Entwickler mit Fehlermeldung
ROBBE
ENTON


docker


docker


MariaDB


docker


Java


MariaDB


docker


Java


MariaDB

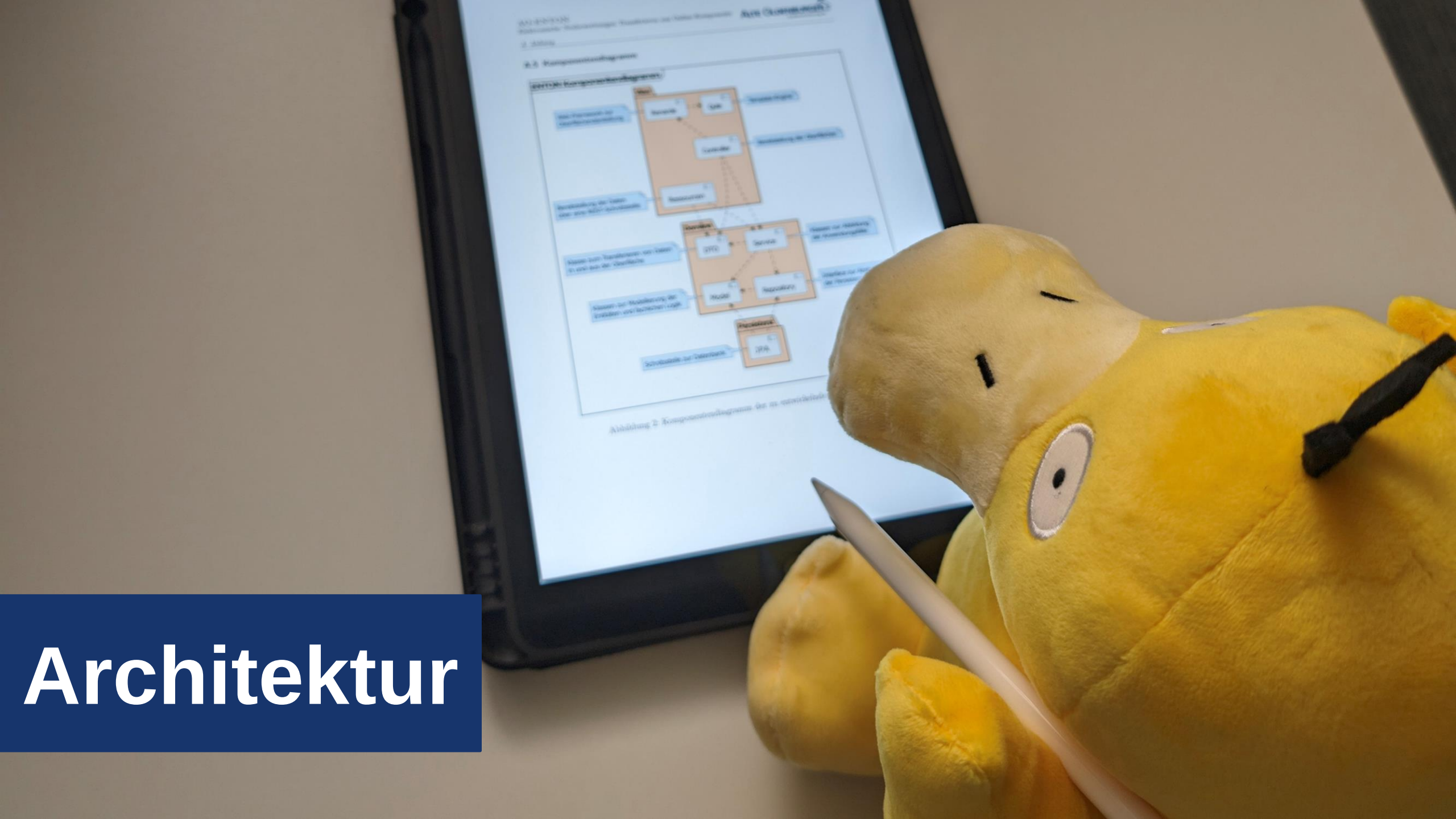

QUARKUS


docker

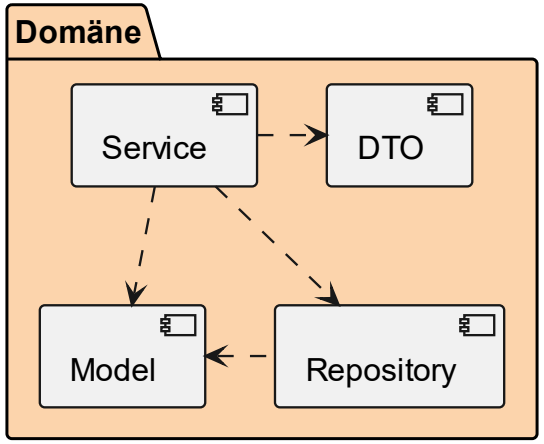

Java

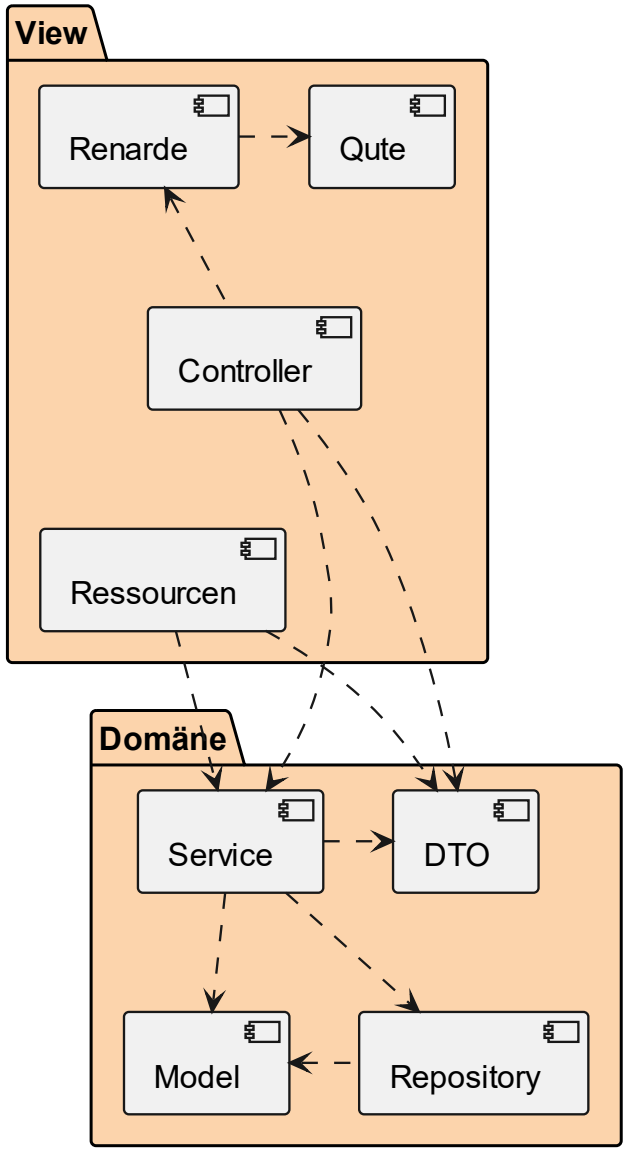

MariaDB

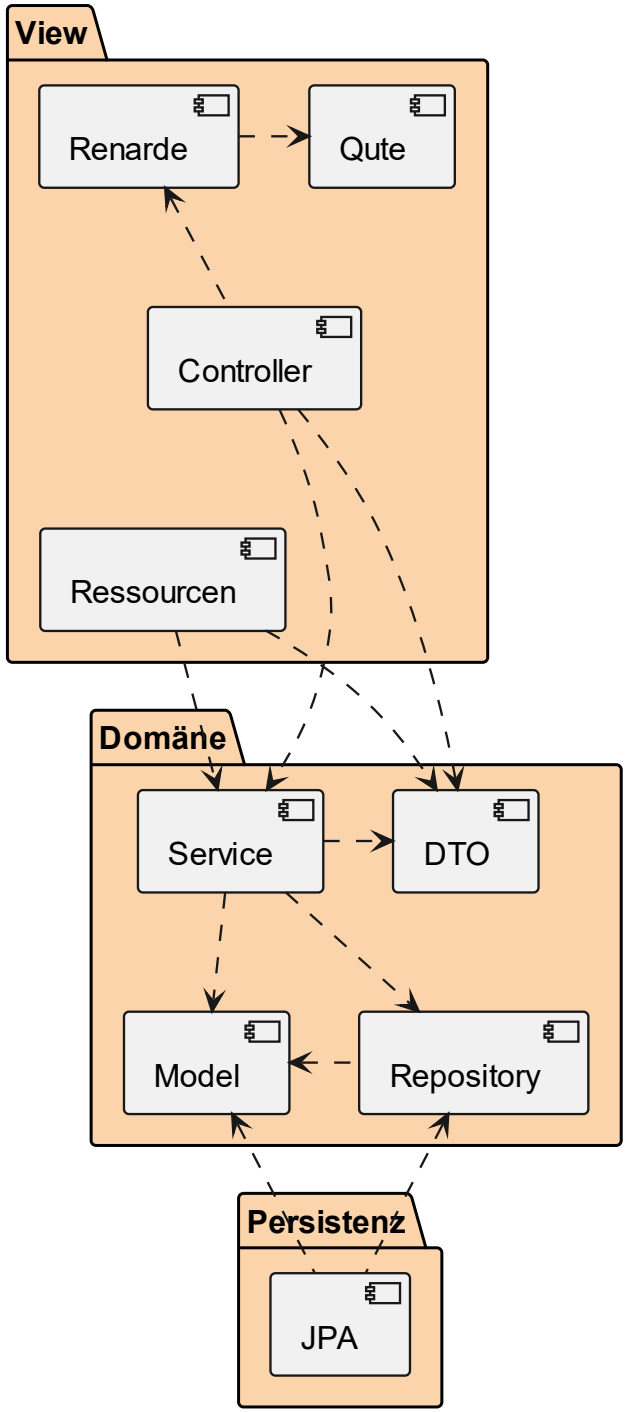

QUARKUS
JAKARTA EE

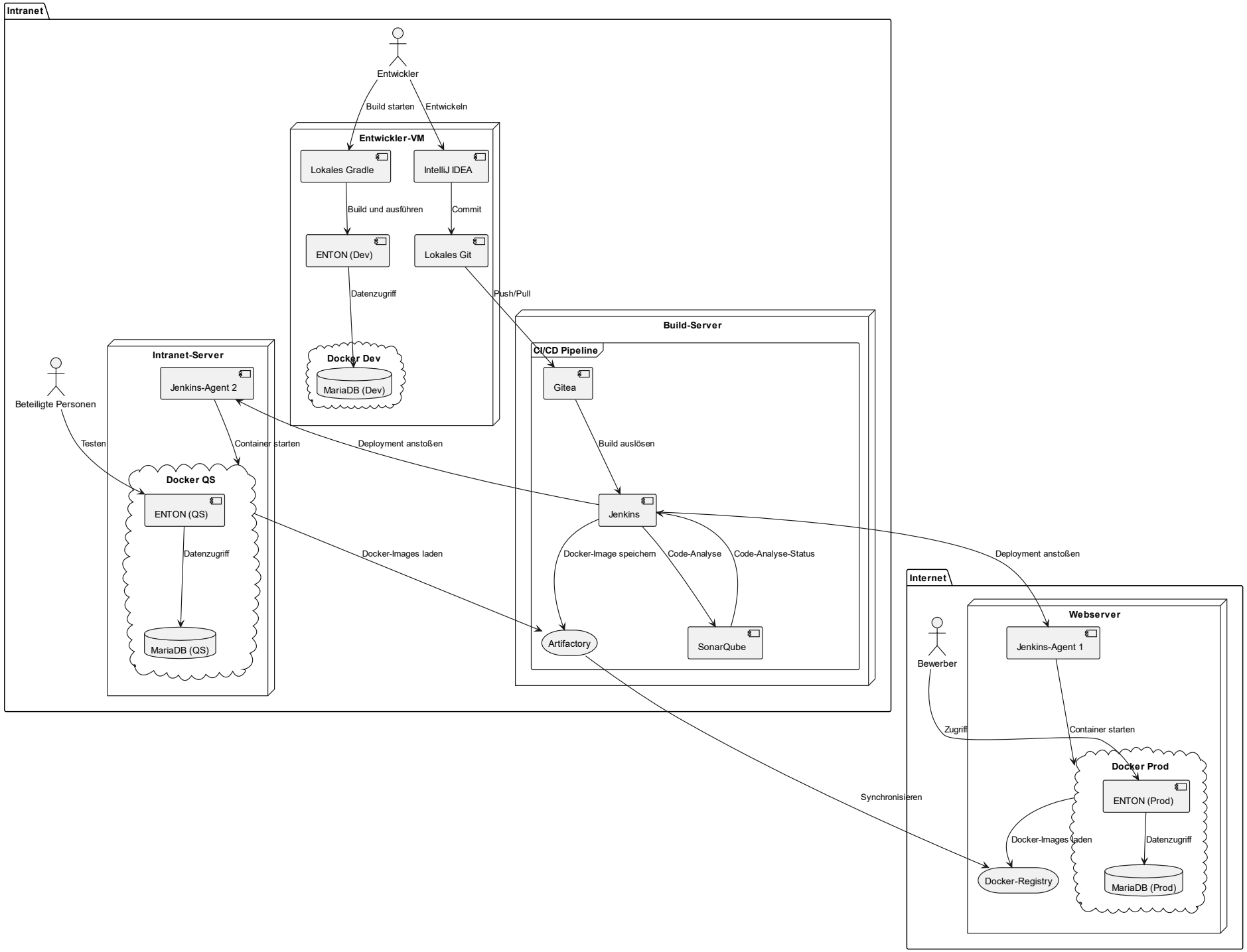


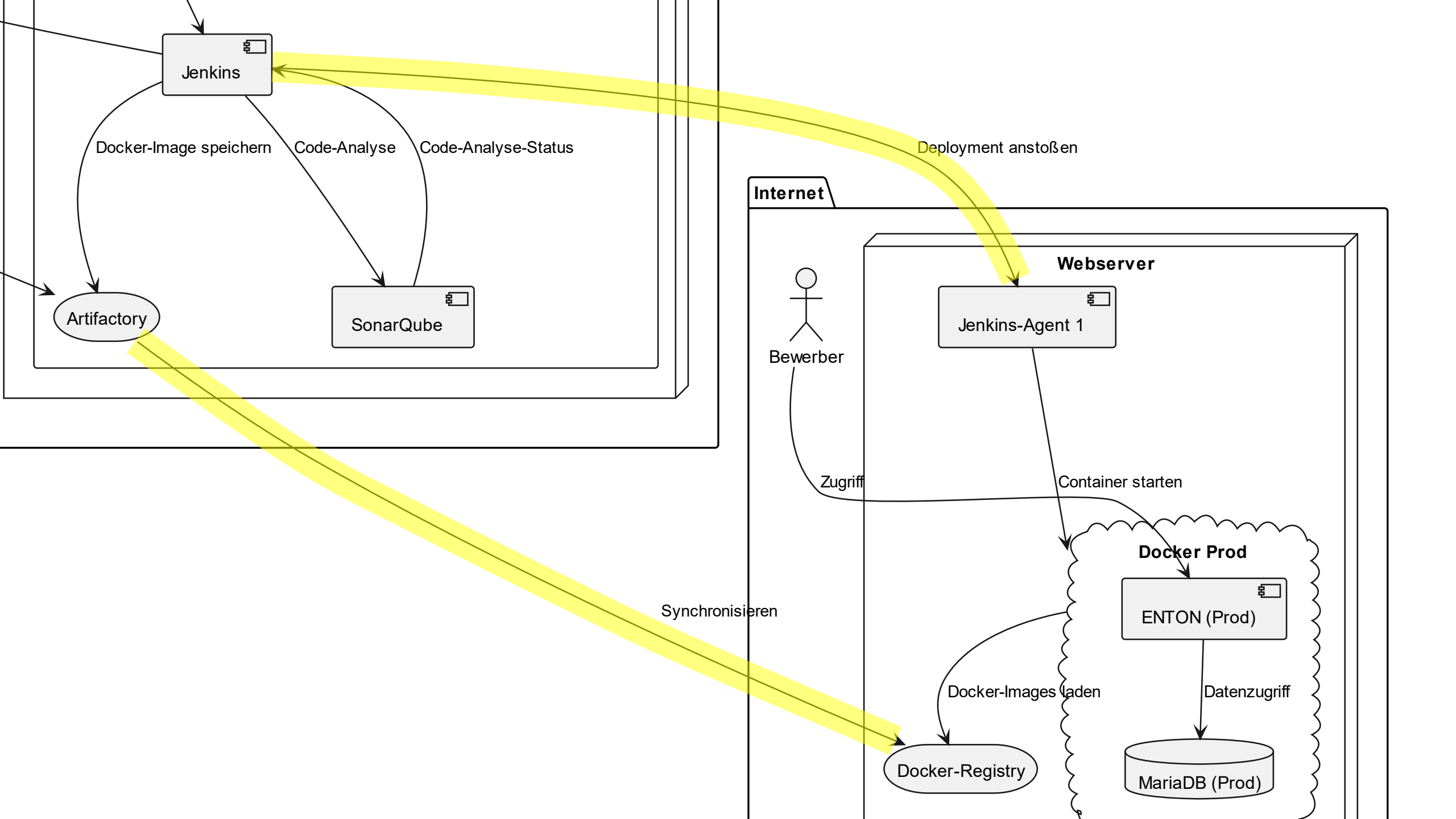
Architektur


Domäne
Service
DTO
Model
Repository


View
Renarde
Qute
Controller
Ressourcen
Domäne
Service
DTO
Model
Repository


View
Renarde
Qute
Controller
Ressourcen
Domäne
Service
DTO
Model
Repository
Persistenz
JPA


Intranet
Entwickler
Build starten
Entwickeln
Entwickler-VM
Lokales Gradle
IntelliJ IDEA
Build und ausführen
Commit
ENTON (Dev)
Lokales Git
Datenzugriff
Push/Pull
Docker Dev
MariaDB (Dev)
Build-Server
CI/CD Pipeline
Gitea
Build auslösen
Intranet-Server
Jenkins-Agent 2
Beteiligte Personen
Testen
Container starten
Deployment anstoßen
Docker QS
ENTON (QS)
Datenzugriff
MariaDB (QS)
Docker-Images laden
Jenkins
Docker-Image speichern
Code-Analyse
Code-Analyse-Status
Artifactory
SonarQube
Deployment anstoßen
Internet
Webserver
Jenkins-Agent 1
Bewerber
Zugriff
Container starten
Docker Prod
ENTON (Prod)
Synchronisieren
Docker-Images laden
Datenzugriff
Docker-Registry
MariaDB (Prod)


Jenkins
Docker-Image speichern
Code-Analyse
Code-Analyse-Status
Artifactory
SonarQube
Deployment anstoßen
Internet
Webserver
Bewerber
Jenkins-Agent 1
Zugriff
Container starten
Docker Prod
ENTON (Prod)
Synchronisieren
Docker-Images laden
Datenzugriff
Docker-Registry
MariaDB (Prod)

1. **Plattform**
    - Java (Version 17.0.2 mit Long-Term-Support) wird als Programmiersprache eingesetzt.
    - Quarkus (Version 3.3.0) wird als Java-Framework eingesetzt, welches die Jakarta EE-Spezifikationen implementiert.
    - Git (Version 2.35.1) zur Versionskontrolle des Quellcodes und Jenkins (Version 2.414.1) zur Automatisierung der Build- und Deployment-Prozesse.
    - RESTEasy (Version 5.0.8.Final) zur Implementierung der REST-APIs.
    - Docker (Version 24.0.5) dient zur Containerisierung und Vereinfachung des Deployments der gesamten Anwendung.
2. **Datenbank**
    - MariaDB (Version 11) als relationales DBMS, ausgewählt für seine Leistungsfähigkeit und Zuverlässigkeit.
    - Hibernate (Version 6), eine Implementierung von JPA, als ORM.
3. **Benutzeroberflächen**
    - Die Benutzeroberflächen werden mit Qute als Template-Engine und Renarde als Web-Framework implementiert. Renarde agiert dabei als Aufsatz auf Qute und bietet zusätzliche Funktionalitäten, wie beispielsweise vereinfachte Formularverarbeitung.
    - Bootstrap für das responsive Layout, ergänzt durch eine spezifische CSS-Datei, die das Corporate Design der Organisation umsetzt.


Faz
Entwurf
Analyse
Implementierung
Planung

Implementierung

Fazit

Entwurf

Planung

Analyse


git


git


Gradle

git

Gradle

Jenkins

Bewerbung einreichen

Für die Stelle Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Pflichtfelder

Bewerbung (PDF-Dokument)

Datei auswählen | Keine ausgewählt

Freiwillige Angaben

E-Mail

max.mustermann@mail.de

Ohne die Angabe Ihrer E-Mail-Adresse werden wir Sie auf postalischem Wege kontaktieren.

Vorname

Max

Nachname

Mustermann

Geschlecht

Wie sind Sie auf uns aufmerksam geworden?

Ich habe die Datenschutzhinweise gelesen und zur Kenntnis genommen.

Bewerbung einreichen

```
{#include includes/main}
    {#title}Bewerbung einreichen{/title}
    <section class="container mt-5">
    {#if stelle.isDefined}
        <div class="row">
            <div class="col">
                <h1 class="fw-bold">Bewerbung einreichen</h1>
                <h2>Für die Stelle {stelle.get.bezeichnung}</h2>
            </div>
        </div>
        <form action="{uri:BewerbungController.bewerbungAnlegen}" method="post" enctype="multipart/form-data" accept-charset="UTF-8">
        <fieldset>
        <legend>Pflichtfelder</legend>
        {#input name = "stelle" type = "hidden" value = stelle.get.schluessel/}
        {#formElement name = "datei" label = "Bewerbung (PDF-Dokument)"}
                {#input name = "datei" type = "file" required = "required"/}
        {/formElement}
        <legend>Freiwillige Angaben</legend>
        {#formElement name = "email" label = "E-Mail"}
                {#input name = "email" type = "email" placeholder = "max.mustermann@mail.de"/}
                {#help text = "Ohne die Angabe Ihrer E-Mail-Adresse werden wir Sie auf postalischem Wege kontaktieren."/}
        {/formElement}
        ...
    {/if}
{/include}
```

```java
@POST
@Path("/bewerben")
@Consumes(MediaType.MULTIPART_FORM_DATA)
public void bewerbungAnlegen(@BeanParam NeueBewerbung neueBewerbung) {
    var bewerbung = bewerbungService.erzeuge(neueBewerbung);
    if (istFehlerhaft(bewerbung)) {
        redirect(BewerbungController.class).bewerben(neueBewerbung.getStelle());
    }

    zeigeGlobaleMeldungAn(Meldung.BEWERBUNG_EINGEREICHT);
    var imPdfGefundeneEmail = neueBewerbung.getEmailAusPdf();
    if (imPdfGefundeneEmail != null && !imPdfGefundeneEmail.isBlank()) {
        var valideBewerbung = bewerbung.get();
        var bewerbungSchluessel = valideBewerbung.getSchluessel().toString();
        redirect(BewerbungController.class).emailGefunden(bewerbungSchluessel, imPdfGefundeneEmail);
    }

    redirect(StartseiteController.class).startseite();
}
```

```java
@POST
@Path("/bewerben")
@Consumes(MediaType.MULTIPART_FORM_DATA)
public void bewerbungAnlegen(@BeanParam NeueBewerbung neueBewerbung) {
    var bewerbung = bewerbungService.erzeuge(neueBewerbung);
    if (istFehlerhaft(bewerbung)) {
        redirect(BewerbungController.class).bewerben(neueBewerbung.getStelle());
    }

    zeigeGlobaleMeldungAn(Meldung.BEWERBUNG_EINGEREICHT);
    var imPdfGefundeneEmail = neueBewerbung.getEmailAusPdf();
    if (imPdfGefundeneEmail != null && !imPdfGefundeneEmail.isBlank()) {
        var valideBewerbung = bewerbung.get();
        var bewerbungSchluessel = valideBewerbung.getSchluessel().toString();
        redirect(BewerbungController.class).emailGefunden(bewerbungSchluessel, imPdfGefundeneEmail);
    }

    redirect(StartseiteController.class).startseite();
}
```

```java
@Transactional
public Validation<Meldungen, Bewerbung> erzeuge(NeueBewerbung neueBewerbung) {
    if (neueBewerbung.getDatenschutzhinweiseGelesen() == null) {
        return Validation.invalid(Meldungen.erzeugeFehler(Meldung.DATENSCHUTZHINWEISE_NICHT_GELESEN));
    }
    var stelle = findeEntitaet(neueBewerbung.getStelle(), stelleRepository::findeMit, Meldung.STELLE_EXISTIERT_NICHT);
    var bewerbung = Datei.erzeuge(neueBewerbung.getDatei());
    Validation<Meldungen, Quelle> optionaleQuelle = neueBewerbung.getQuelle().isBlank()
            ? Validation.valid(null)
            : findeEntitaet(neueBewerbung.getQuelle(), quelleRepository::findeMit, Meldung.QUELLE_EXISTIERT_NICHT);

    if (neueBewerbung.getEmail().isBlank()) {
        sucheEmailInPdf(neueBewerbung.getDatei())
                .peek(neueBewerbung::setEmailAusPdf);
    }

    return Validation.combine(stelle,
                    bewerbung,
                    validiereUndErzeuge(neueBewerbung.getVorname(), Vorname::erzeuge),
                    validiereUndErzeuge(neueBewerbung.getNachname(), Nachname::erzeuge),
                    validiereUndErzeuge(neueBewerbung.getGeschlecht(), Geschlecht::erzeuge),
                    validiereUndErzeuge(neueBewerbung.getEmail(), Email::erzeuge),
                    optionaleQuelle)
            .ap(Bewerbung::erzeuge)
            .mapError(Meldungen::aus)
            .flatMap(Function.identity())
            .peek(bewerbungRepository::speichere);
}
```

```
@Transactional
public Validation<Meldungen, Bewerbung> erzeuge(NeueBewerbung neueBewerbung) {
    if (neueBewerbung.getDatenschutzhinweiseGelesen() == null) {
        return Validation.invalid(Meldungen.erzeugeFehler(Meldung.DATENSCHUTZHINWEISE_NICHT_GELESEN));
    }
    var stelle = findeEntitaet(neueBewerbung.getStelle(), stelleRepository::findeMit, Meldung.STELLE_EXISTIERT_NICHT);
    var bewerbung = Datei.erzeuge(neueBewerbung.getDatei());
    Validation<Meldungen, Quelle> optionaleQuelle = neueBewerbung.getQuelle().isBlank()
```

```
if (neueBewerbung.getEmail().isBlank()) {
    sucheEmailInPdf(neueBewerbung.getDatei())
            .peek(neueBewerbung::setEmailAusPdf);
}
```

```
                validiereUndErzeuge(neueBewerbung.getEmail(), Email::erzeuge),
                optionaleQuelle)
        .ap(Bewerbung::erzeuge)
        .mapError(Meldungen::aus)
        .flatMap(Function.identity())
        .peek(bewerbungRepository::speichere);
}
```

```java
public Option<String> sucheEmailInPdf(byte[] pdfInhalt) {
    return Try.of(() -> Loader.loadPDF(pdfInhalt))
            .mapTry(geladenePdf -> new PDFTextStripper().getText(geladenePdf))
            .mapTry(gefundenerText -> Pattern
                    .compile("[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}")
                    .matcher(gefundenerText))
            .filter(Matcher::find)
            .map(Matcher::group)
            .toOption();
}
```

```java
public Option<String> sucheEmailInPdf(byte[] pdfInhalt) {
    return Try.of(() -> Loader.loadPDF(pdfInhalt))
            .mapTry(geladenePdf -> new PDFTextStripper().getText(geladenePdf))
            .mapTry(gefundenerText -> Pattern
                    .compile("[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}")
                    .matcher(gefundenerText))
            .filter(Matcher::find)
            .map(Matcher::group)
            .toOption();
}
```

```java
public Option<String> sucheEmailInPdf(byte[] pdfInhalt) {
    return Try.of(() -> Loader.loadPDF(pdfInhalt))
            .mapTry(geladenePdf -> new PDFTextStripper().getText(geladenePdf))
            .mapTry(gefundenerText -> Pattern
                    .compile("[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}")
                    .matcher(gefundenerText))
            .filter(Matcher::find)
            .map(Matcher::group)
            .toOption();
}
```

```java
@Transactional
public Validation<Meldungen, Bewerbung> erzeuge(NeueBewerbung neueBewerbung) {
    if (neueBewerbung.getDatenschutzhinweiseGelesen() == null) {
        return Validation.invalid(Meldungen.erzeugeFehler(Meldung.DATENSCHUTZHINWEISE_NICHT_GELESEN));
    }
    var stelle = findeEntitaet(neueBewerbung.getStelle(), stelleRepository::findeMit, Meldung.STELLE_EXISTIERT_NICHT);
    var bewerbung = Datei.erzeuge(neueBewerbung.getDatei());
    Validation<Meldungen, Quelle> optionaleQuelle = neueBewerbung.getQuelle().isBlank()
```

```java
if (neueBewerbung.getEmail().isBlank()) {
    sucheEmailInPdf(neueBewerbung.getDatei())
            .peek(neueBewerbung::setEmailAusPdf);
}
```

```java
                validiereUndErzeuge(neueBewerbung.getEmail(), Email::erzeuge),
                optionaleQuelle)
        .ap(Bewerbung::erzeuge)
        .mapError(Meldungen::aus)
        .flatMap(Function.identity())
        .peek(bewerbungRepository::speichere);
}
```

Vielen Dank für Ihre Bewerbung! Wir werden uns in Kürze bei Ihnen melden.

Stellenangebote bei der ALTE OLDENBURGER Krankenversicherung

Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Ausbildungsstart zum 01.08.2024.

Mehr erfahren | Jetzt bewerben

Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Ausbildungsstart zum 01.08.2025.

Mehr erfahren | Jetzt bewerben

Duales Studium Wirtschaftsinformatik 2024 (m/w/d)

Ausbildungsstart zum 01.08.2024.

Mehr erfahren | Jetzt bewerben

Duales Studium Wirtschaftsinformatik 2025 (m/w/d)

Ausbildungsstart zum 01.08.2025.

Mehr erfahren | Jetzt bewerben

Benutzerdokumentation

Laden Sie unsere umfassende Benutzerdokumentation herunter, um mehr über die Funktionalitäten dieser Plattform zu erfahren. Die Dokumentation enthält eine Schritt-für-Schritt-Anleitung für den Bewerbungsprozess. Klicken Sie dazu auf den folgenden Button:

Dokumentation herunterladen

Hilfe und Support

Für Rückfragen oder bei Problemen stehen wir Ihnen gerne zur Verfügung. Sie erreichen uns über die folgenden Kontaktmöglichkeiten:

Email: bewerbung@alte-oldenburger.de

Telefon: 04441 9050

Vielen Dank für Ihre Bewerbung! Wir werden uns in Kürze bei Ihnen melden.

Stellenangebote bei der ALTE OLDENBURGER Krankenversicherung

Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Ausbildungsstart zum 01.08.2024.

Mehr erfahren | Jetzt bewerben

Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Ausbildungsstart zum 01.08.2025.

Mehr erfahren | Jetzt bewerben

Duales Studium Wirtschaftsinformatik 2024 (m/w/d)

Ausbildungsstart zum 01.08.2024.

Mehr erfahren | Jetzt bewerben

Vielen Dank für Ihre Bewerbung! Wir werden uns in Kürze bei Ihnen melden.

Benutzerdokumentation

Laden Sie unsere umfassende Benutzerdokumentation herunter, um mehr über die Funktionalitäten dieser Plattform zu erfahren. Die Dokumentation enthält eine Schritt-für-Schritt-Anleitung für den Bewerbungsprozess. Klicken Sie dazu auf den folgenden Button:

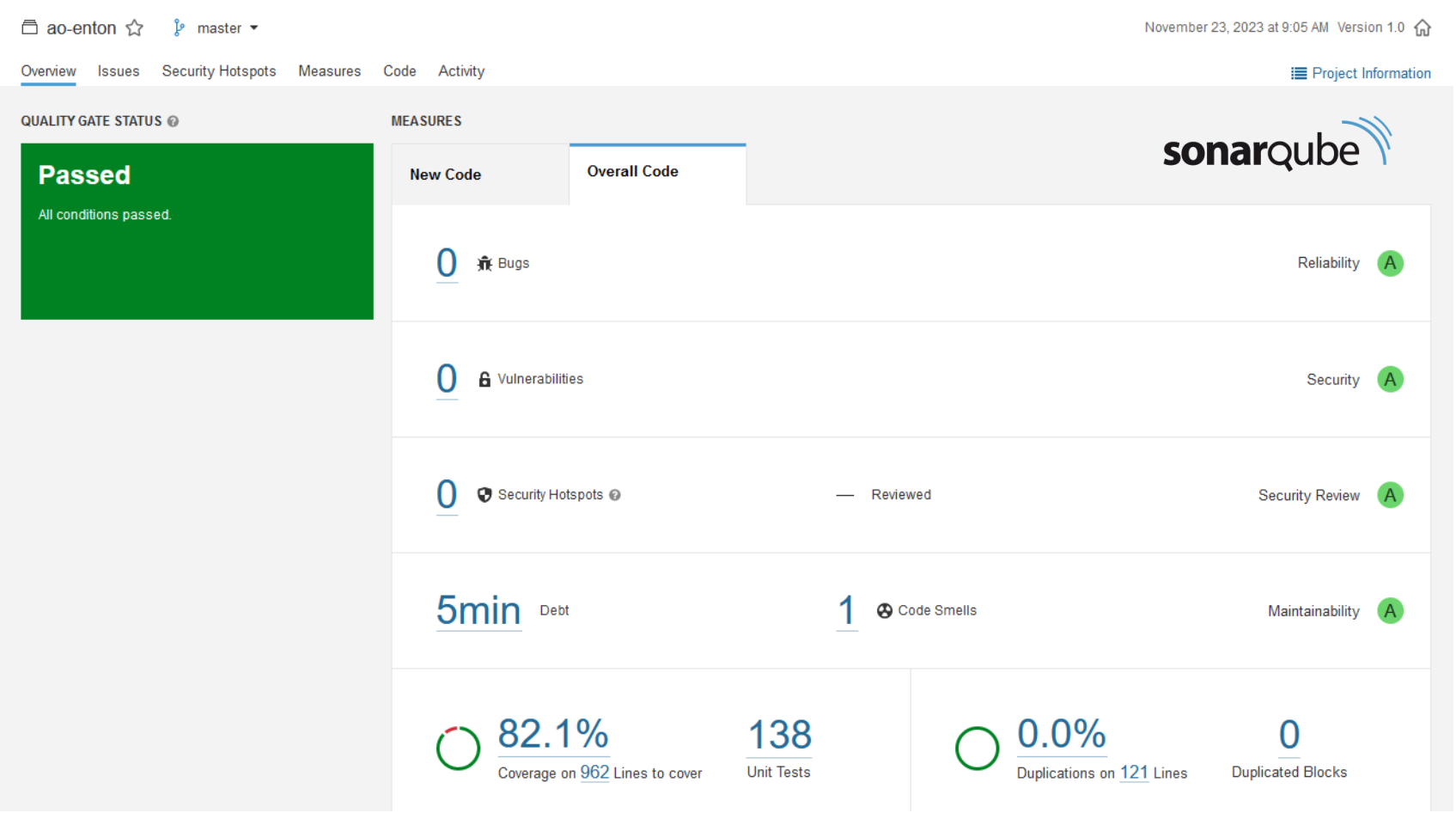
Dokumentation herunterladen

Hilfe und Support

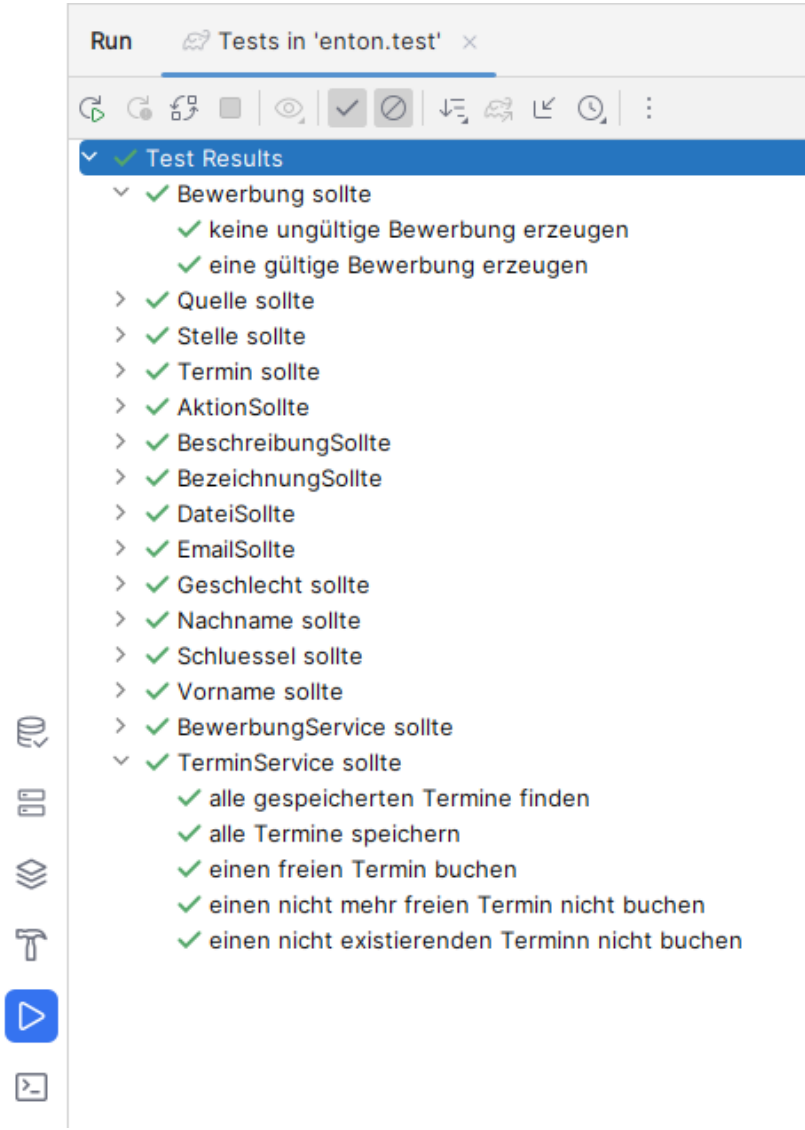
Für Rückfragen oder bei Problemen stehen wir Ihnen gerne zur Verfügung. Sie erreichen uns über die folgenden Kontaktmöglichkeiten:

Email: bewerbung@alte-oldenburger.de

Telefon: 04441 9050



Qualitätssicherung



ao-enton ☆ master

November 23, 2023 at 9:05 AM Version 1.0

Overview | Issues | Security Hotspots | Measures | Code | Activity

Project Information

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

sonarqube

New Code | Overall Code

0 Bugs — Reliability A

0 Vulnerabilities — Security A

0 Security Hotspots — Reviewed — Security Review A

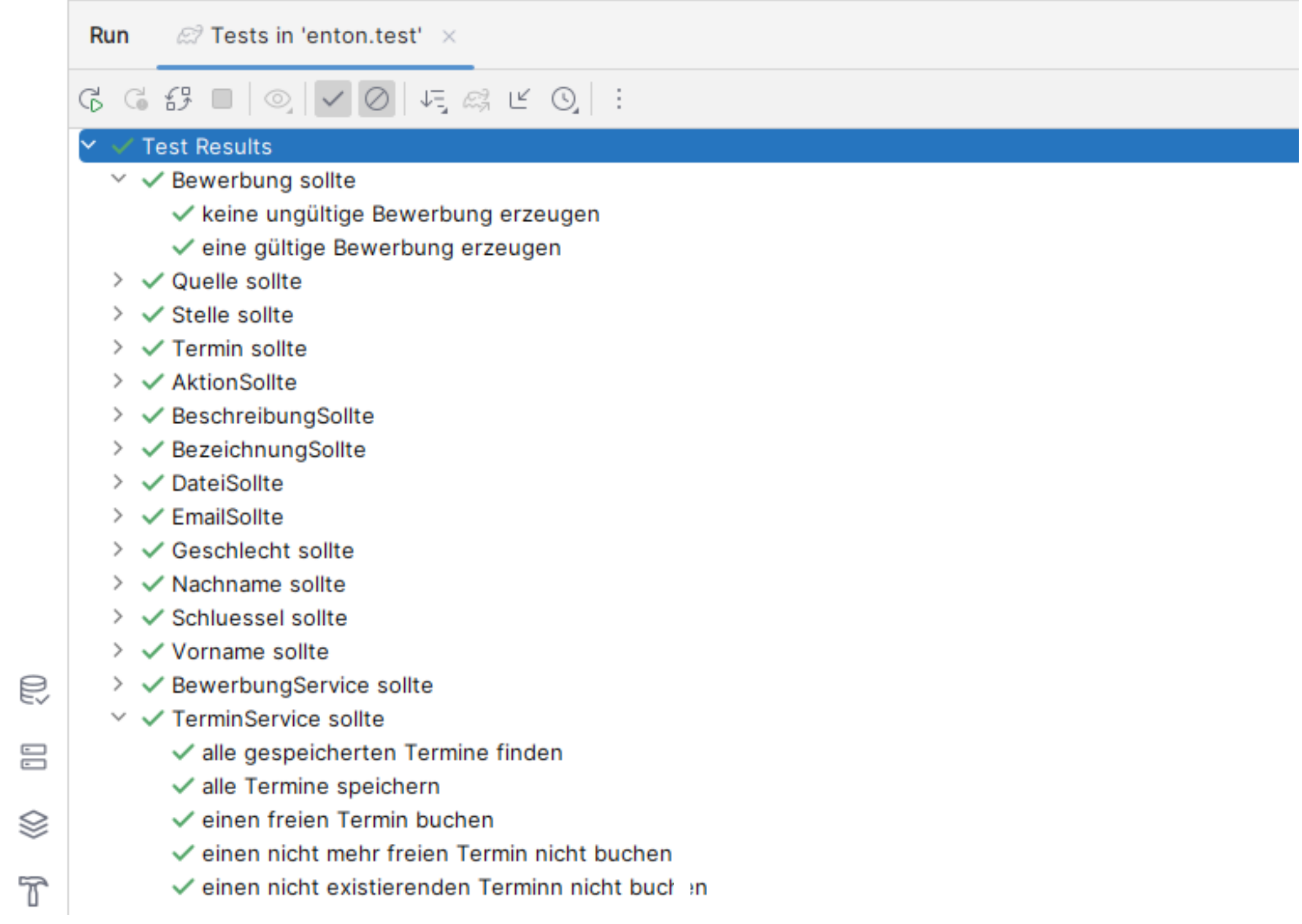
5min Debt — 1 Code Smells — Maintainability A

82.1% Coverage on 962 Lines to cover — 138 Unit Tests

0.0% Duplications on 121 Lines — 0 Duplicated Blocks

Ticket #49180: Implementiere API #5

Zusammengeführt Christian Eirich hat 8 Commits von `49180` nach `qs` vor 2 Monaten zusammengeführt


Run
Tests in 'enton.test'
Test Results
Bewerbung sollte
keine ungültige Bewerbung erzeugen
eine gültige Bewerbung erzeugen
Quelle sollte
Stelle sollte
Termin sollte
AktionSollte
BeschreibungSollte
BezeichnungSollte
DateiSollte
EmailSollte
Geschlecht sollte
Nachname sollte
Schluessel sollte
Vorname sollte
BewerbungService sollte
TerminService sollte
alle gespeicherten Termine finden
alle Termine speichern
einen freien Termin buchen
einen nicht mehr freien Termin nicht buchen
einen nicht existierenden Terminn nicht buch
en

```java
@Test
@DisplayName("einen nicht existierenden Termin nicht buchen")
void test05() {
    when(terminRepository.findeMit(any(Schluessel.class))).thenReturn(Option.none());
    var zuBuchenderTermin = new ZuBuchenderTermin(UUID.randomUUID().toString(),
            UUID.randomUUID().toString(),
            Aktion.ANWENDEN.toString());

    var ergebnis = sut.bucheTermin(zuBuchenderTermin);

    assertAll(
            () -> assertThat(ergebnis.isInvalid()).isTrue(),
            () -> assertThat(ergebnis.getError()).contains(Meldung.TERMIN_EXISTIERT_NICHT));
}
```

```java
@Test
@DisplayName("einen nicht existierenden Termin nicht buchen")
void test05() {
    when(terminRepository.findeMit(any(Schluessel.class))).thenReturn(Option.none());
    var zuBuchenderTermin = new ZuBuchenderTermin(UUID.randomUUID().toString(),
            UUID.randomUUID().toString(),
            Aktion.ANWENDEN.toString());

    var ergebnis = sut.bucheTermin(zuBuchenderTermin);

    assertAll(
            () -> assertThat(ergebnis.isInvalid()).isTrue(),
            () -> assertThat(ergebnis.getError()).contains(Meldung.TERMIN_EXISTIERT_NICHT));
}
```

```java
@Test
@DisplayName("einen nicht existierenden Termin nicht buchen")
void test05() {
    when(terminRepository.findeMit(any(Schluessel.class))).thenReturn(Option.none());
    var zuBuchenderTermin = new ZuBuchenderTermin(UUID.randomUUID().toString(),
            UUID.randomUUID().toString(),
            Aktion.ANWENDEN.toString());

    var ergebnis = sut.bucheTermin(zuBuchenderTermin);

    assertAll(
            () -> assertThat(ergebnis.isInvalid()).isTrue(),
            () -> assertThat(ergebnis.getError()).contains(Meldung.TERMIN_EXISTIERT_NICHT));
}
```

```
@Test
@DisplayName("einen nicht existierenden Termin nicht buchen")
void test05() {
    when(terminRepository.findeMit(any(Schluessel.class))).thenReturn(Option.none());
    var zuBuchenderTermin = new ZuBuchenderTermin(UUID.randomUUID().toString(),
            UUID.randomUUID().toString(),
            Aktion.ANWENDEN.toString());

    var ergebnis = sut.bucheTermin(zuBuchenderTermin);

    assertAll(
            () -> assertThat(ergebnis.isInvalid()).isTrue(),
            () -> assertThat(ergebnis.getError()).contains(Meldung.TERMIN_EXISTIERT_NICHT));
}
```

Entwicklerdokumentation

ENTON

ENTON stellt eine Online-Komponente zum bereits bestehenden Bewerbungsmanagementsystem ROBBE dar. Es ermöglicht Bewerbern, sich online zu bewerben und nach Aufforderung Termine zu vereinbaren.

Lokale Entwicklung

1. Repository klonen: `git clone https://scm.ao-devnet/Java/ao-enton.git`
2. In das Projektverzeichnis navigieren: `cd ENTON`
3. Entwicklungsmodus starten: `./gradlew quarkusDev`
4. Eine Datenbank wird nicht benötigt, da die Anwendung im Entwicklungsmodus eine In-Memory-Datenbank verwendet.
5. Die Anwendung ist lokal unter folgender URL erreichbar: `http://localhost:8080/`

Die API-Dokumentation wird automatisch von Quarkus generiert und bei jeder Änderung aktualisiert. Sie ist unter folgender URL erreichbar: `http://localhost:8080/q/swagger-ui/`

Architektur

ENTON setzt auf eine Schichtenarchitektur, die eine klare Trennung der Verantwortlichkeiten gewährleistet. Dies erleichtert die Wartbarkeit, Testbarkeit und Skalierbarkeit der Anwendung.

Frameworks und Technologien: Quarkus, MariaDB, Vavr.

Die Domänen-Schicht ist der zentrale Teil der Anwendung. Hier werden die Entitäten und die Geschäftslogik definiert.

Entitäten und der fachlichen Logik.
erung der Geschäftslogik und Use-Cases.
es für CRUD-Operationen. Die Implementierung erfolgt in der Persistenz-Schicht.
austausch zwischen UI/API und den Service-Klassen.

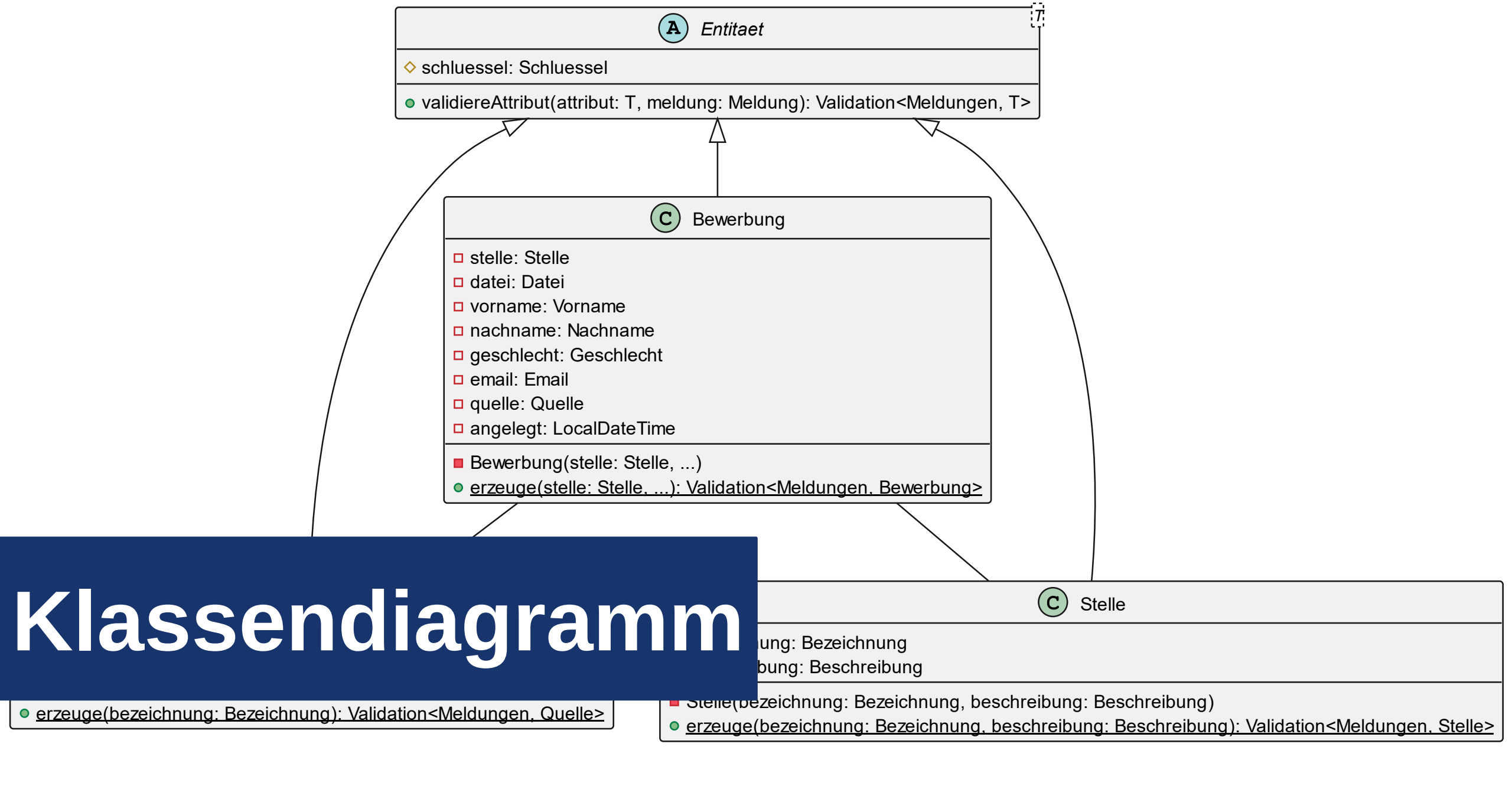
ndig für die Benutzeroberfläche. Sie verwendet Qute und Renarde für dynamische Webseiten.

llt die REST-API von ENTON bereit und enthält spezialisierte Klassen für die Interaktion mit ROBBE.

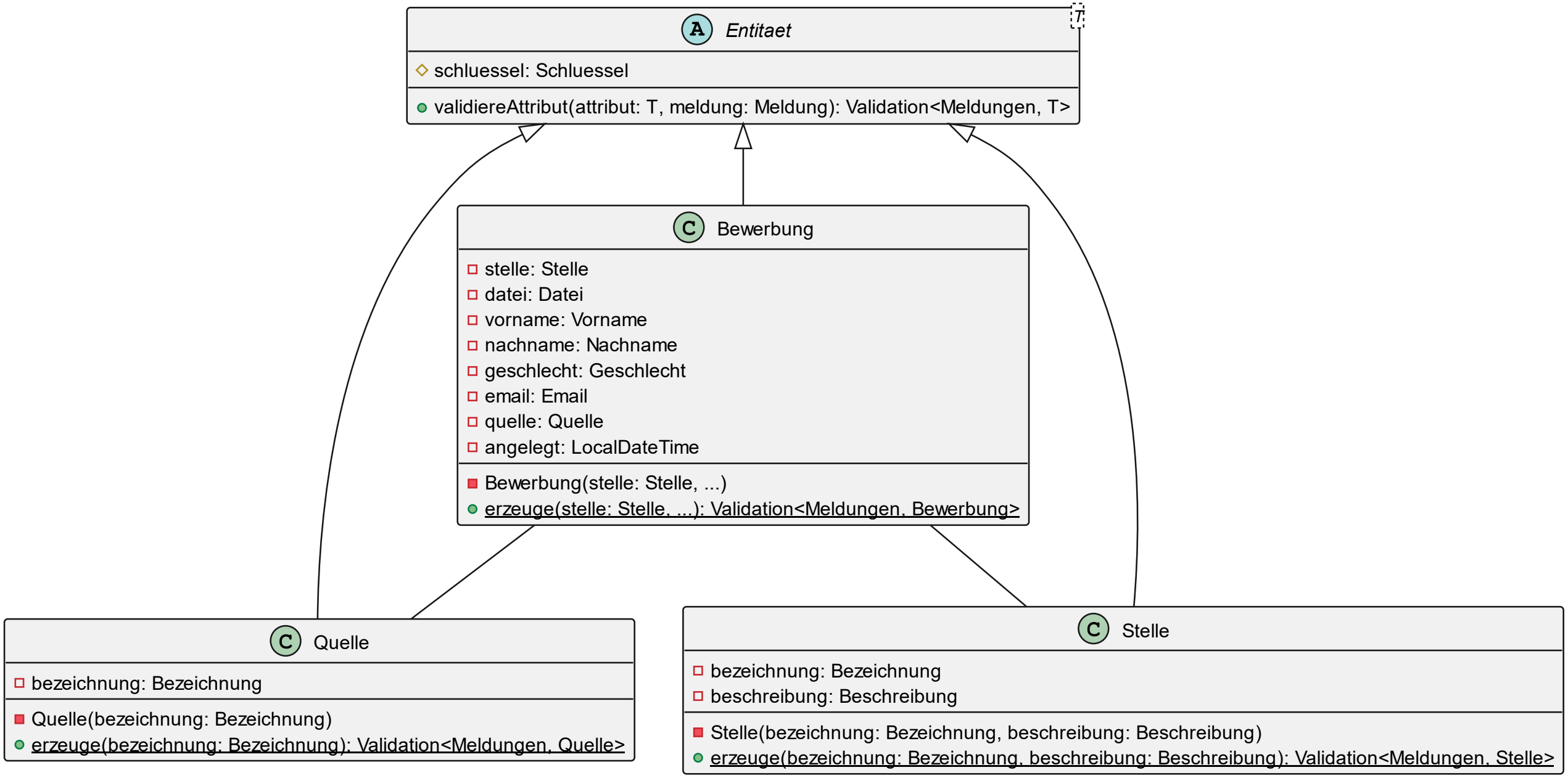
Die Persistence-Schicht ist die Schnittstelle zur Datenbank. Sie nutzt JPA für die Objekt-Relationale Abbildung.

Glossar

README



Klassendiagramm


Entitaet
T
schluessel: Schluessel
validiereAttribut(attribut: T, meldung: Meldung): Validation<Meldungen, T>
Bewerbung
stelle: Stelle
datei: Datei
vorname: Vorname
nachname: Nachname
geschlecht: Geschlecht
email: Email
quelle: Quelle
angelegt: LocalDateTime
Bewerbung(stelle: Stelle, ...)
erzeuge(stelle: Stelle, ...): Validation<Meldungen, Bewerbung>
Quelle
bezeichnung: Bezeichnung
Quelle(bezeichnung: Bezeichnung)
erzeuge(bezeichnung: Bezeichnung): Validation<Meldungen, Quelle>
Stelle
bezeichnung: Bezeichnung
beschreibung: Beschreibung
Stelle(bezeichnung: Bezeichnung, beschreibung: Beschreibung)
erzeuge(bezeichnung: Bezeichnung, beschreibung: Beschreibung): Validation<Meldungen, Stelle>

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
io.vavr.control.Validation<Meldungen,Bewerbung>	erzeuge(NeueBewerbung neueBewerbung)	Erstellt eine neue Bewerbung und speichert sie in der Datenbank.
List<GespeicherteBewerbung>	findeAlle()	Ruft alle gespeicherten Bewerbungen aus der Datenbank ab.
io.vavr.control.Validation<Meldungen,Bewerbung>	findeMit(String bewerbungSchluessel)	Sucht eine Bewerbung in der Datenbank anhand ihres eindeutigen Schlüssels.
io.vavr.control.Validation<Meldungen,GefundeneEmail>	setzeGefundeneEmail(GefundeneEmail gefundeneEmail)	Aktualisiert die E-Mail-Adresse einer Bewerbung in der Datenbank.
io.vavr.control.Option<String>	sucheEmailInPdf(byte[] pdfInhalt)	Durchsucht den Inhalt eines PDF-Dokuments nach einer E-Mail-Adresse.

Methods inherited from class net.aokv.enton.domain.service.Service

findeEntitaet, findeMit

oject

ashCode, notify, notifyAll, toString, wait, wait, wait

JavaDoc

Method Details

sucheEmailInPdf

`public io.vavr.control.Option<String> sucheEmailInPdf(byte[] pdfInhalt)`

Durchsucht den Inhalt eines PDF-Dokuments nach einer E-Mail-Adresse. Verwendet RegEx zur Identifizierung der E-Mail.

Parameters:

`pdfInhalt` - Byte-Array, das den Inhalt des PDF-Dokuments enthält.

Returns:

Eine Option, die die gefundene E-Mail-Adresse enthält, oder None, wenn keine gefunden wurde.

erzeuge

`public io.vavr.control.Validation<Meldungen,Bewerbung> erzeuge(NeueBewerbung neueBewerbung)`

Erstellt eine neue Bewerbung und speichert sie in der Datenbank. Diese Methode führt auch Validierungen durch und setzt die E-Mail-Adresse aus dem PDF, falls nicht angegeben.

Parameters:

`neueBewerbung` - Das DTO, das die Daten für die neue Bewerbung enthält.

Returns:

Eine Validation-Instanz, die entweder die erfolgreich gespeicherte Bewerbung oder eine Liste von Meldungen enthält.

findeAlle

`public List<GespeicherteBewerbung> findeAlle()`

Ruft alle gespeicherten Bewerbungen aus der Datenbank ab.

Returns:

Eine Liste von GespeicherteBewerbung-DTOs, die die Daten aller gespeicherten Bewerbungen enthalten.

findeMit

`public io.vavr.control.Validation<Meldungen,Bewerbung> findeMit(String bewerbungSchluessel)`

Sucht eine Bewerbung in der Datenbank anhand ihres eindeutigen Schlüssels.

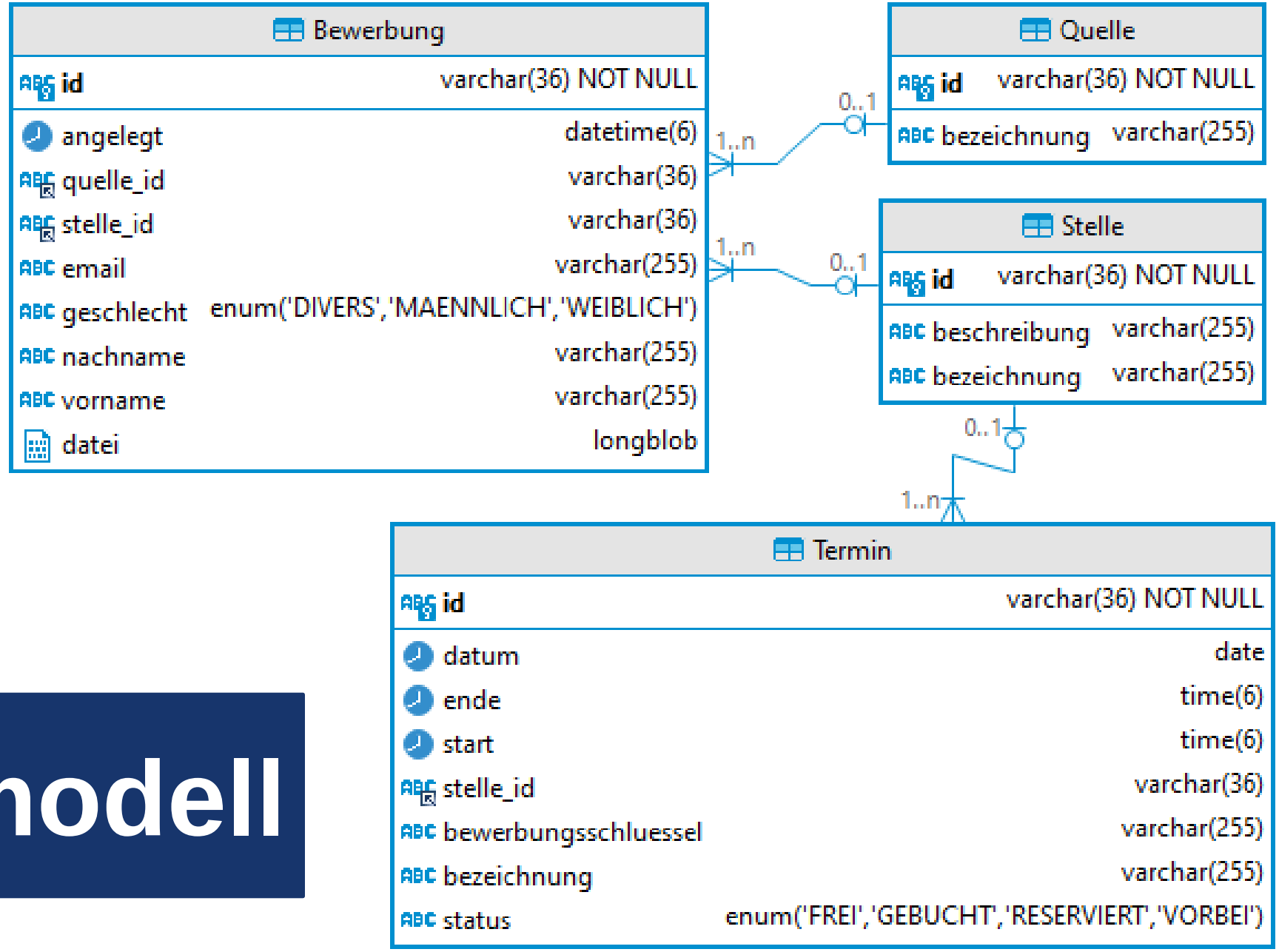
Parameters:

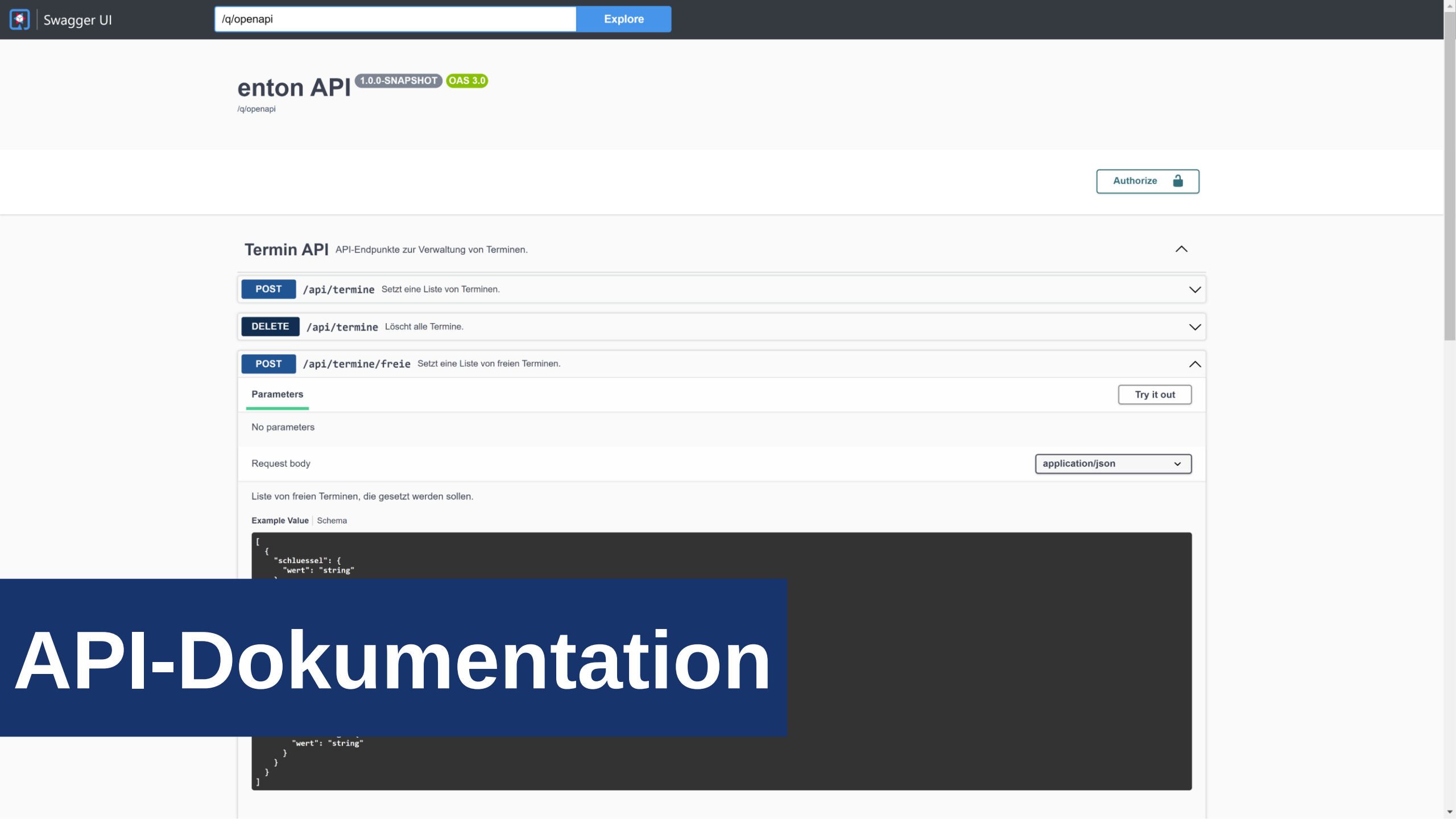
`bewerbungSchluessel` - Der eindeutige Schlüssel der Bewerbung als String.

Returns:

Eine Validation-Instanz, die entweder die gefundene Bewerbung oder eine Liste von Meldungen enthält.

Tabellenmodell




Swagger UI
/q/openapi
Explore
enton API
1.0.0-SNAPSHOT
OAS 3.0
/q/openapi
Authorize
Termin API
API-Endpunkte zur Verwaltung von Terminen.
POST
/api/termine
Setzt eine Liste von Terminen.
DELETE
/api/termine
Löscht alle Termine.
POST
/api/termine/freie
Setzt eine Liste von freien Terminen.
Parameters
Try it out
No parameters
Request body
application/json
Liste von freien Terminen, die gesetzt werden sollen.
Example Value
Schema
[
{
"schluessel": {
"wert": "string"
"wert": "string"
}
}
}
]
API-Dokumentation

Swagger UI

/q/openapi

Explore

enton API 1.0.0-SNAPSHOT OAS 3.0

/q/openapi

Authorize

Termin API API-Endpunkte zur Verwaltung von Terminen.

POST /api/termine Setzt eine Liste von Terminen.

DELETE /api/termine Löscht alle Termine.

POST /api/termine/freie Setzt eine Liste von freien Terminen.

Parameters

Try it out

No parameters

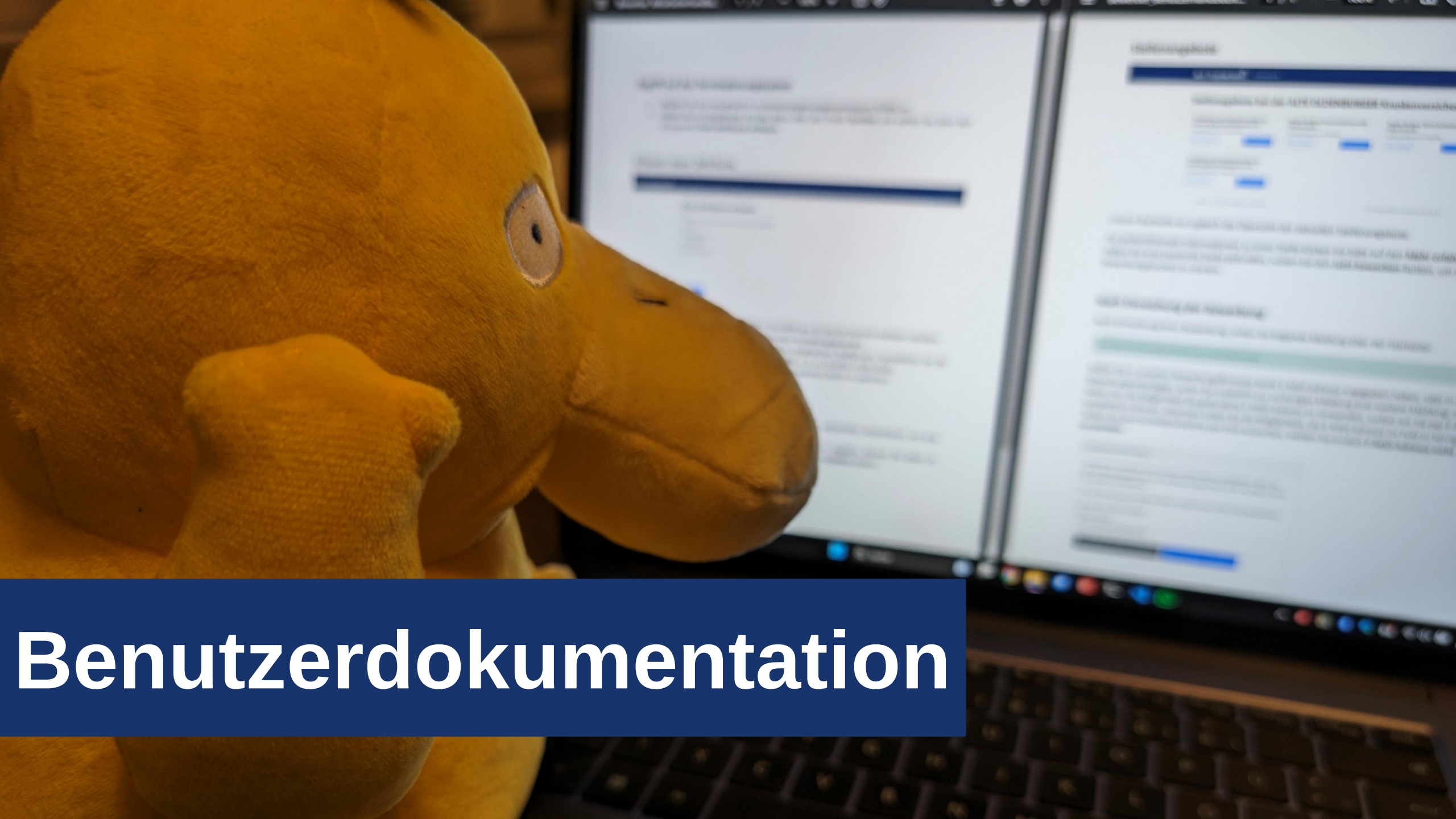
Request body

application/json

Liste von freien Terminen, die gesetzt werden sollen.

Example Value | Schema


```
[
  {
    "schluessel": {
      "wert": "string"
    },
    "bezeichnung": {
      "wert": "string"
    },
    "bewerbungsschluessel": "string",
    "datum": "2022-03-10",
    "start": "13:45.30.123456789",
    "ende": "13:45.30.123456789",
    "status": "FREI",
    "stelle": {
      "schluessel": {
        "wert": "string"
      },
      "bezeichnung": {
        "wert": "string"
      },
      "beschreibung": {
        "wert": "string"
      }
    }
  }
]
```



Benutzerdokumentation

Stellenangebote bei der ALTE OLDENBURGER Krankenversicherung

Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Ausbildungsstart zum 01.08.2024.

Mehr erfahren | Jetzt bewerben

Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Ausbildungsstart zum 01.08.2025.

Mehr erfahren | Jetzt bewerben

Duales Studium Wirtschaftsinformatik 2024 (m/w/d)

Ausbildungsstart zum 01.08.2024.

Mehr erfahren | Jetzt bewerben

Duales Studium Wirtschaftsinformatik 2025 (m/w/d)

Ausbildungsstart zum 01.08.2025.

Mehr erfahren | Jetzt bewerben

Benutzerdokumentation

Laden Sie unsere umfassende Benutzerdokumentation herunter, um mehr über die Funktionalitäten dieser Plattform zu erfahren. Die Dokumentation enthält eine Schritt-für-Schritt-Anleitung für den Bewerbungsprozess. Klicken Sie dazu auf den folgenden Button:

Dokumentation herunterladen

Hilfe und Support

Für Rückfragen oder bei Problemen stehen wir Ihnen gerne zur Verfügung. Sie erreichen uns über die folgenden Kontaktmöglichkeiten:

Email: bewerbung@alte-oldenburger.de

Telefon: 04441 9050


Fazit
Implementierung
Entwurf
Planung
Analyse

Fazit


Brigador

Bewerbung einreichen

Für die Stelle Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)

Pflichtfelder

Bewerbung (PDF-Dokument)

Datei auswählen | Keine ausgewählt

Freiwillige Angaben

E-Mail

max.mustermann@mail.de

Ohne die Angabe Ihrer E-Mail-Adresse werden wir Sie auf postalischem Wege kontaktieren.

Vorname

Max

Nachname

Mustermann

Geschlecht

Wie sind Sie auf uns aufmerksam geworden?

Ich habe die Datenschutzhinweise gelesen und zur Kenntnis genommen.

Bewerbung einreichen

Termin auswählen

Bitte wählen Sie aus der folgenden Liste einen Termin, der Ihnen passt.

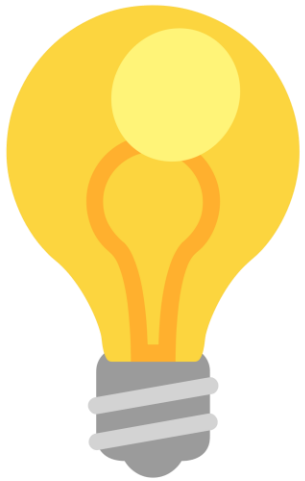
Sie wurden eingeladen zu **Vorstellungsgespräch**.
Sie haben sich auf die Stelle **Ausbildung zum Fachinformatiker für Anwendungsentwicklung (m/w/d)** beworben.

Verfügbare Termine

Datum	Zeitraum	Auswählen
Sonntag, 5. November 2023	13:00 - 14:00 Uhr	○
Donnerstag, 2. November 2023	10:00 - 11:00 Uhr	○
Mittwoch, 1. November 2023	09:00 - 10:00 Uhr	○
Freitag, 3. November 2023	11:00 - 12:00 Uhr	○
Samstag, 4. November 2023	12:00 - 13:00 Uhr	○

Keiner der Termine passt

Termin buchen



LESSONS LEARNED

Ausblick

ALTE OLDENBURGER
Private Krankenversicherung

Projektname: ENTON

Prüfling: Christian Eirich