ARCHITEKTURSPEZIFIKATION
PRICE TRACKER

Metin Kerem Öztürk, e200503057@stud.tau.edu.tr
Yavuz Selim Tunçer, e200503001@stud.tau.edu.tr

TÜRK-ALMAN ÜNİVERSİTESİ
TÜRKISCH-DEUTSCHE UNIVERSITÄT

PRICE TRACKER

Stakeholder: DI. Ömer Karacan, omer.karacan@tau.edu.tr

INHALTSVERZEICHNIS

ARCHITEKTURÜBERBLICK

Präsentation

Frontend-Integration

Webserver

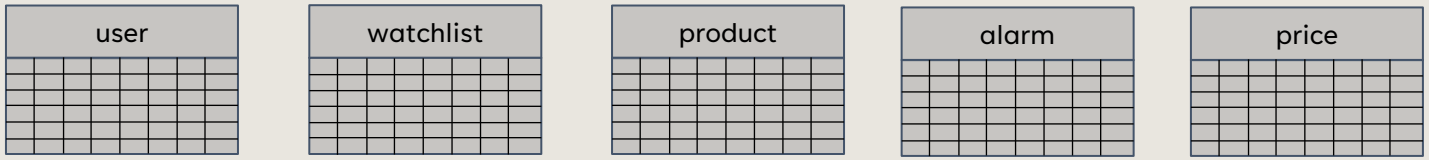
UserController | WatchlistController | AlarmController | PriceController

UserService | WatchlistService | ProductService | AlarmService | PriceService | EmailService

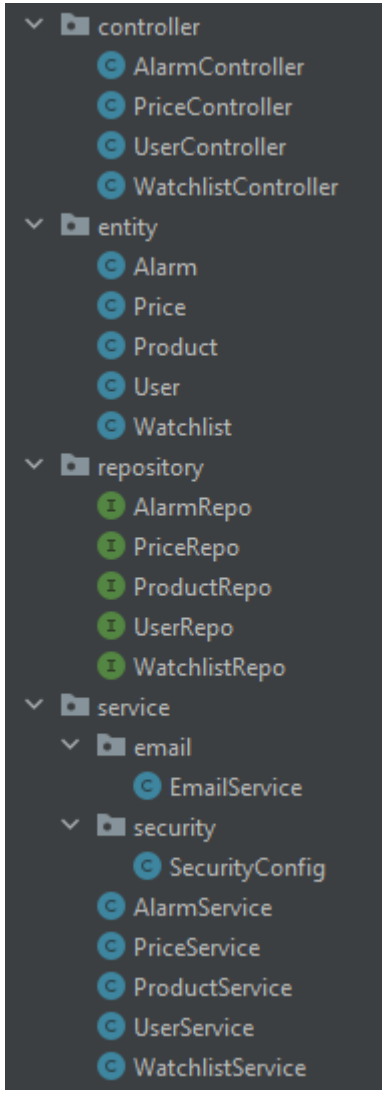
UserRepo | WatchlistRepo | ProductRepo | AlarmRepo | PriceRepo

Microservices

Datenbank



[Github](#)



„CONTROLLER“ KLASSEN

UserController	WatchlistController	AlarmController	PriceController
getUserByEmail createUser login	getWatchlistsForUser createWatchlistForUser deleteWatchlist setAlarm deleteAlarm	loadAlarm	startChecking listPrices

USER CONTROLLER

getUserByEmail - GET

Es verarbeitet eine GET-Anfrage, um Benutzerdetails basierend auf einem bereitgestellten E-Mail-Parameter abzurufen. Wenn der Benutzer existiert, werden seine Details ohne Passwort im JSON-Format mit einem HTTP-Statuscode 200 zurückgegeben. Wenn der Benutzer nicht gefunden wird, wird eine Fehlermeldung mit einem HTTP-Statuscode 404 zurückgegeben.

Antwort

```
{
   "id": 1,
   "email": "james.bond@gmail.com",
   "name": "james",
   "surname": "bond",
   "password": null
}
```

User

- int id
- String email
- String name
- String surname
- String password
- List<Watchlist> watchlists

USER CONTROLLER

createUser - POST

User
-int id
-String email
-String name
-String surname
-String password
-List<Watchlist> watchlists

Es prüft anhand der angegebenen E-Mail-Adresse, ob der Benutzer bereits existiert. Wenn der Benutzer existiert, wird eine Konfliktantwort mit dem HTTP-Statuscode 409 zurückgegeben. Wenn der Benutzer nicht existiert, erstellt der Code einen neuen Benutzer mit den bereitgestellten Details, einschließlich eines Kennworts, das normalerweise aus Sicherheitsgründen verschlüsselt werden sollte. Es hat sich bewährt, Benutzerkennwörter in einem verschlüsselten Format zu speichern, um sie vor unbefugtem Zugriff zu schützen. Nachdem der neue Benutzer erstellt wurde, gibt der Code eine Erfolgsantwort im JSON-Format mit dem HTTP-Statuscode 200 (OK) zurück.

Request body

```
{
   "email": "anakin@gmail.com",
   "name": "anakin",
   "surname": " skywalker ",
   "password": "padme"
}
```

Antwort

```
{
   "id": 2,
   "email": " anakin@gmail.com",
   "name": "anakin",
   "surname": "skywalker",
   "password": "$2a$10$86i4LVG/FpO3n5Pujlv9Le88T1A3uUN7dBMBDZQEb5EkWP2697pK."
}
```

USER CONTROLLER

login - POST

Es akzeptiert die E-Mail-Adresse und das Passwort des Benutzers als Anmeldeinformationen im JSON-Format. Der Code prüft dann, ob ein Benutzer mit der angegebenen E-Mail im System vorhanden ist. Wenn der Benutzer nicht existiert, wird eine 404-Antwort zurückgegeben, die anzeigt, dass der Benutzer nicht gefunden wurde. Wenn der Benutzer existiert, prüft der Code, ob das bereitgestellte Passwort mit dem im System gespeicherten Passwort übereinstimmt. Wenn das Passwort nicht übereinstimmt, gibt der Code eine 401-Antwort mit einer Fehlermeldung zurück. Wenn das Passwort übereinstimmt, gibt der Code eine Erfolgsantwort im JSON-Format mit einem HTTP-Statuscode von 200 (OK) zurück, einschließlich der Informationen des Benutzers.

User

- int id
- String email
- String name
- String surname
- String password
- List<Watchlist> watchlists

Request body

```
{
   "email": "anakin@gmail.com",
   "password": "padme"
}
```

Antwort

```
{
   "id": 2,
   "email": " anakin@gmail.com",
   "name": "anakin",
   "surname": "skywalker",
   "password": "$2a$10$86i4LVG/FpO3n5Pujlv9Le88T1A3uUN7dBMBDZQEb5EkWP2697pK."
}
```

WATCHLIST CONTROLLER

getWatchlistsForUser - GET

Wenn eine GET-Anforderung an diese URL mit einem bestimmten user_id-Wert gestellt wird, wird diese Methode ausgelöst und gibt eine Antwort in Form einer Liste von Watchlist-Objekten zurück, die sich auf diese user_id beziehen. Die Antwort wird in ein ResponseEntity-Objekt eingeschlossen, das den HTTP-Statuscode auf FOUND (302) setzt, was anzeigt, dass die angeforderte Ressource gefunden wurde und der Client zu einer anderen URL umleiten sollte.

Antwort


```
[
  {
    "id": 1,
    "product": {
      "id": 1,
      "name": "Arsen Lüpen Seçme Eserler - Maurice Leblanc",
      "image": "https://productimages.hepsiburada.net/s/55/550/11201887076402.jpg",
      "url": "https://www.hepsiburada.com/arsen-lupen-secme-eserler-maurice-leblanc-p-hbv0000196ai1"
    },
    "alarm": {
      "id": 10,
      "productId": 1,
      "watchlist_id": 1,
      "date_created": "2023-04-28",
      "condition": "BELOW_TARGET",
      "target_price": 150.0,
      "date_triggered": "2023-05-03"
    }
]
```


Watchlist

- int id
- Product product
- Alarm alarm
- User user

Product

- int id
- String name
- String image
- String url
- List<Watchlist> watchlists
- List<Price> prices

WATCHLIST CONTROLLER

createWatchlistForUser - POST

Es nimmt E-Mail-, Namens-, URL- und Bildparameter, erstellt ein neues Produktobjekt und prüft, ob ein Benutzer mit der angegebenen E-Mail-Adresse im System vorhanden ist. Wenn der Benutzer existiert, wird geprüft, ob ein Produkt mit der angegebenen URL existiert. Wenn es nicht vorhanden ist, erstellt es ein neues Product-Objekt und gibt eine Antwort mit einem vom watchlistService erstellten Watchlist-Objekt zurück. Wenn das Produkt bereits vorhanden ist, wird das vorhandene Produktobjekt zurückgegeben. Wenn der Benutzer nicht existiert, wird eine NOT_FOUND-Antwort zurückgegeben. Die Antwort wird in ein ResponseEntity-Objekt eingeschlossen, das den HTTP-Statuscode je nach Ergebnis der Operation auf CREATED oder NOT_FOUND setzt.

Request parameters
email=....
&name=...
&url=...
&image=...

Antwort

```
{
   "id": 6,
   "product": {
      "id": 3,
      "name": "Bargello ERKEK ....",
      "image": "https://cdn.dsmcdn.com/ty96/....jpg",
      "url": "https://www.trendyol.com/bargello/erkek-par..."
   },
   "alarm": null
}
```

Watchlist

- int id
- Product product
- Alarm alarm
- User user

Product

- int id
- String name
- String image
- String url
- List<Watchlist> watchlists
- List<Price> prices

WATCHLIST CONTROLLER

deleteWatchlist - POST

Es nimmt einen Parameter watchlist_id und erstellt ein neues Watchlist-Objekt mit der angegebenen ID. Anschließend ruft es die Methode deleteWatchlist des Objekts watchlistService auf und übergibt das erstellte Watchlist-Objekt als Parameter, um die Watchlist aus dem System zu löschen. Schließlich gibt es eine Antwort mit dem HTTP-Statuscode „OK“ (200) und der Meldung „Produkt gelöscht“ zurück. die das erfolgreiche Löschen der Watchlist anzeigt. Die Antwort wird in ein ResponseEntity-Objekt eingeschlossen.

Antwort

Product deleted.

Watchlist

- int id
- Product product
- Alarm alarm
- User user

Product

- int id
- String name
- String image
- String url
- List<Watchlist> watchlists
- List<Price> prices

WATCHLIST CONTROLLER

setAlarm - POST

Es nimmt ein Alarm-Objekt als Anforderungstext, der eine ID einer Beobachtungsliste und eine Zeit zum Einstellen eines Alarms für diese Beobachtungsliste enthält. Anschließend extrahiert es die watchlist_id aus dem Alarm-Objekt und ruft die findById-Methode des watchlistService-Objekts auf, um das Watchlist-Objekt mit der angegebenen ID zu erhalten. Anschließend ruft es die setAlarm-Methode des watchlistService-Objekts auf und übergibt das abgerufene Watchlist-Objekt und das Alarm-Objekt als Parameter, um einen Alarm für die angegebene Watchlist festzulegen. Schließlich gibt es eine Antwort mit einem HTTP-Statuscode von OK (200) und einer Meldung "Alarm gesetzt!" die die erfolgreiche Einstellung des Alarms anzeigt. Die Antwort wird in ein ResponseEntity-Objekt eingeschlossen.

Request body

```
{
  "productId": 1,
  "watchlist_id": 1,
  "date_created": "2023-04-28",
  "condition": "BELOW_TARGET",
  "target_price": 150.0
}
```

Antwort

Alarm set!

Watchlist

- int id
- Product product
- Alarm alarm
- User user

Product

- int id
- String name
- String image
- String url
- List<Watchlist> watchlists
- List<Price> prices

WATCHLIST CONTROLLER

deleteAlarm - POST

Es nimmt einen watchlist_id-Parameter und verwendet die findById-Methode des watchlistService-Objekts, um das Watchlist-Objekt mit der angegebenen ID zu finden. Anschließend ruft es die Methode deleteAlarm des Objekts watchlistService auf und übergibt das abgerufene Watchlist-Objekt als Parameter, um den mit der angegebenen Watchlist verknüpften Alarm zu löschen. Schließlich gibt es eine Antwort mit einem HTTP-Statuscode von OK (200) und einer Meldung „Alarm gelöscht!“ zurück. die das erfolgreiche Löschen des Alarms anzeigt. Die Antwort wird in ein ResponseEntity-Objekt eingeschlossen.

Antwort

Alarm deleted!

Watchlist

- int id
- Product product
- Alarm alarm
- User user

Product

- int id
- String name
- String image
- String url
- List<Watchlist> watchlists
- List<Price> prices

ALARM CONTROLLER

loadAlarm - GET

Es nimmt einen id-Parameter und verwendet die findAlarmById-Methode des alarmService-Objekts, um das Alarm-Objekt mit der angegebenen ID zu finden. Wenn das Alarm-Objekt gefunden wird, gibt es eine Antwort mit dem HTTP-Statuscode OK (200) und dem Alarm-Objekt im Antworttext zurück, eingeschlossen in ein ResponseEntity-Objekt. Wenn das Alarm-Objekt nicht gefunden wird, gibt es eine Antwort mit dem HTTP-Statuscode NOT_FOUND (404) zurück, der anzeigt, dass die angeforderte Ressource nicht gefunden wurde, indem es die notFound-Methode der ResponseEntity-Klasse aufruft. Der Antworttext ist leer und die Antwort ist außerdem in ein ResponseEntity-Objekt eingeschlossen.

Antwort


```
{
   "id": 12,
   "productId": 1,
   "watchlist_id": 1,
   "date_created": "2023-04-28",
   "condition": "BELOW_TARGET",
   "target_price": 150.0,
   "date_triggered": null
}
```


Alarm

- int id
- int productId
- int watchlist_id
- LocalDate date_created
- String condition
- double target_price
- LocalDate date_triggered

PRICE CONTROLLER

listPrices - GET

Es nimmt einen id-Parameter und verwendet die findAlarmById-Methode des alarmService-Objekts, um das Alarm-Objekt mit der angegebenen ID zu finden. Wenn das Alarm-Objekt gefunden wird, gibt es eine Antwort mit dem HTTP-Statuscode OK (200) und dem Alarm-Objekt im Antworttext zurück. Wenn das Alarm-Objekt nicht gefunden wird, gibt es eine Antwort mit dem HTTP-Statuscode NOT_FOUND zurück. Es nimmt einen product_id-Parameter und verwendet die getPricesForLast7Days-Methode des priceService-Objekts, um eine Liste von Price-Objekten für das angegebene Produkt für die letzten 7 Tage abzurufen. Er gibt dann eine Antwort mit einem HTTP-Statuscode von FOUND (302) zurück.

Antwort

```
[
  {
    "id": 35,
    "date": "2023-05-04",
    "price": 137.0
  },
  {
    "id": 34,
    "date": "2023-05-03",
    "price": 137.0
  }, ...
]
```

Price

- int id
- LocalDate date
- double price
- Product product

PRICE CONTROLLER

startChecking - POST

Es ruft die updatePrices-Methode des priceService-Objekts auf, bei der es sich um eine periodische Methode handelt, die mit der regelmäßigen Überprüfung von Preisen und Alarmen beginnt. Die Methode akzeptiert keine Parameter und löst einfach die updatePrices-Methode aus, um mit der Preisprüfung zu beginnen. Es gibt dann eine Antwort mit einem HTTP-Statuscode von OK (200) und einer Nachricht „Gestartet“ zurück, die anzeigt, dass der Preisprüfungsprozess gestartet wurde. Der Antworttext enthält die Nachricht und wird mithilfe der ok- und body-Methoden der ResponseEntity-Klasse in ein ResponseEntity-Objekt eingeschlossen.

Price

- int id
- LocalDate date
- double price
- Product product

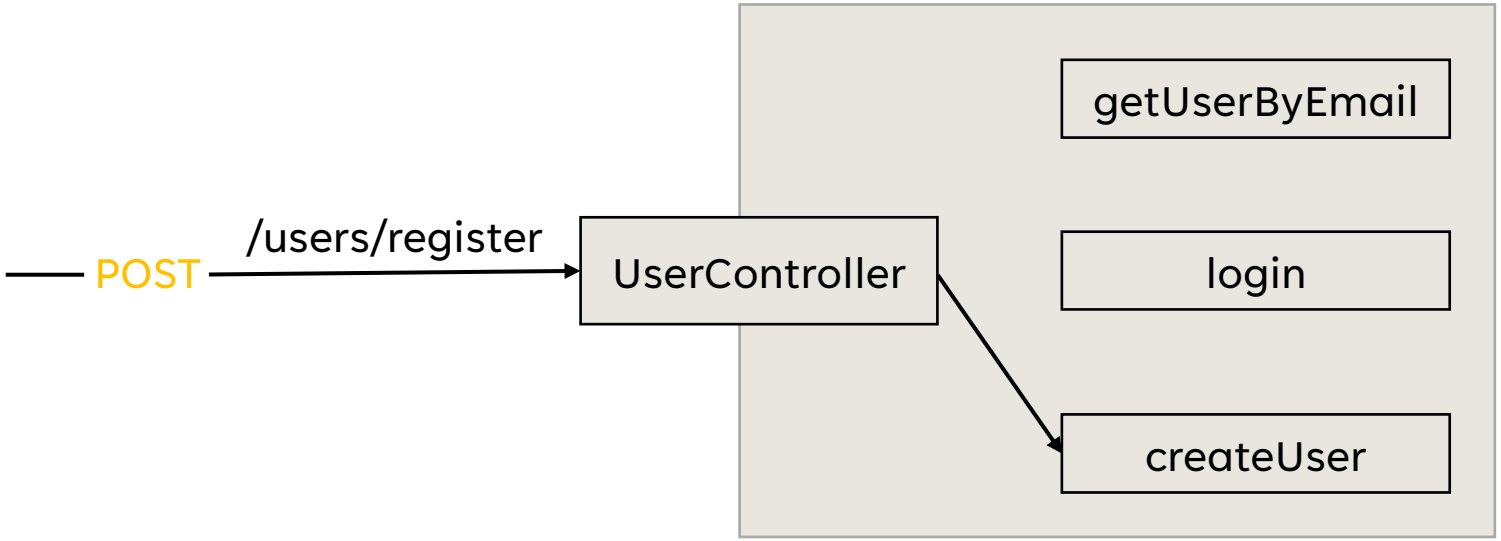
Antwort

Started

RÜCKVERFOLGBARKEIT DER ANFORDERUNGEN

Use Case Registrierung

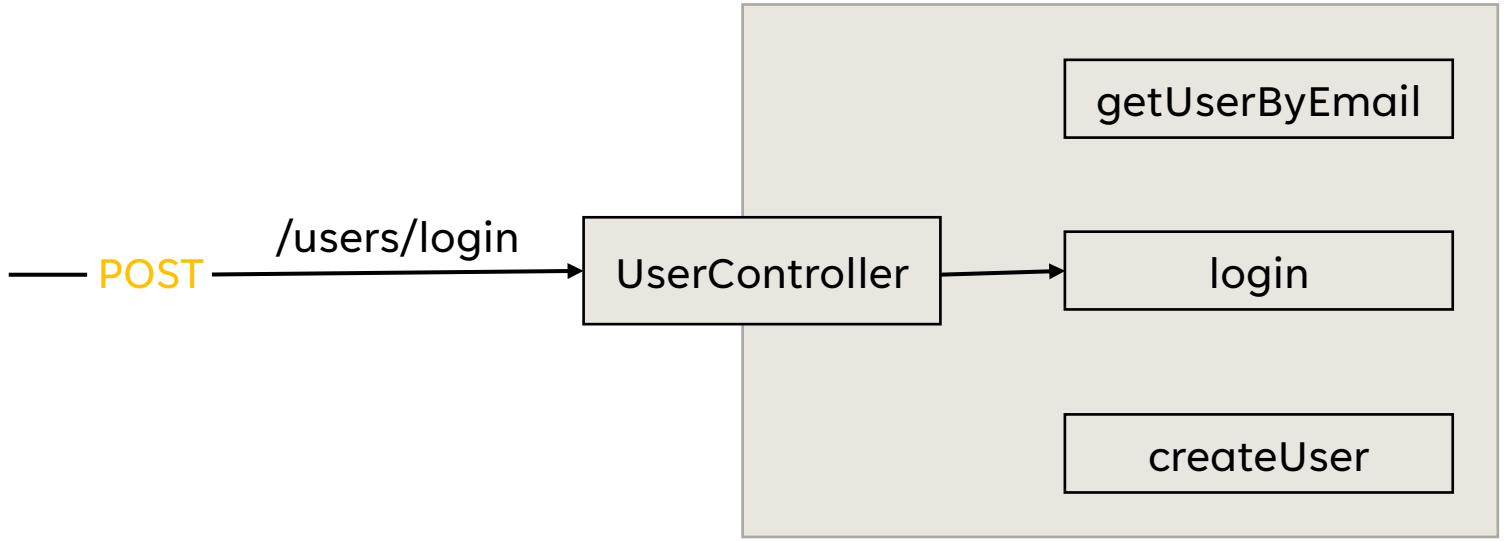
Funktion: Registrierung
Eingaben: Vorname, Nachname, E-Mail-Adresse, Passwort
Verarbeitungsschritte:
• Prüfe, ob die E-Mail-Adresse bereits in der Datenbank vorhanden ist.
• Wenn nicht, wird ein neuer Benutzer mit den eingegebenen Daten in der Datenbank erstellt.
• Der Benutzer wird zur Login-Seite weitergeleitet.
• Falls die E-Mail-Adresse bereits vorhanden ist, wird eine Fehlermeldung ausgegeben.
Ausgaben: Der Benutzer wird zur Login-Seite weitergeleitet oder eine Fehlermeldung wird angezeigt.



RÜCKVERFOLGBARKEIT DER ANFORDERUNGEN

Use Case Login

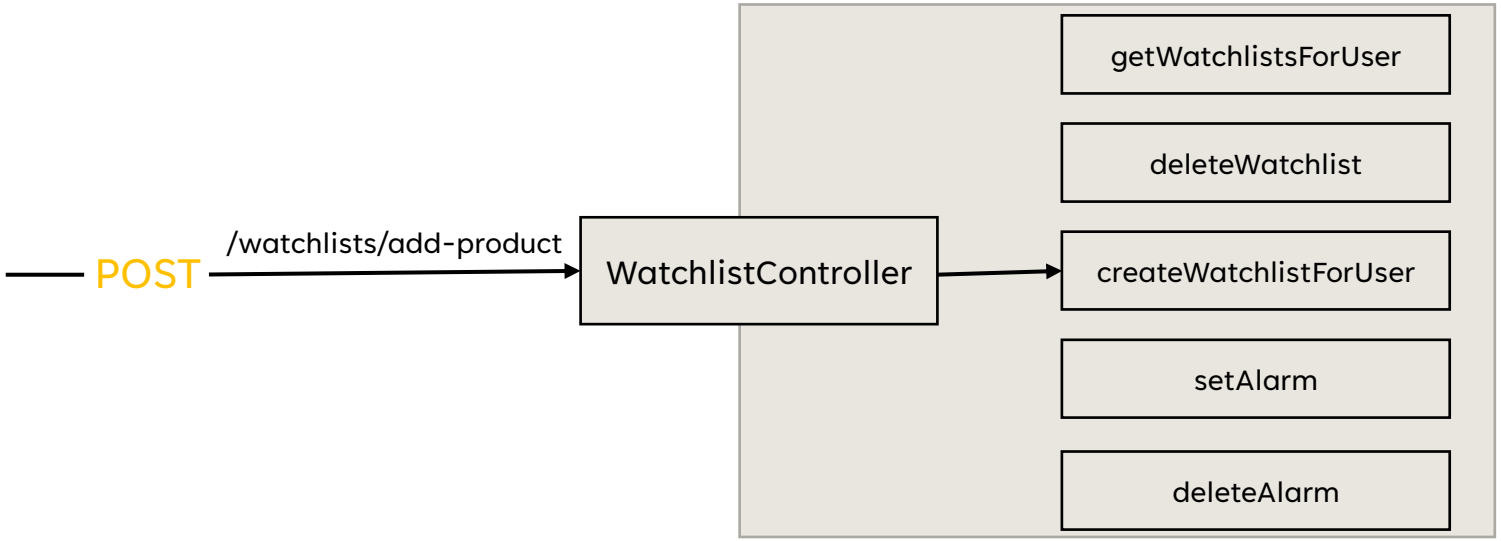
Funktion: Login
Eingaben: E-Mail-Adresse, Passwort
Verarbeitungsschritte:
• Prüfe, ob die E-Mail-Adresse und das Passwort in der Datenbank vorhanden sind.
• Wenn ja, wird der Benutzer zur Hauptseite weitergeleitet.
• Falls die E-Mail-Adresse oder das Passwort falsch sind, wird eine Fehlermeldung ausgegeben.
Ausgaben: Der Benutzer wird zur Hauptseite weitergeleitet oder eine Fehlermeldung wird angezeigt.



RÜCKVERFOLGBARKEIT DER ANFORDERUNGEN

Use Case Produkt hinzufügen

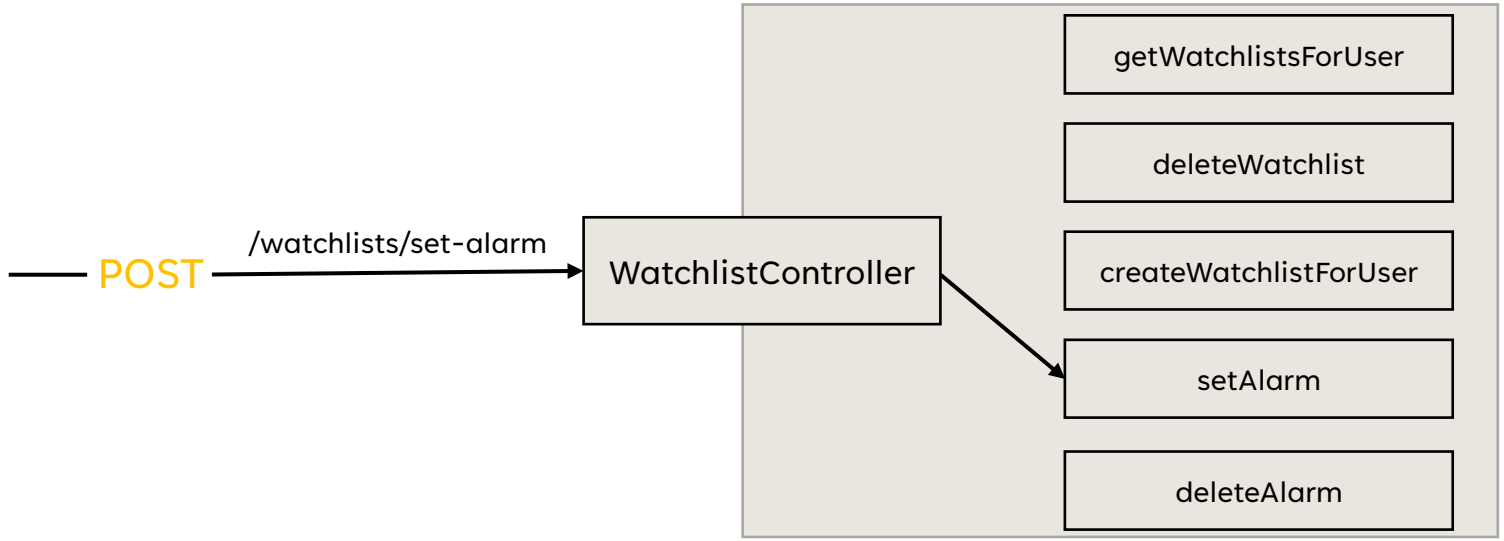
Funktion: Produkt hinzufügen
Eingaben: URL einer beliebten Einkaufsseite
Verarbeitungsschritte:
• Prüfe, ob die URL bereits in der Datenbank vorhanden ist.
• Wenn nicht, wird ein neues Produkt mit den Daten aus der URL in der Datenbank erstellt.
• Das Produkt wird zur Watchlist des Benutzers hinzugefügt.
Ausgaben: Das Produkt wird zur Watchlist des Benutzers hinzugefügt.



RÜCKVERFOLGBARKEIT DER ANFORDERUNGEN

Use Case Erstelle Alarm

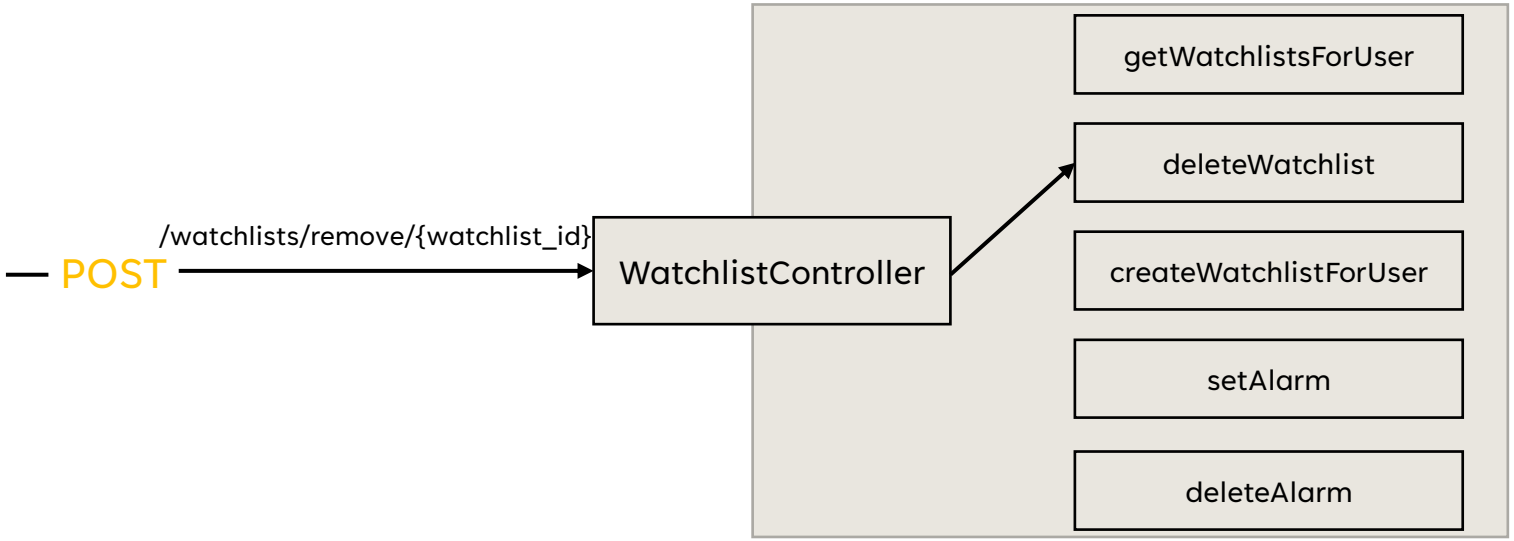
Funktion: Erstelle Alarm
Eingaben: Benutzer-ID, Produkt-ID, Bedingung und Ziel-Preis
Verarbeitungsschritte:
• Überprüfe, ob der Benutzer bereits einen Alarm für das Produkt und die Bedingung erstellt hat
• Füge einen neuen Alarm hinzu, falls keine Übereinstimmungen gefunden wurden, und speichere die Alarmdaten in der Datenbank
Ausgaben: Bestätigungsmeldung, dass der Alarm erfolgreich erstellt wurde, oder Fehlermeldung, wenn der Benutzer bereits einen Alarm für das Produkt und die Bedingung hat oder der Benutzer oder das Produkt nicht existieren.



RÜCKVERFOLGBARKEIT DER ANFORDERUNGEN

Use Case Entferne Produkt aus der Watchlist

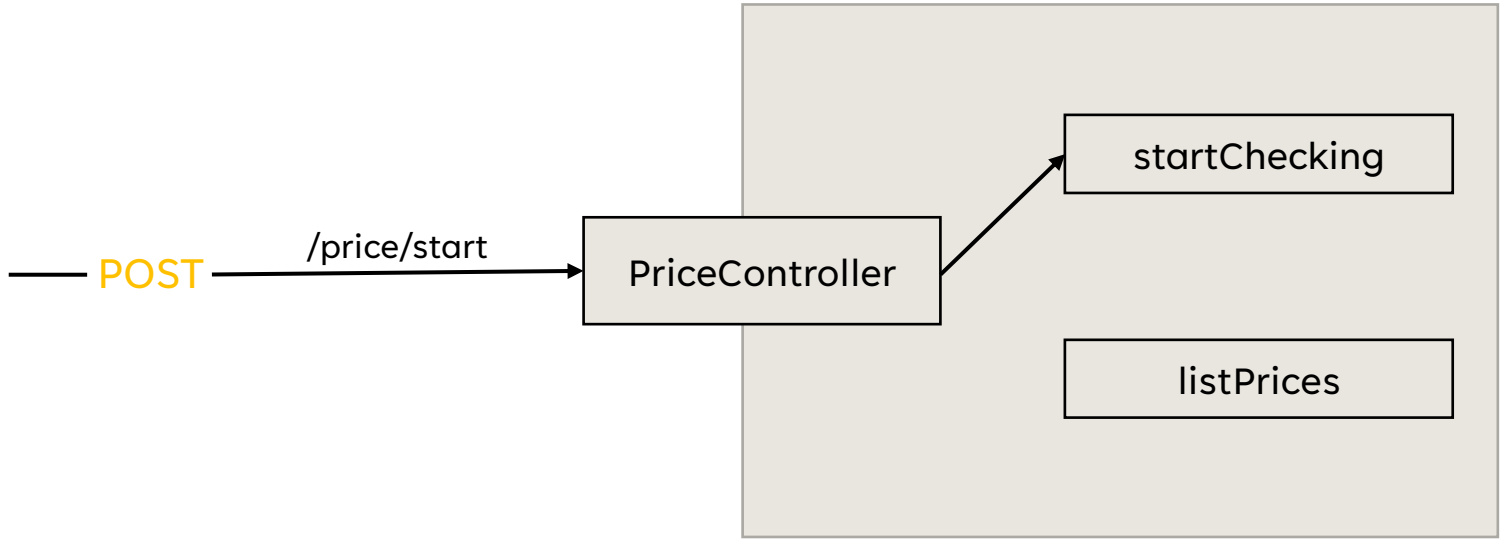
Funktion: Entferne Produkt aus der Watchlist
Eingaben: Benutzer-ID und Produkt-ID
Verarbeitungsschritte:
• Überprüfe, ob der Benutzer existiert und das Produkt auf seiner Watchlist hat
• Entferne das Produkt aus der Watchlist des Benutzers
Ausgaben: Bestätigungsmeldung, dass das Produkt erfolgreich aus der Watchlist entfernt wurde, oder Fehlermeldung, wenn der Benutzer das



RÜCKVERFOLGBARKEIT DER ANFORDERUNGEN

Use Case Alarm festlegen

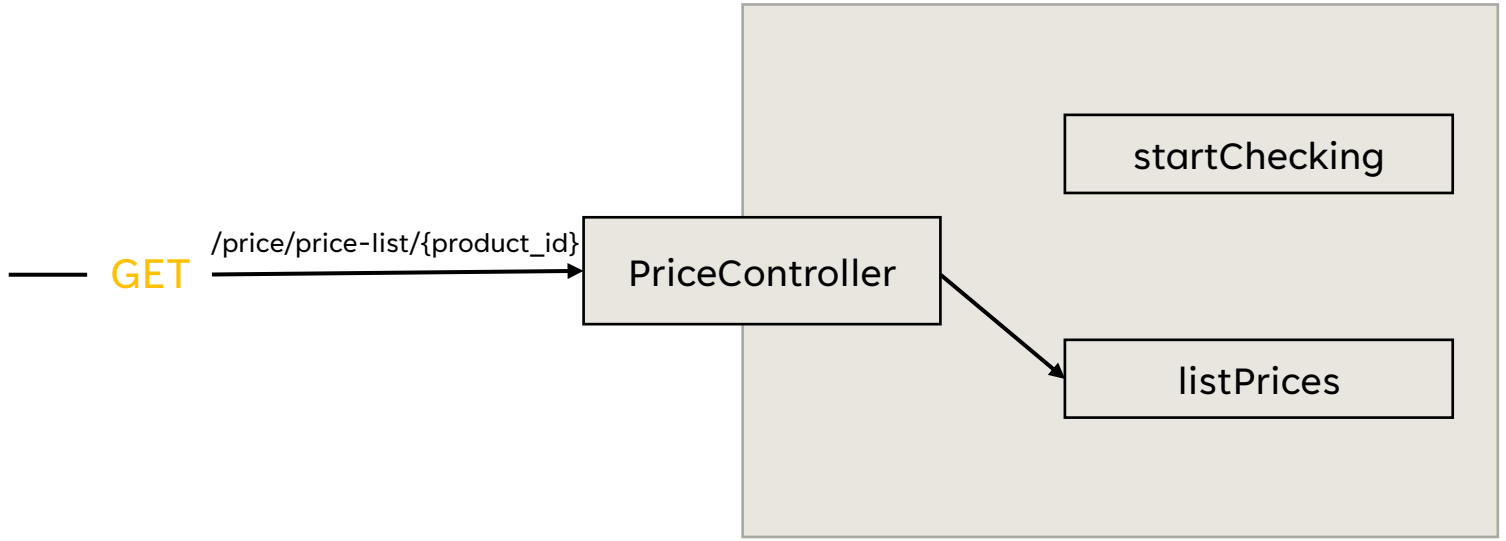
Funktion: Alarm festlegen
Eingabe: Benutzer-ID, Produkt-ID, Preislimit
Verarbeitungsschritte:
• Der Alarm wird für das ausgewählte Produkt mit dem angegebenen Preislimit in der Datenbank gespeichert.
• Das System überprüft regelmäßig den Preis des Produkts und sendet eine E-Mail an den Benutzer, wenn der Preis unter das Limit fällt.
Ausgaben: Der Alarm wird erfolgreich festgelegt und der Benutzer wird per E-Mail benachrichtigt, wenn das Preislimit unterschritten wird.



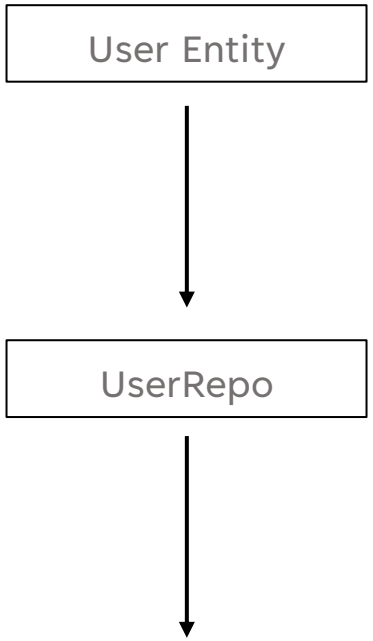
RÜCKVERFOLGBARKEIT DER ANFORDERUNGEN

Use Case Preisverlauf anzeigen

Funktion: Preisverlauf anzeigen
Eingaben: Produkt aus der Watchlist
Verarbeitungsschritte:
• Wenn der Benutzer keine Watchlist hat, werden die in der Datenbank registrierten Opportunity-Produkte angezeigt.
• Wenn der Benutzer eine Watchlist hat Preisdaten des ausgewählten Produkts werden aus der Datenbank abgerufen.
• Die Preisentwicklung wird in einem grafischen Format angezeigt.
Ausgaben: Der Preisverlauf des ausgewählten Produkts wird angezeigt.



USER REPOSITORY

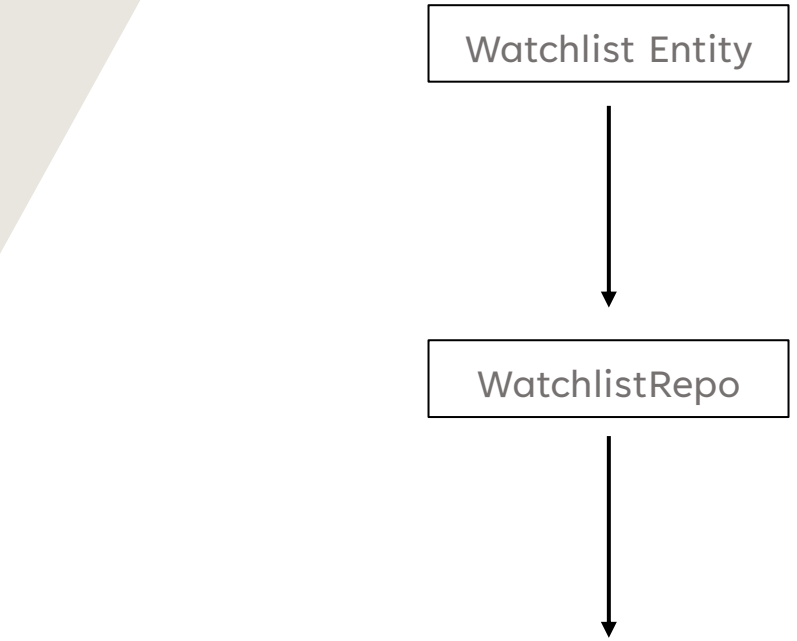


User

- int id
- String email
- String name
- String surname
- String password
- List<Watchlist> watchlists

id	email	name	password	surname
1	james.bond@g...	James	$2a$10$wHH...	Bond
2	anakin@gm...	Anakin	$2a$10$saSf...	Skywalker
3	jack.dowson@g...	Jack	$2a$10$86i...	Dowson
4	rose@gma...	Rose	$2a$10$e27...	Bukater

WATCHLIST REPOSITORY

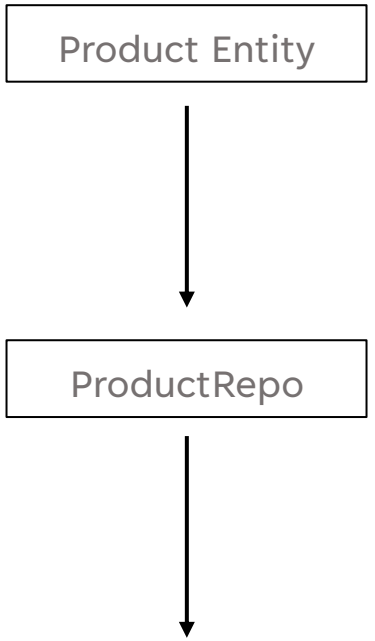


Watchlist

- int id
- Product product
- Alarm alarm
- User user

id	alarm_id	product_id	user_id
1	1	1	2
2	4	3	5
3	null	2	1
4	null	2	8

PRODUCT REPOSITORY

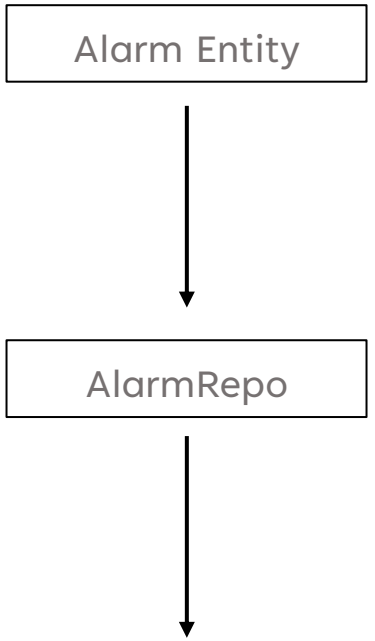


Product

- int id
- String name
- String image
- String url
- List<Watchlist> watchlists
- List<Price> prices

id	image	name	url
1	https://productimages.hepsiburad...	IPhone 13...	https://www.hepsibu...
2	https://m.media-amazon.com/ima...	Xiaomi Redmi...	https://www.amazo...
3	https://cdn.dsmcdn.com/ty96/prod...	Nike Air Ma...	https://www.trendyol...
4	https://productimages.hepsiburad...	Karamazov Kar...	https://www.hepsibu...

ALARM REPOSITORY

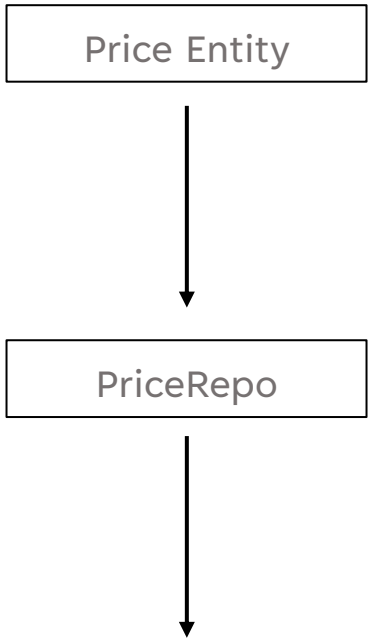


Alarm

- int id
- int productId
- int watchlist_id
- LocalDate date_created
- String condition
- double target_price
- LocalDate date_triggered

id	condit	date_created	date_triggered	product_id	target_price	watchlist_id
1	BELOW_PRICE	2023-04-28	2023-04-30	2	150.0	1
2	EQUALS_TARGET	2023-04-29	null	3	2000.0	3
3	ANY_CHANGE	2023-04-30	2023-05-02	5	3444.0	2
4	ABOVE_TARGET	2023-05-01	null	1	20.0	7

PRICE REPOSITORY



Price

- int id
- LocalDate date
- double price
- Product product

id	date	price	product_id
1	2023-04-28	200	1
2	2023-04-29	200	1
3	2023-04-28	1400	2
4	2023-04-29	1300	2