myInventory

Modul: Backend-Entwicklung

Marvin Kotzian

AGENDA

Knowledge Foundation
@ Reutlingen University

- Problemstellung
- Funktionen
- Technische Highlights
- Architektur / Projektstruktur
- API / Live Demo
- Fazit / Ausblick

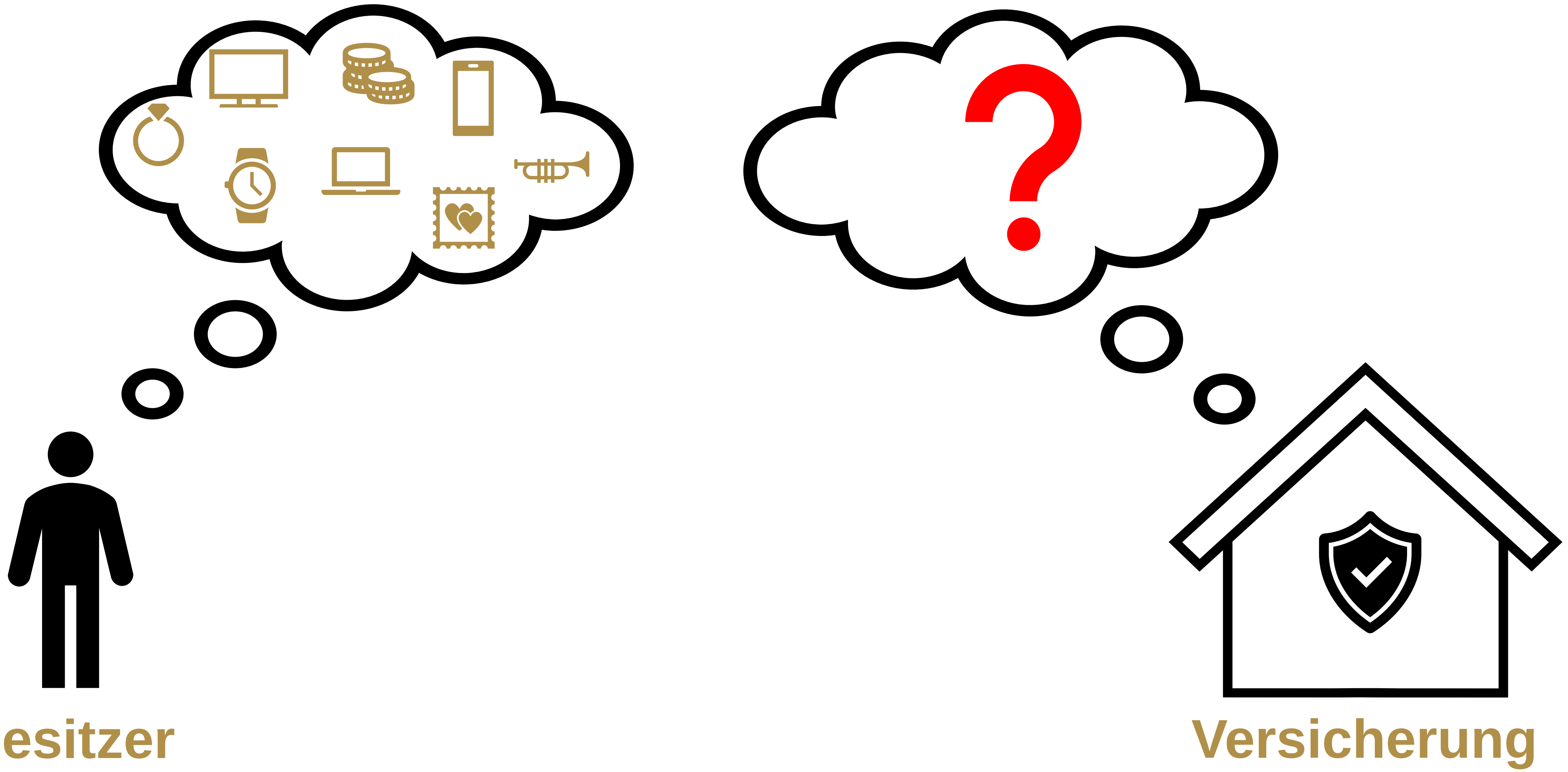
PROBLEMSTELLUNG

Knowledge Foundation
@ Reutlingen University

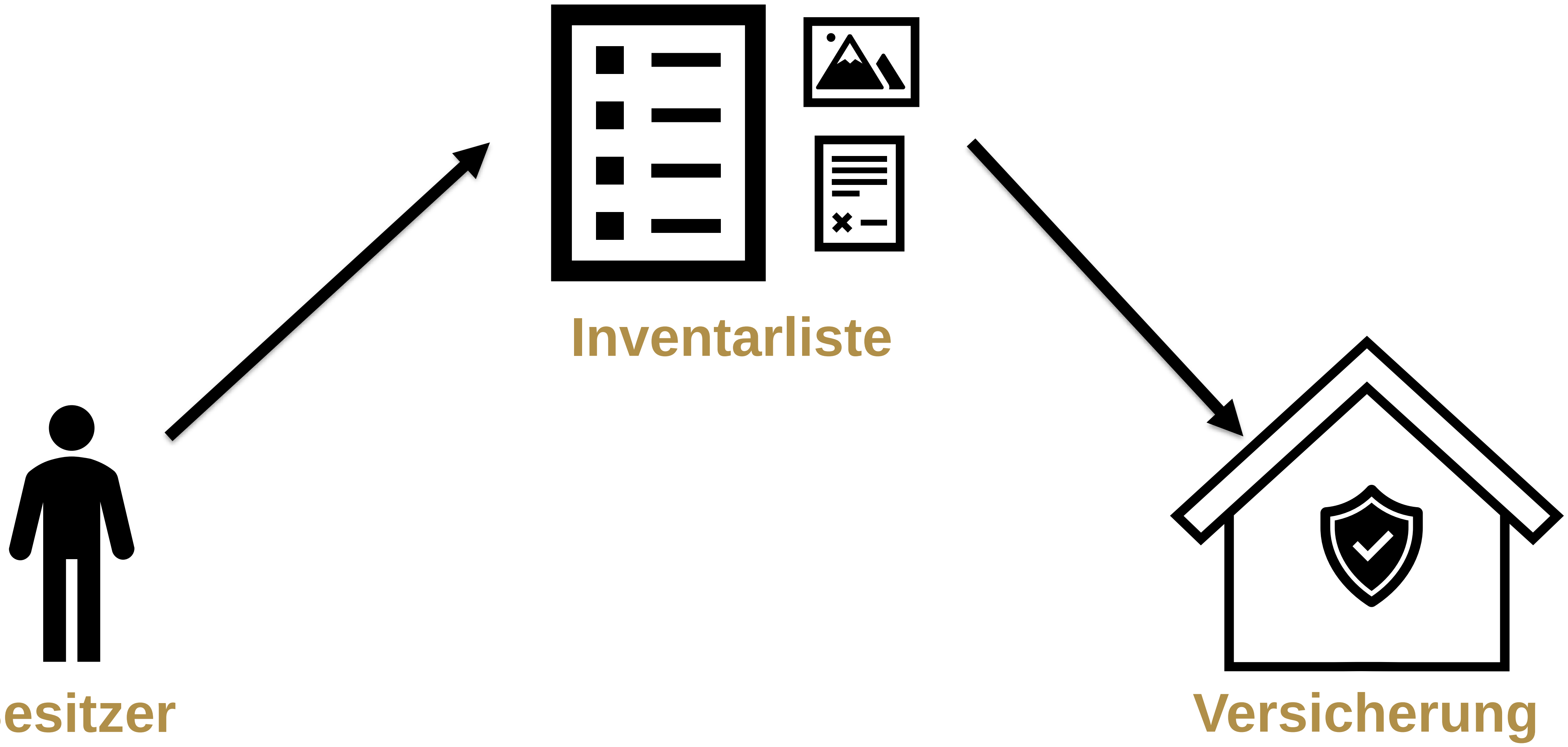
PROBLEMSTELLUNG

Versicherung

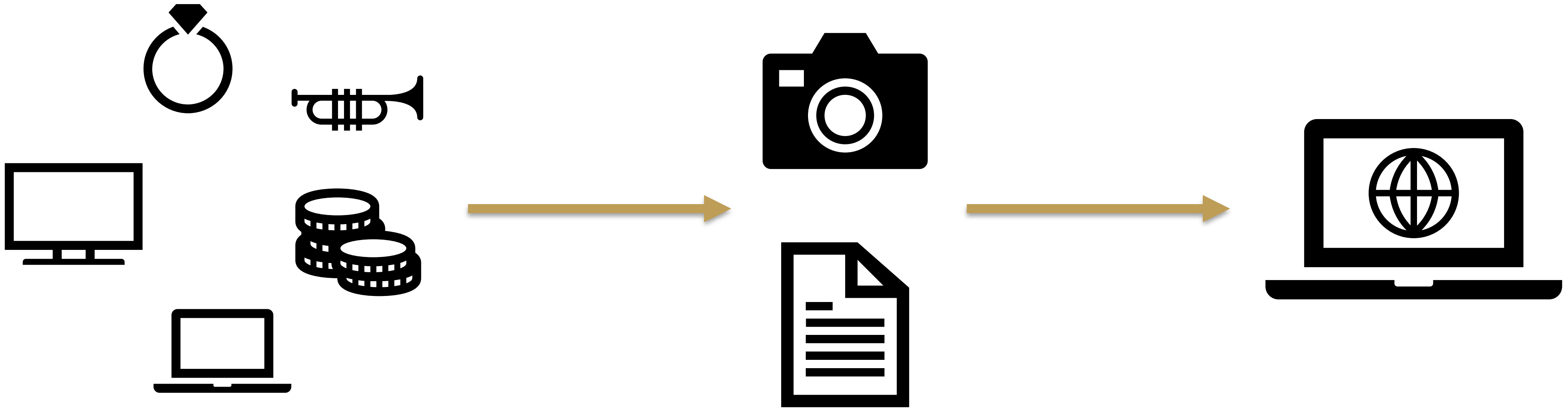
PROBLEMSTELLUNG



PROBLEMSTELLUNG



LÖSUNG – myInventory



Besitz

Foto / Rechnung / weitere Infos

Upload zu myInventory

myInventory – FUNKTIONEN

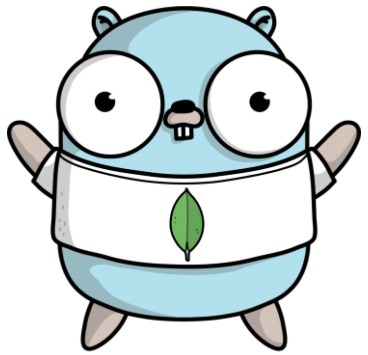
- Benutzerauthentifizierung
  - Registrierung
  - Anmeldung
- Inventarisierung (Infos, Bilder, Rechnungen)
  - Inventar erfassen
  - Inventar auslesen
  - Inventar editieren
  - Inventar löschen

myInventory – Technologien

Gin Web Framework

Web Framework

https://github.com/gin-gonic/gin



MongoDB Go Driver

MongoDB Bibliothek für GO

https://github.com/mongodb/mongo-go-driver

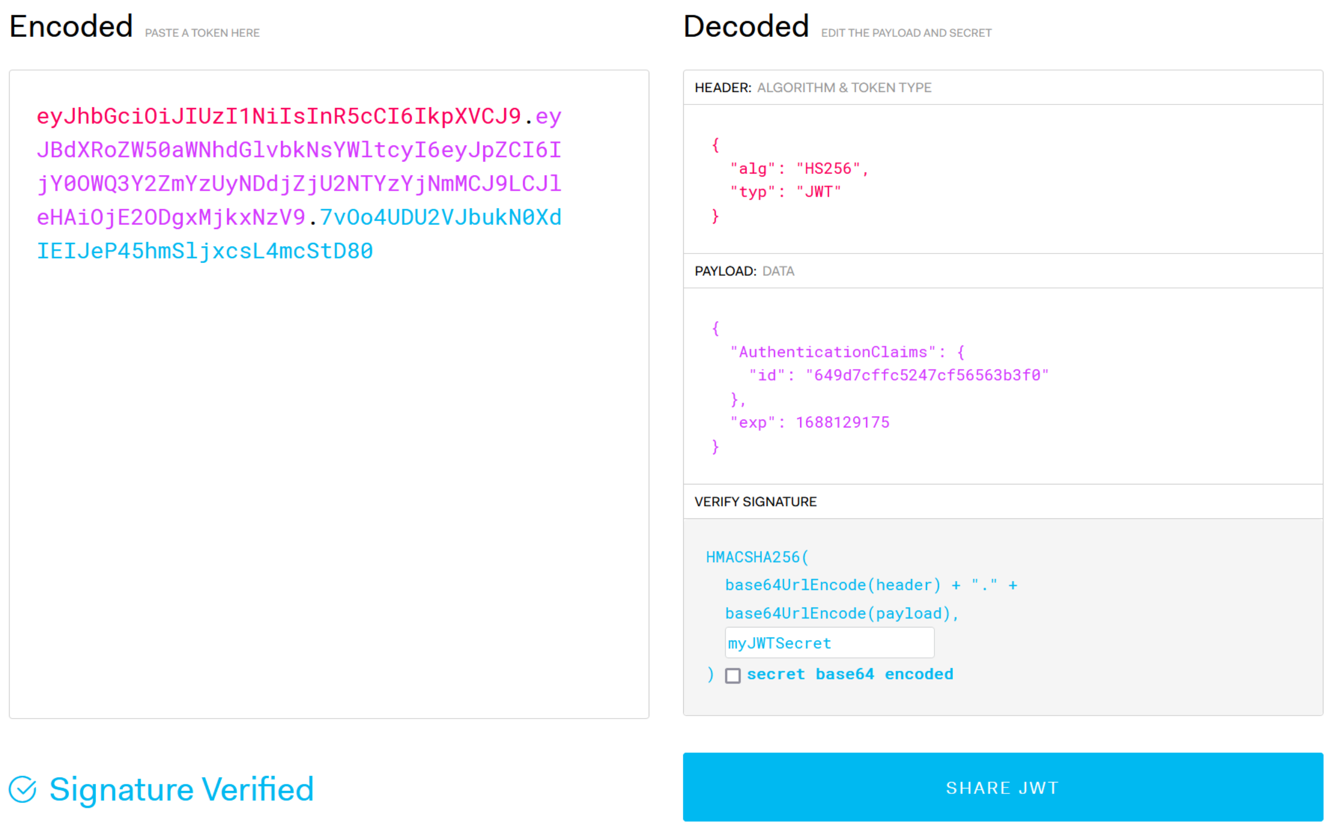
myInventory – Technologien

Knowledge Foundation
@ Reutlingen University

jwt-go

JWT Token Bibliothek für Go

https://github.com/golang-jwt/jwt



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JBdXRoZW50aWNhdGlvbkNsYWltcyI6eyJpZCI6I
jY0OWQ3Y2ZmYzUyNDdjZjU2NTYzYjNmMCJ9LCJl
eHAiOjE2ODgxMjkxNzV9.7vOo4UDU2VJbukN0Xd
IEIJeP45hmSljxcsL4mcStD80
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "AuthenticationClaims": {
    "id": "649d7cffc5247cf56563b3f0"
  },
  "exp": 1688129175
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  myJWTSecret
) ☐ secret base64 encoded
```

Signature Verified

SHARE JWT

myInventory – Datenmodell

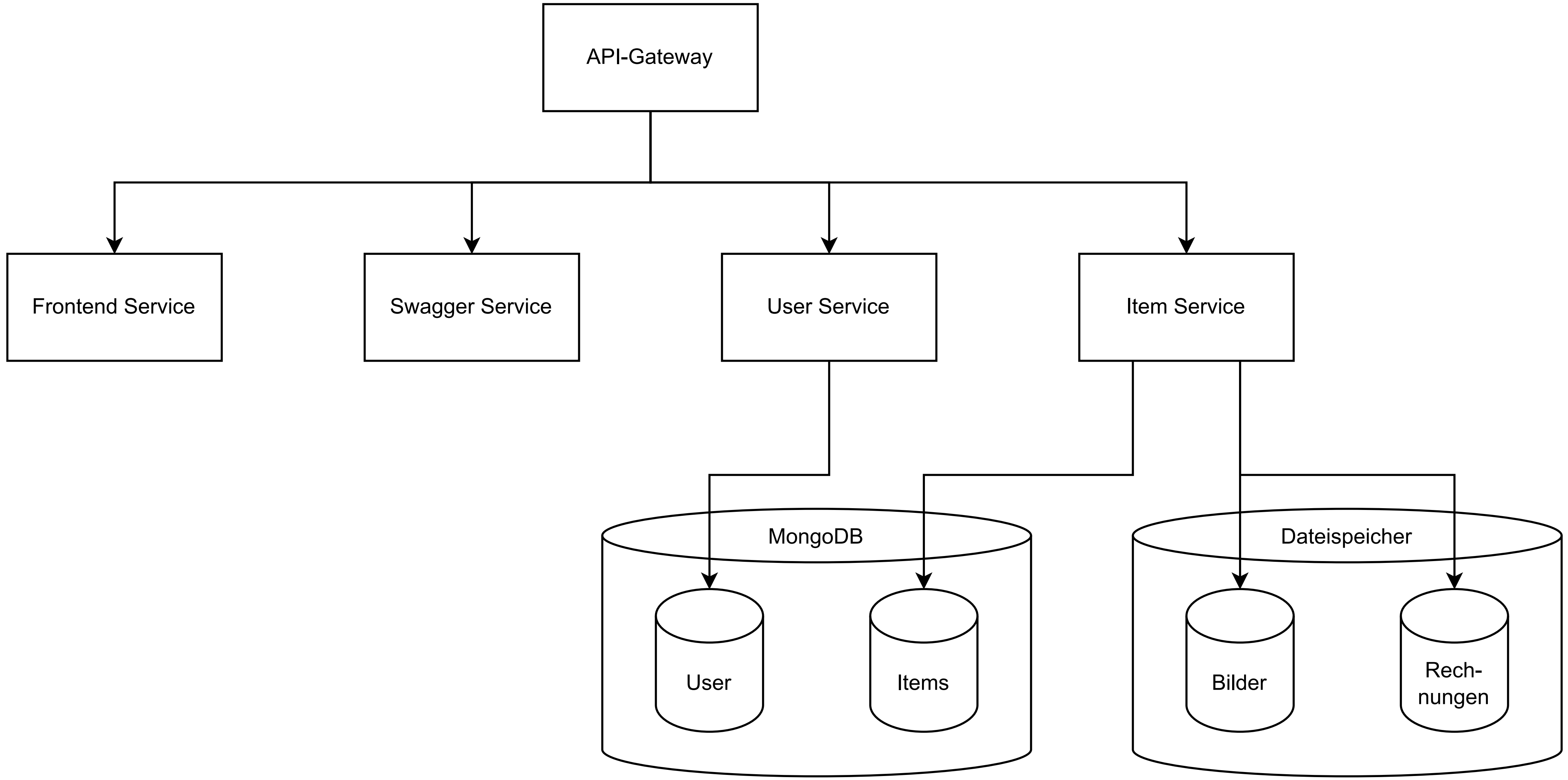
```
type User struct {
    ID        string
    FirstName string
    LastName  string
    Email     string
    Password  string
}
```

User Modell

```
type File struct {
    ID       string
    FileName string
}

type Images struct {
    PreviewImageID string
    Images         []File
}

type PurchaseInfo struct {
    Price float64
    Date      time.Time
    Place     string
    Quantity  int
    Invoice   File
}

type Item struct {
    ID           string
    Name         string
    Description  string
    PurchaseInfo PurchaseInfo
    Images       Images
    OwnerID      string
}
```

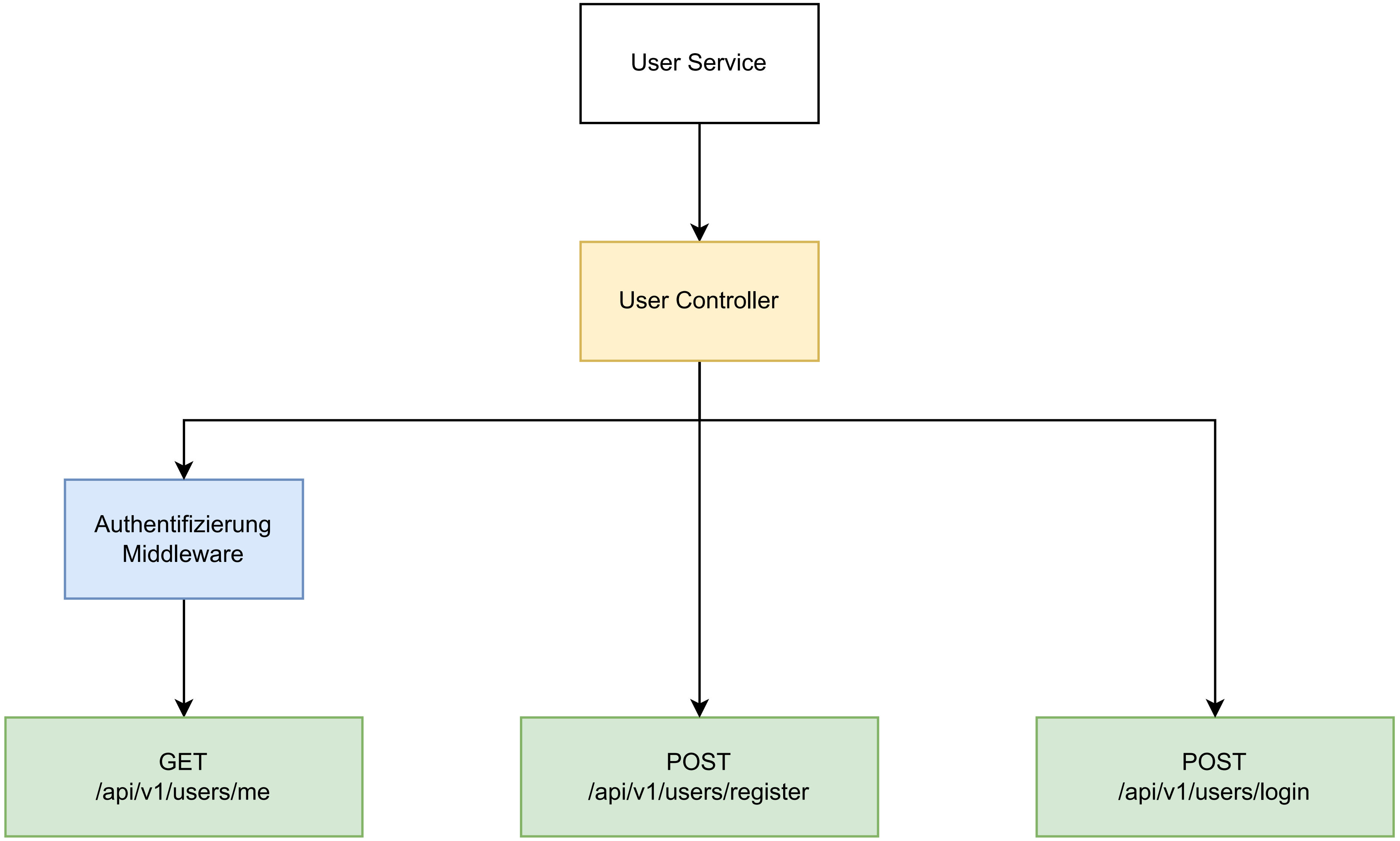
Item Modell

myInventory – ARCHITEKTUR



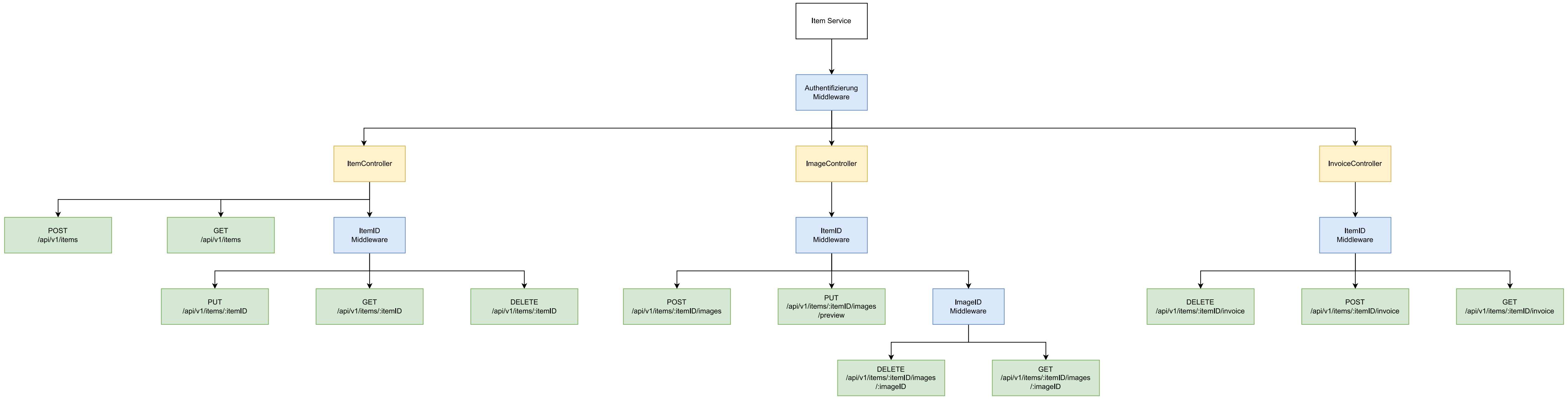
myInventory – USER SERVICE ARCHITEKTUR

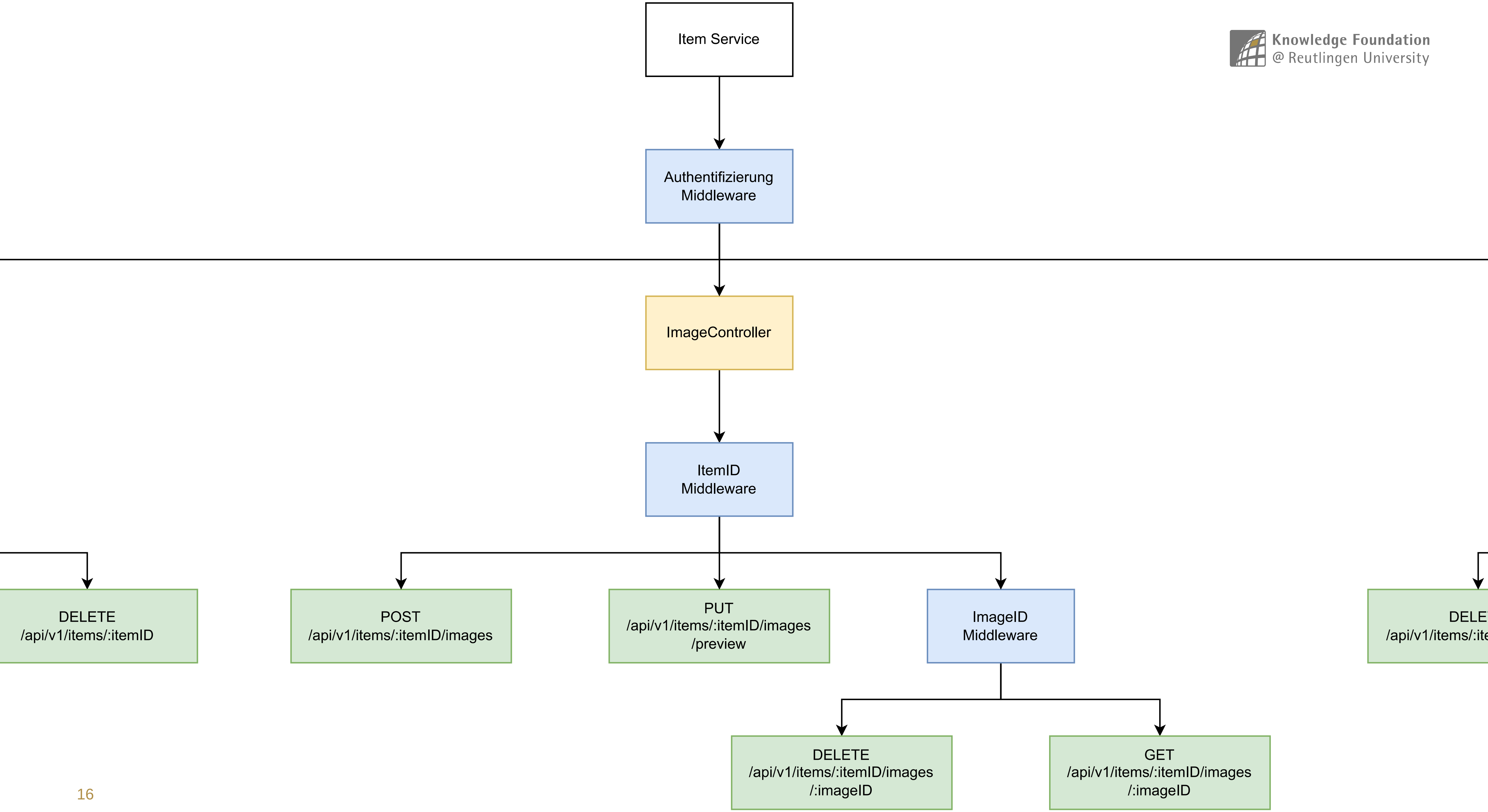
Knowledge Foundation
@ Reutlingen University

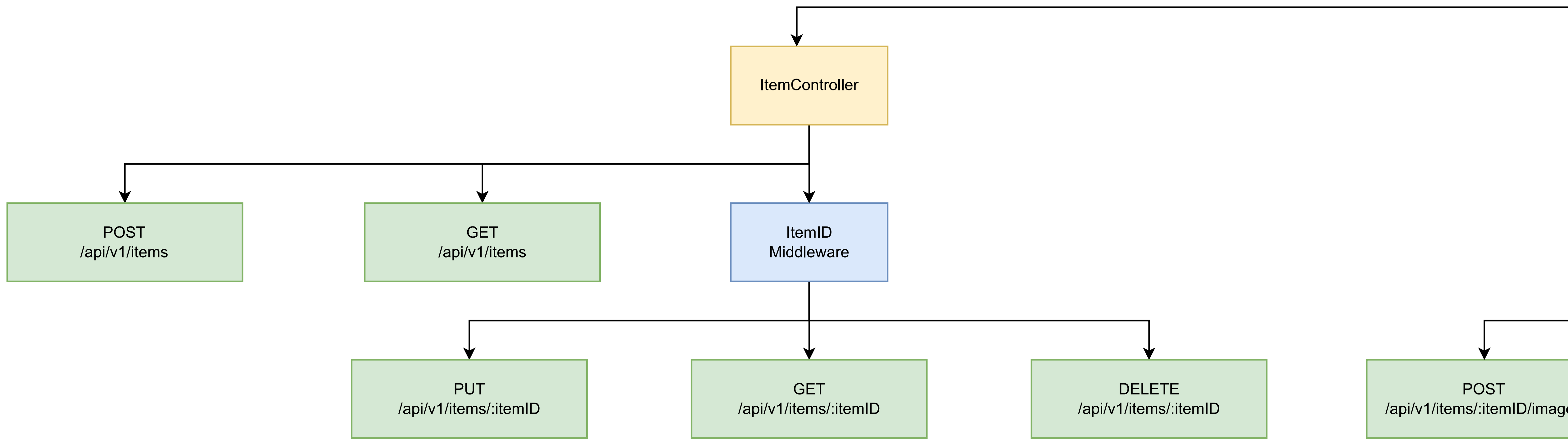


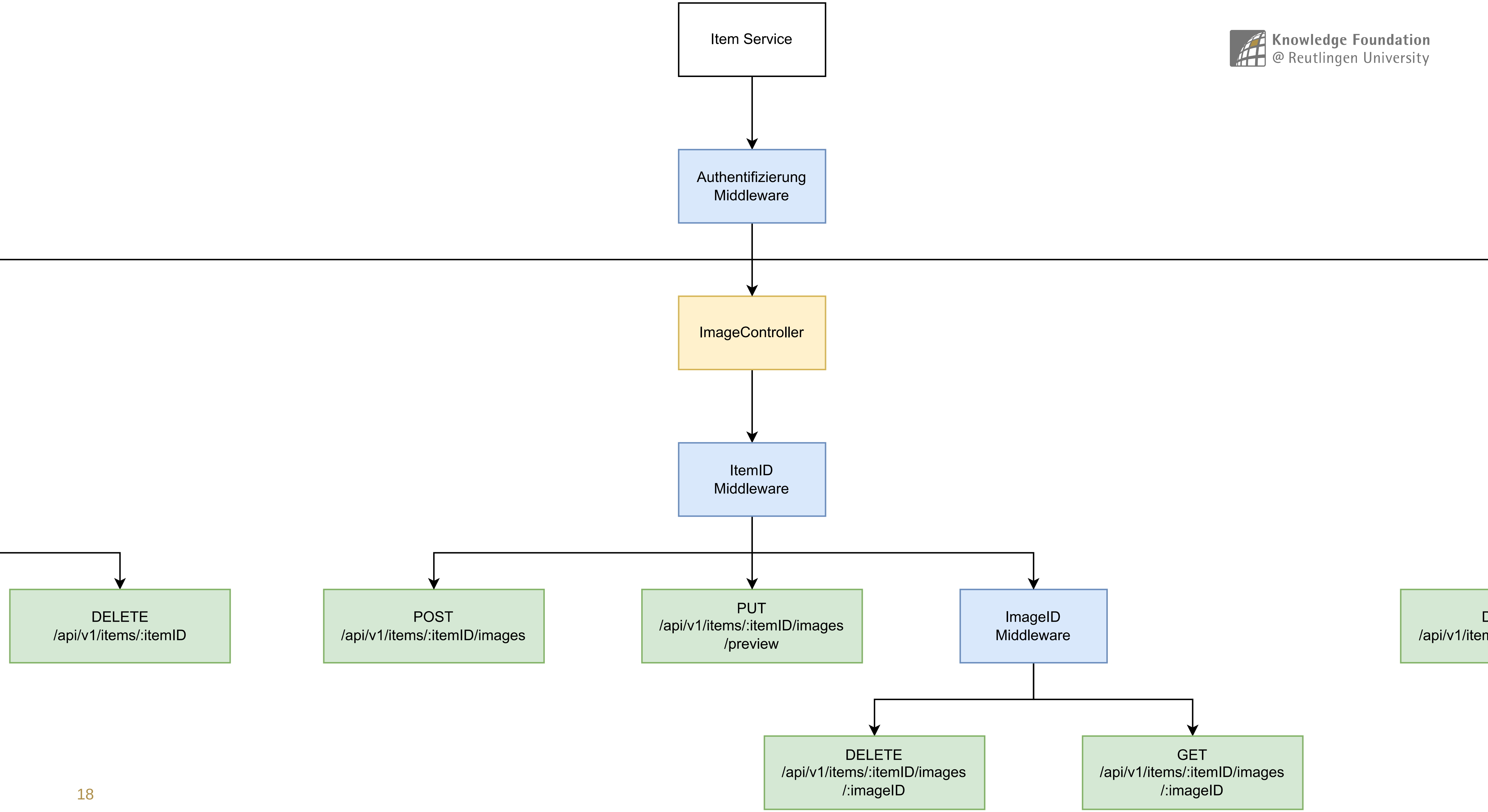
Knowledge Foundation
@ Reutlingen University

myInventory – ITEM SERVICE ARCHITEKTUR

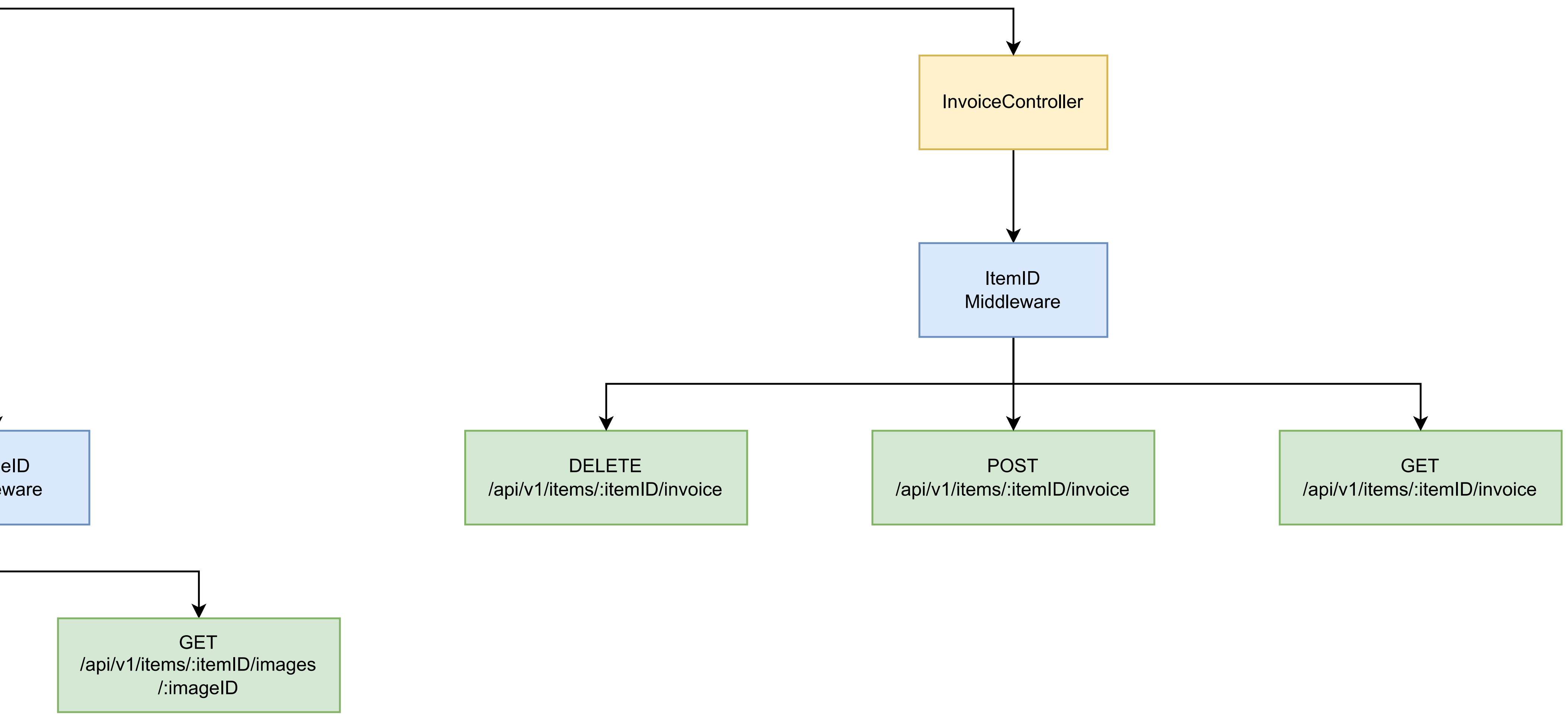

Item Service
Authentifizierung Middleware
ItemController
ImageController
InvoiceController
POST /api/v1/items
GET /api/v1/items
ItemID Middleware
ItemID Middleware
ItemID Middleware
PUT /api/v1/items/:itemID
GET /api/v1/items/:itemID
DELETE /api/v1/items/:itemID
POST /api/v1/items/:itemID/images
PUT /api/v1/items/:itemID/images /preview
ImageID Middleware
DELETE /api/v1/items/:itemID/invoice
POST /api/v1/items/:itemID/invoice
GET /api/v1/items/:itemID/invoice
DELETE /api/v1/items/:itemID/images /:imageID
GET /api/v1/items/:itemID/images /:imageID


Item Service
Authentifizierung Middleware
ImageController
ItemID Middleware
DELETE /api/v1/items/:itemID
POST /api/v1/items/:itemID/images
PUT /api/v1/items/:itemID/images /preview
ImageID Middleware
DELE /api/v1/items/:ite
DELETE /api/v1/items/:itemID/images /:imageID
GET /api/v1/items/:itemID/images /:imageID


ItemController
POST
/api/v1/items
GET
/api/v1/items
ItemID
Middleware
PUT
/api/v1/items/:itemID
GET
/api/v1/items/:itemID
DELETE
/api/v1/items/:itemID
POST
/api/v1/items/:itemID/imag

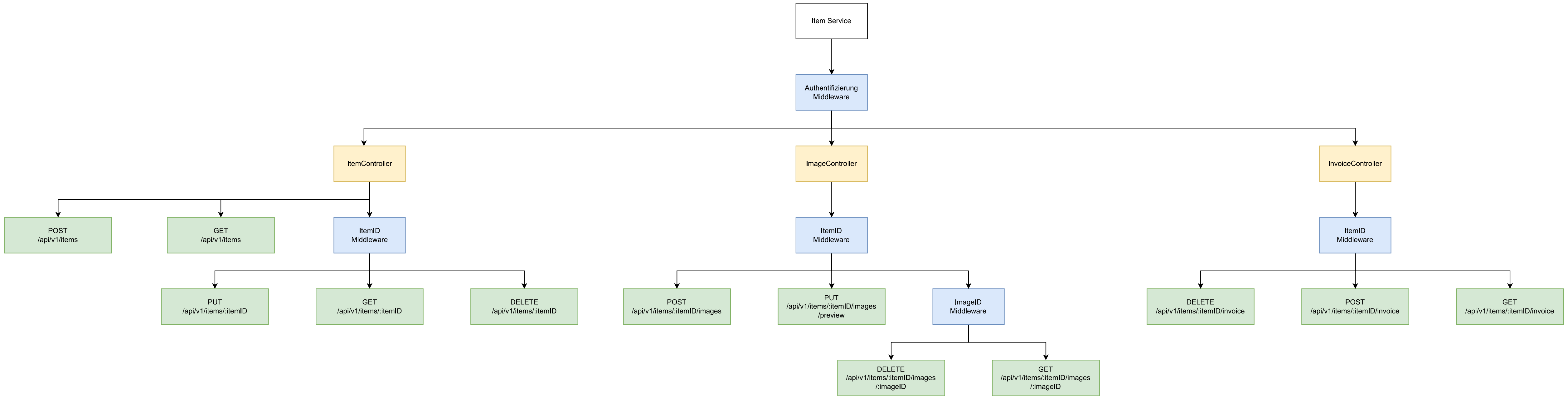

Item Service
Authentifizierung Middleware
ImageController
ItemID Middleware
DELETE /api/v1/items/:itemID
POST /api/v1/items/:itemID/images
PUT /api/v1/items/:itemID/images /preview
ImageID Middleware
D /api/v1/iten
DELETE /api/v1/items/:itemID/images /:imageID
GET /api/v1/items/:itemID/images /:imageID

Knowledge Foundation @ Reutlingen University


InvoiceController
ItemID Middleware
DELETE /api/v1/items/:itemID/invoice
POST /api/v1/items/:itemID/invoice
GET /api/v1/items/:itemID/invoice
eID
eware
GET /api/v1/items/:itemID/images /:imageID

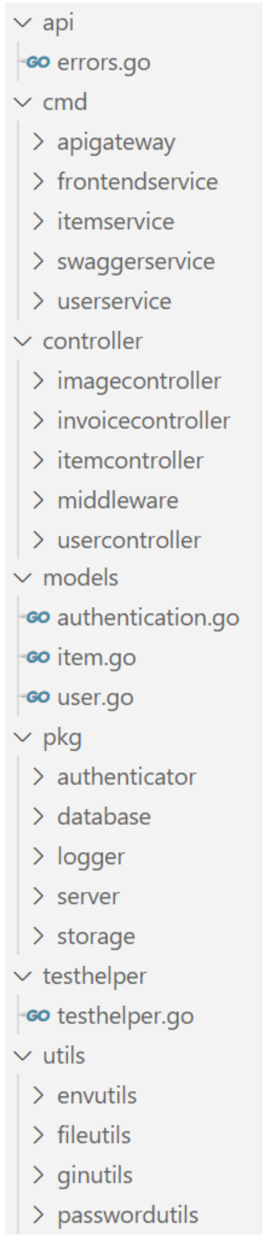
myInventory – ITEM SERVICE ARCHITEKTUR

Knowledge Foundation
@ Reutlingen University


Item Service
Authentifizierung Middleware
ItemController
ImageController
InvoiceController
POST /api/v1/items
GET /api/v1/items
ItemID Middleware
ItemID Middleware
ItemID Middleware
PUT /api/v1/items/:itemID
GET /api/v1/items/:itemID
DELETE /api/v1/items/:itemID
POST /api/v1/items/:itemID/images
PUT /api/v1/items/:itemID/images /preview
ImageID Middleware
DELETE /api/v1/items/:itemID/images /:imageID
GET /api/v1/items/:itemID/images /:imageID
DELETE /api/v1/items/:itemID/invoice
POST /api/v1/items/:itemID/invoice
GET /api/v1/items/:itemID/invoice

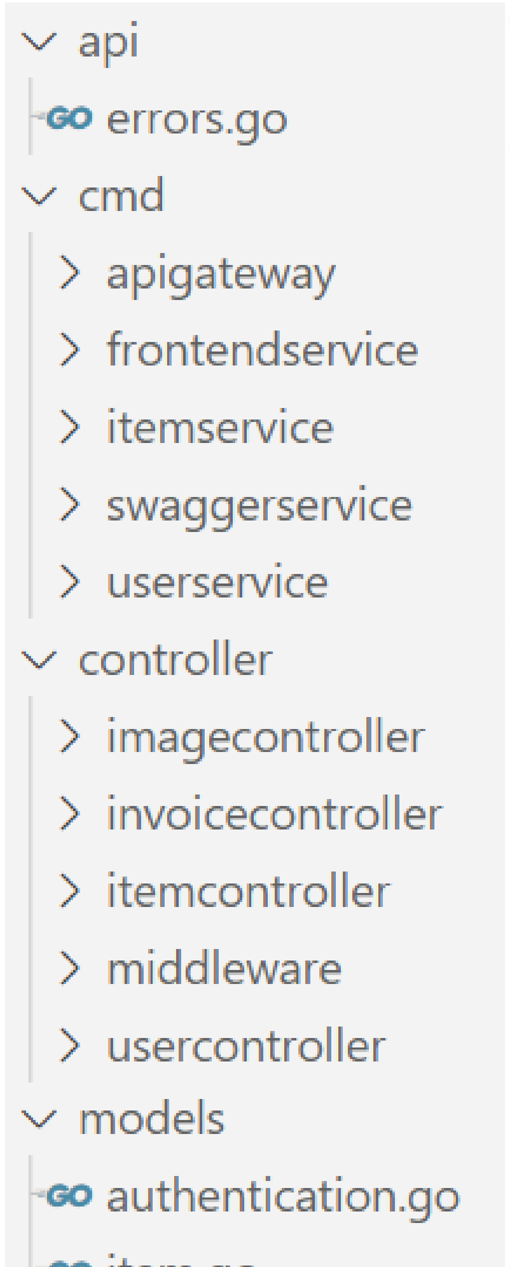
myInventory – Projektstruktur

Knowledge Foundation
@ Reutlingen University



```
api
  errors.go
cmd
  apigateway
  frontendservice
  itemservice
  swaggerservice
  userservice
controller
  imagecontroller
  invoicecontroller
  itemcontroller
  middleware
  usercontroller
models
  authentication.go
  item.go
  user.go
pkg
  authenticator
  database
  logger
  server
  storage
testhelper
  testhelper.go
utils
  envutils
  fileutils
  ginutils
  passwordutils
```

myInventory – Projektstruktur

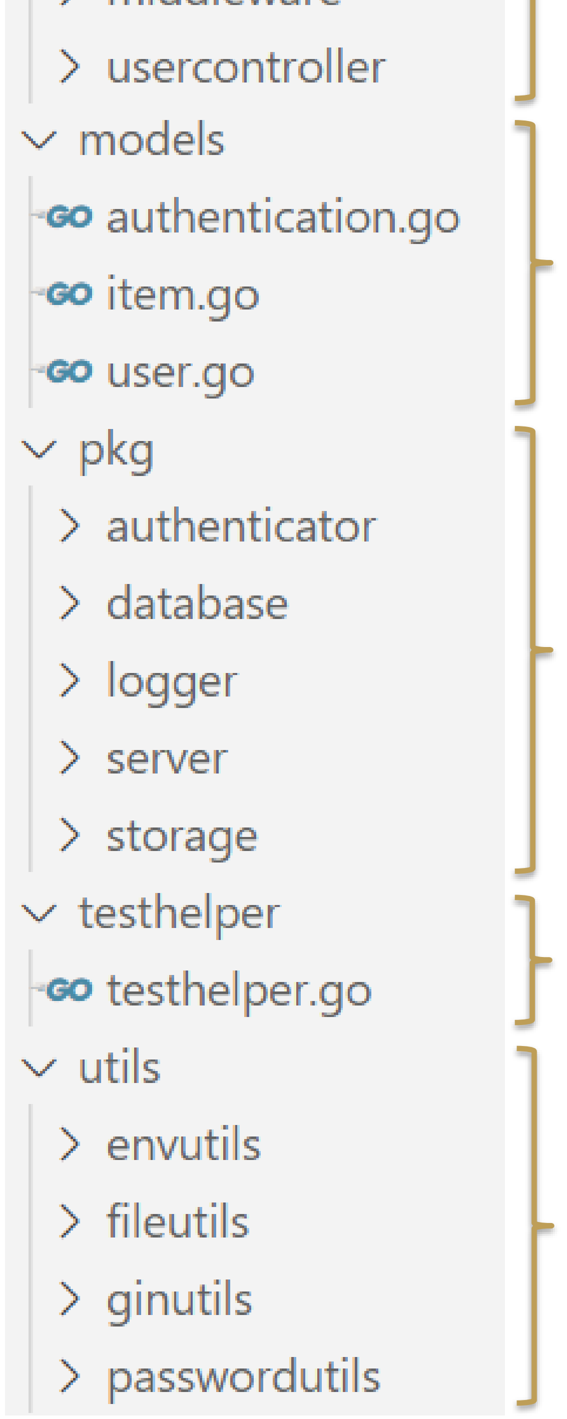


```
v api
  errors.go
v cmd
  > apigateway
  > frontendservice
  > itemservice
  > swaggerservice
  > userservice
v controller
  > imagecontroller
  > invoicecontroller
  > itemcontroller
  > middleware
  > usercontroller
v models
  authentication.go
```

Fehlercodes

Services

Controller (Endpunkte) / Middleware



```
  > middleware
  > usercontroller
v models
  authentication.go
  item.go
  user.go
v pkg
  > authenticator
  > database
  > logger
  > server
  > storage
v testhelper
  testhelper.go
v utils
  > envutils
  > fileutils
  > ginutils
  > passwordutils
```

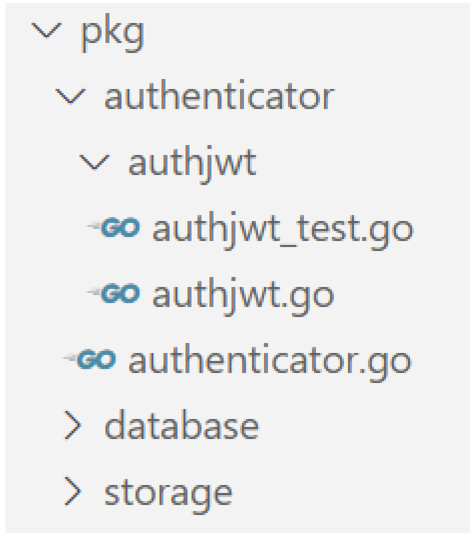
Models

Externe Abhängigkeiten

Hilfsbibliothek für Integrationstests

Hilfsdienste

myInventory – Projektstruktur



Externe Bibliothek

Interface

```go
// The authenticator package is used to include libraries for creating and parsing tokens.
package authenticator

import (
    "errors"

    "github.com/markot99/myinventory-backend/models"
)

var ErrTokenInvalid = errors.New("token invalid")
var ErrGeneratingToken = errors.New("error generating token")

// Authenticator is the interface that wraps the basic methods for creating and parsing tokens
type Authenticator interface {
    // GenerateToken is used to create a token from the given AuthenticationClaims
    GenerateToken(authClaims *models.AuthenticationClaims) (string, error)
    // ValidateToken is used to parse a token and return the AuthenticationClaims
    ValidateToken(token string) (models.AuthenticationClaims, error)
}
```

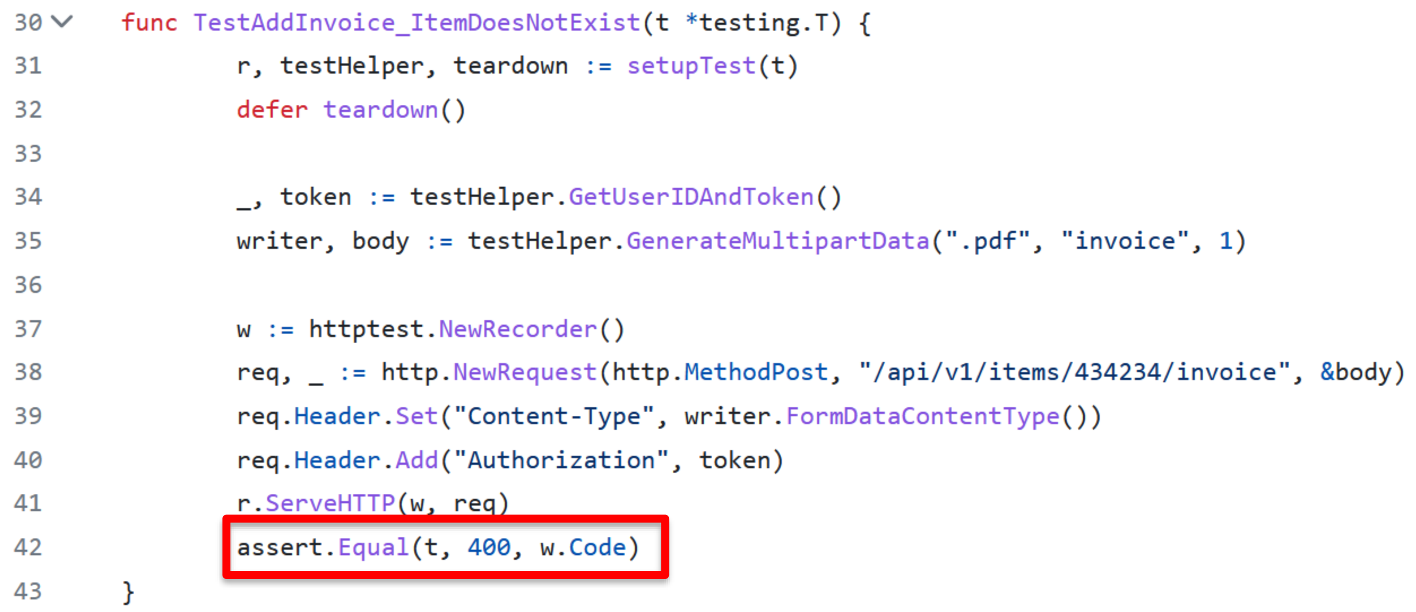
myInventory – Testing

- Kombination aus Unit und Integrationstests
- ~75 % Testabdeckung

Testify

Testing Framework für Go

https://github.com/stretchr/testify



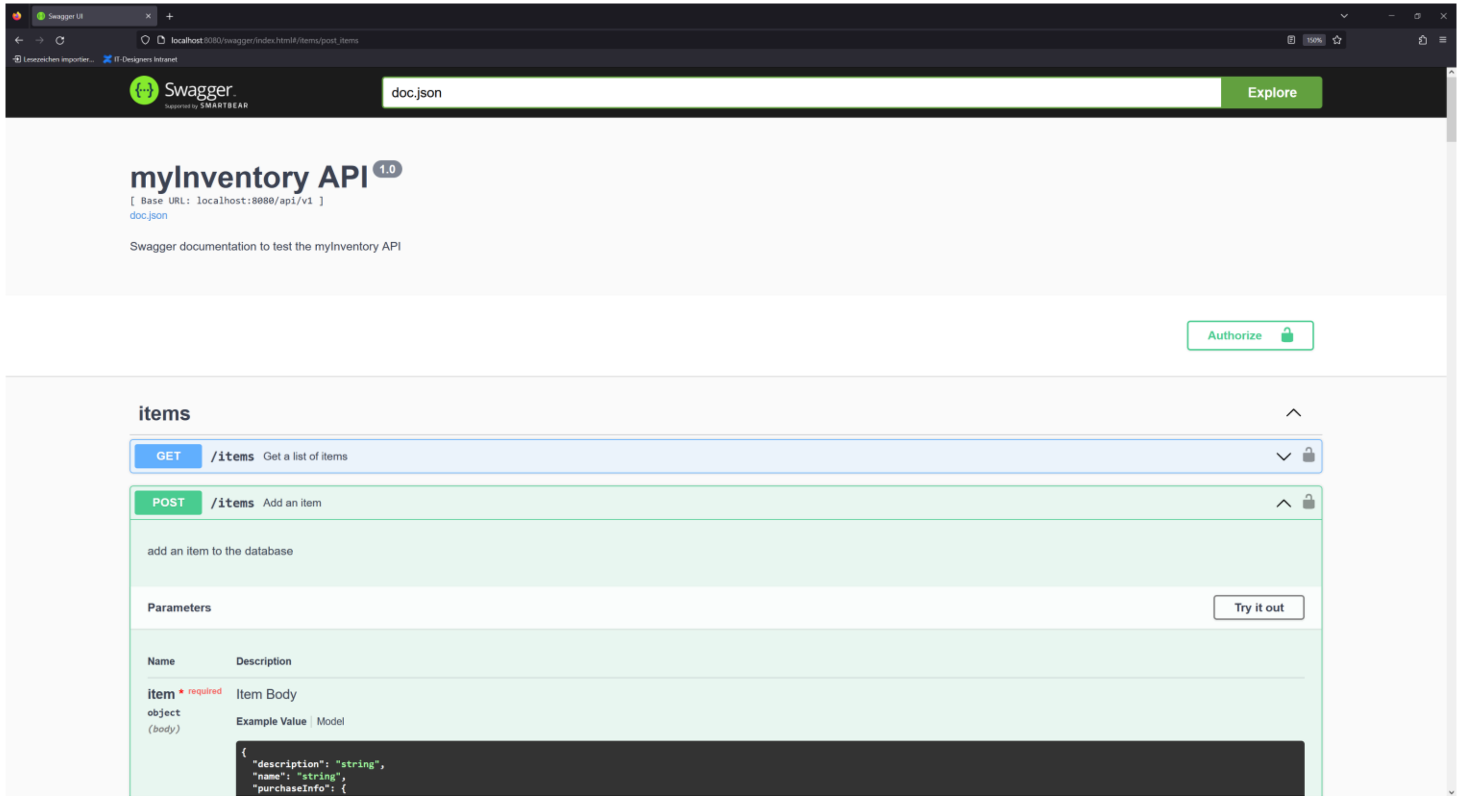
```go
func TestAddInvoice_ItemDoesNotExist(t *testing.T) {
	r, testHelper, teardown := setupTest(t)
	defer teardown()

	_, token := testHelper.GetUserIDAndToken()
	writer, body := testHelper.GenerateMultipartData(".pdf", "invoice", 1)

	w := httptest.NewRecorder()
	req, _ := http.NewRequest(http.MethodPost, "/api/v1/items/434234/invoice", &body)
	req.Header.Set("Content-Type", writer.FormDataContentType())
	req.Header.Add("Authorization", token)
	r.ServeHTTP(w, req)
	assert.Equal(t, 400, w.Code)
}
```

myInventory – Testing

swag

Swagger Dokumentation Generator

https://github.com/swaggo/swag



myInventory API 1.0

`[ Base URL: localhost:8080/api/v1 ]`

doc.json

Swagger documentation to test the myInventory API

items

GET `/items` Get a list of items

POST `/items` Add an item

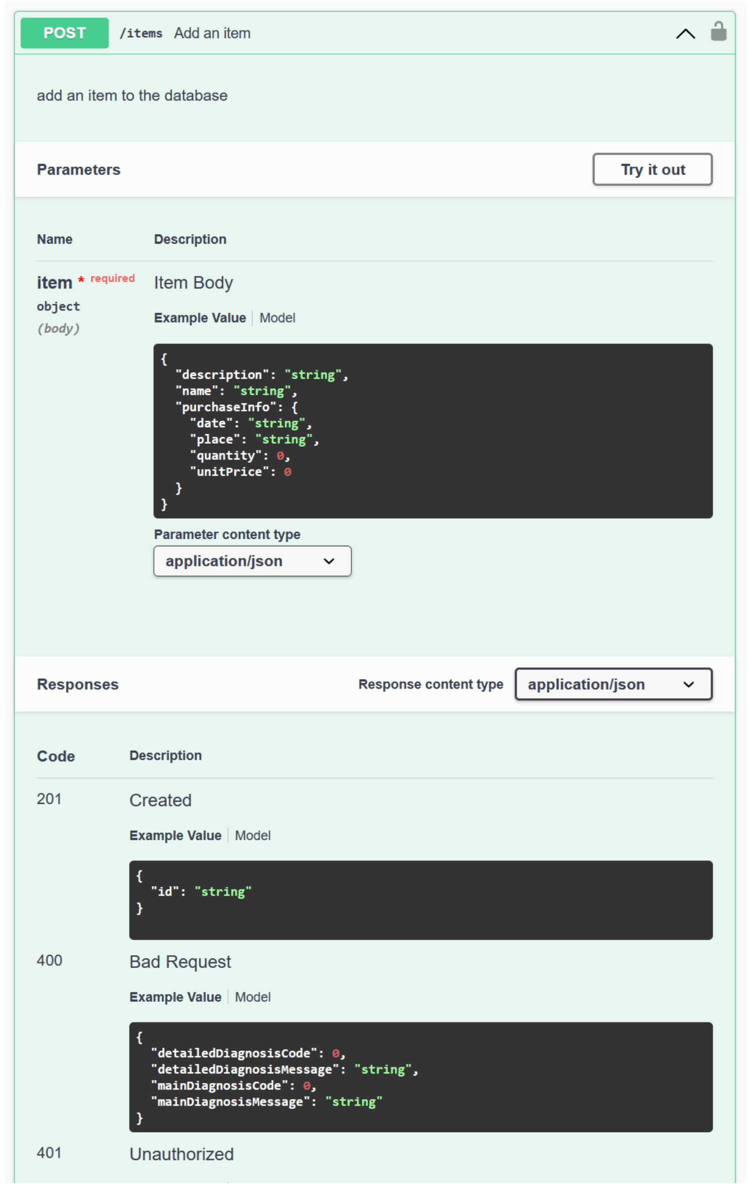
add an item to the database

Parameters

Name	Description
item * required `object` *(body)*	Item Body **Example Value** | Model

```
{
  "description": "string",
  "name": "string",
  "purchaseInfo": {
```

myInventory – Testing



POST /items Add an item

add an item to the database

Parameters — Try it out

Name	Description
item * required object *(body)*	Item Body

Example Value | Model

```
{
  "description": "string",
  "name": "string",
  "purchaseInfo": {
    "date": "string",
    "place": "string",
    "quantity": 0,
    "unitPrice": 0
  }
}
```

Parameter content type: application/json

Responses — Response content type: application/json

Code	Description
201	Created
400	Bad Request
401	Unauthorized

201 Created — Example Value | Model

```
{
  "id": "string"
}
```

400 Bad Request — Example Value | Model

```
{
  "detailedDiagnosisCode": 0,
  "detailedDiagnosisMessage": "string",
  "mainDiagnosisCode": 0,
  "mainDiagnosisMessage": "string"
}
```

```go
type AddItemRequestBody struct {
        Name         string              `json:"name" binding:"required"`
        Description  string              `json:"description"`
        PurchaseInfo AddItemPurchaseInfo `json:"purchaseInfo" binding:"required"`
}

type AddItemResponseBody struct {
        ID string `json:"id"`
}

// AddItem, uses authentication middleware
//
//      @Summary      Add an item
//      @Description  Add an item to the database
//      @Security     JWT
//      @Tags         items
//      @Accept       json
//      @Param        item body AddItemRequestBody true "Item Body"
//      @Success      201 {object} AddItemResponseBody
//      @Failure      400 {object} api.APIErrorResponse
//      @Failure      401 {object} api.APIErrorResponse
//      @Failure      500 {object} api.APIErrorResponse
//      @Router       /items [post]
func (controller ItemController) AddItem(c *gin.Context) {
```

Live Demo

FAZIT UND AUSBLICK