Table of Contents

Basics

This chapter explains the basics of the RESTful API Gateway. What is an*API Token*, how to use this RESTful API Gateway etc.

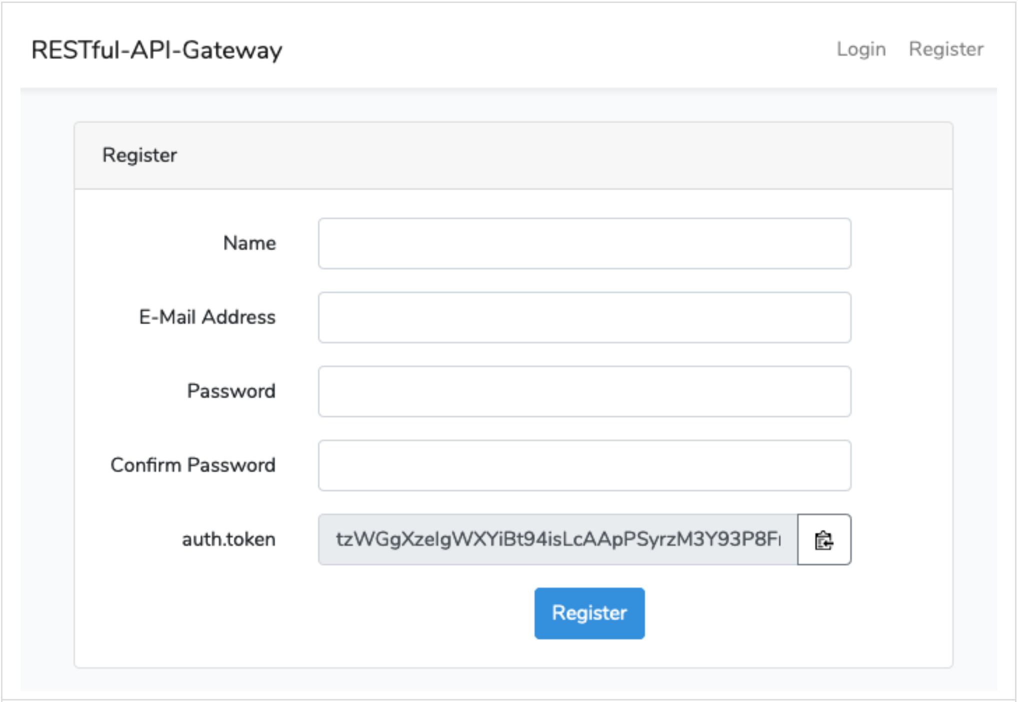
Table of Contents

User / API Token

Almost all activities require an user / API token. In this system user tokens and API tokens will be the same. How to pass the API token see here.

To get an API token you have to register yourself on the registration page: `/gateway/register` .

Here you can enter your data to register yourself.



Name	
E-Mail Address	
Password	
Confirm Password	
auth.token	tzWGgXzelgWXYiBt94isLcAApPSyrzM3Y93P8Fı

Figure 10-1: *Registration*

> **Important**: Copy and save the API token. Only the hash will be saved that means the token can **not** be restored if you lose it.

After you are registered you are also able to log yourself in.

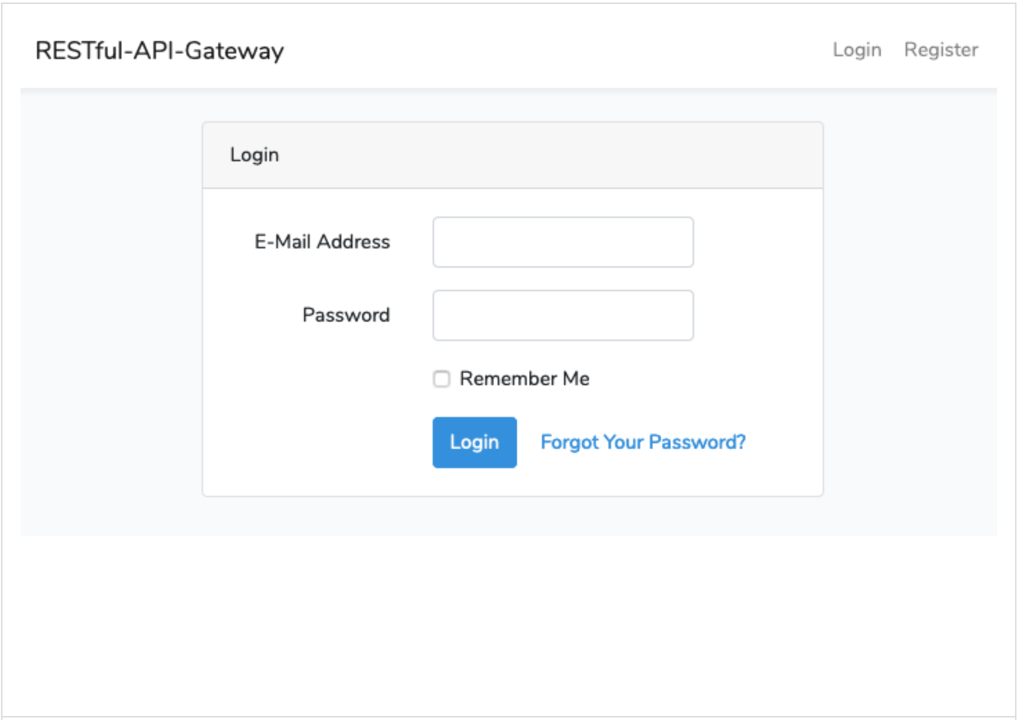


Figure 10-2: *Login*

Authentication

The authentication is provided with Laravels API authentication.

> There are several ways of passing the API token to your application. [...] You may choose any of these approaches based on the needs of your application.

For more information see the Laravel documentation.

One of the easiest ways is to add the query operator api_token:
`<url>` `?api_token=` `<your-api-token>`

Responses

By default, the response data will be JSON. The supported response types are JSON and XML. To set the response type add `Accept: application/json` or `Accept: application/xml` to the headers.

Examples

```
# REST: JSON
curl "<url>"

# REST: XML
curl -H "Accept: application/xml" "<url>"

# GraphQL: JSON
curl -H "Content-Type: application/graphql" -X POST "<url>"

# GraphQL: XML
curl -H "Accept: application/xml" -H "Content-Type: application/graphql" -X POST "<url>"
```

Registration

This chapter deals with the registration process and how to offer your APIs.

Table of Contents

API Definition

To register an API the system requires an OpenAPI specification v3 of your API.

> **OpenAPI Specification** (formerly Swagger Specification) is an API description format for REST APIs. An OpenAPI file allows you to describe your entire API.

For more information about OpenAPI visit the homepage or look into the decisions of the implementation chapter.

Register API

The registration of you API is simple. Just send your OpenAPI definition to the system. You can pass the OpenAPI definition via url, text or file.

```
# system url
url="http://localhost/gateway/api?api_token=<api-token>"

# option a) OpenAPI definition via url
data='{
    "url": "<url-to-openapi-yaml>"
}'
# option b) OpenAPI definition via text
data='{
    "definition": "<openapi-yaml-as-string>"
}'
# option c) OpenAPI definition via file
data='{
    "file": "<file-upload-with-openapi-yaml>"
}'

# send request
curl -d "$data" -H "Content-Type: application/json" -X POST "$url"
```

> **Note:** The response is the saved API with its Id.
>
> **Note:** RESTful resources should always be plural and in english. Due to some reasons some APIs ignore this unwritten rule. To match the resource `user` and `users` together and fix the singular form, each resource name will be tried to pluralize.

Change resource names

If you are using non standard resource names you have to change them when you register the API.

Otherwise a resource matching will not be possible.

```
url="http://localhost/gateway/api?api_token=<api-token>"
data='{
    "url": "<url-to-openapi-yaml>",
    "dictionary": {
        "Benutzer": "users",
        "Kommentare": "comments"
    }
}'
curl -d "$data" -H "Content-Type: application/json" -X POST "$url"
```

> **Note:** If possible, a resource will be automatically pluralized.
> `/user -> /users`

Resource whitelist/blacklist

To restrict your offered resources you can use a whitelist or blacklist. With the keywords `include` and `exclude` you are able to define which resources we want to offer and which not.

```
# whitelist with include
url="http://localhost/gateway/api?api_token=<api-token>"
data='{
    "url": "<url-to-openapi-yaml>",
    "include": [
        "/users",
        "/users/{id}/comments",
        "..."
    ]
}'
curl -d "$data" -H "Content-Type: application/json" -X POST "$url"

# blacklist with exclude
url="http://localhost/gateway/api?api_token=<api-token>"
data='{
    "url": "<url-to-openapi-yaml>",
    "exclude": [
        "/users",
        "/users/{id}/comments",
        "..."
    ]
}'
curl -d "$data" -H "Content-Type: application/json" -X POST "$url"
```

List all your APIs

To get a list of all your offers you can log yourself in on the webpage or request them as a JSON response.

```
curl "http://localhost/gateway/api?api_token=<api-token>"
```

Show API

To get the data of one of your offers you can log yourself in on the webpage or request them as a JSON response.

```
curl "http://localhost/gateway/api/<api-id>?api_token=<api-token>"
```

Update API

An API update is nothing else then a deletion of the existing API and a store of the new one.

Like in the registration it is possible to pass the OpenAPI definition via url, text or file.

```
url="http://localhost/gateway/api/<api-id>?api_token=<api-token>"
data='{
    "url": "<url-to-openapi-yaml>"
}'
curl -d "$data" -H "Content-Type: application/json" -X PUT "$url"
```

Instead of PUT you can also you PATCH.

Delete API

To delete an API offer just send a delete request.

```
curl -X DELETE "http://localhost/gateway/api/<api-id>?api_token=<api-token>"
```

API Evaluation

Each resource is evaluated by its response or transfer time.
You can create a cron job to evaluate all resources periodically or start it manually.
With each resource changes, e.g. new API registration, the system will start an evaluation automatically.

```
# manually start api generation
php artisan api:evaluate
# or
./devops art api:evaluate
```

Gateway Open API Specification

The RESTful API Gateway will always generate a new OpenAPI specification or definition after API changes and api evaluations. This specification is available under /openapi.yaml.

```
# manually start api generation
php artisan api:generate-definition
# or
./devops art api:generate-definition
```

Request Data

This chapter will show you how to request data from the RESTful API Gateway.

Table of Contents

Get data via REST API

By default all avalible data will be responded. If you want to limit the response to just a few fields you could pass the fields via the query operator.

```
# field limitation examples
-d 'fields=id|username|comments.id|comments.content'
-d 'fields=id|username|comments'
-d 'fields=id|username|comments.*'
-d 'fields=*'

# or
"<url>?fields=..."
```

Examples

```
# get all users
curl "http://localhost/users?api_token=<api-token>"

# get all users only the id
curl "http://localhost/users?api_token=<api-token>&fields=id"

# get user with id
curl "http://localhost/users/b6665598-c35c-4b70-bbdb-dc4e14f1a6e9?api_token=<api-token>"

# get user comments
curl "http://localhost/users/b6665598-c35c-4b70-bbdb-dc4e14f1a6e9/comments?api_token=<api-token>"

# get user comment with id
curl "http://localhost/users/b6665598-c35c-4b70-bbdb-dc4e14f1a6e9/comments/0a8b39e3-4bcb-442d-8543-94bd2d46a381?api_token=<api-token>"
```

Get data via GraphQL

For GraphQL support you need to add `Content-Type: application/graphql` to the headers.

To request data via GraphQL there are two posibile ways. Send your GraphQL Query directly or in JSON encapsulated.

```
# Request via GraphQL Query
{
    users {
        id,
        comments {
            id,
            content
        },
        username
    }
}

# Request via JSON
{ "query": "{users{id,comments{id,content},username}}" }
```

Examples
The following examples will be sent directly as GraphQL Queries.

```
# get all users
data='{
    users {
        id,
        comments {
            id,
            content
        },
        username
    }
}'
curl -d "$data" -H "Content-Type: application/graphql" -X POST "http://localhost?api_token=<api-token>"
```

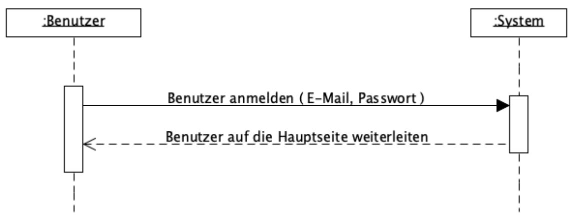
```
# get user with id
data='{
    user (id: "b6665598-c35c-4b70-bbdb-dc4e14f1a6e9") {
        id,
        comments {
            id,
            content
        },
        username
    }
}'
curl -d "$data" -H "Content-Type: application/graphql" -X POST "http://localhost?api_token=<api-token>"

# get user comments
data='{
    comments {
        id,
        content
    }
}'
curl -d "$data" -H "Content-Type: application/graphql" -X POST "http://localhost/users/b6665598-c35c-4b70-bbdb-dc4e14f1a6e9?api_token=<api-token>"

# get user comment with id
data='{
    comment (id: "0a8b39e3-4bcb-442d-8543-94bd2d46a381") {
        id,
        content
    }
}'
curl -d "$data" -H "Content-Type: application/graphql" -X POST "http://localhost/users/b6665598-c35c-4b70-bbdb-dc4e14f1a6e9?api_token=<api-token>"
```

Communication

Multiple sequence and activity diagrams should visualize the communication flow of the system.

Table of Contents

User Login / Registration



:Benutzer
:System
Benutzer anmelden (E-Mail, Passwort)
Benutzer auf die Hauptseite weiterleiten

Figure 40-1: *Sequence Diagram: User login (german)*

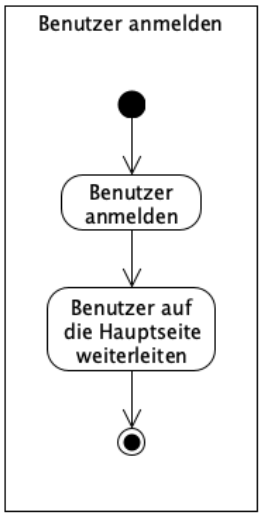


Figure 40-2: *Activity Diagram: User login (german)*

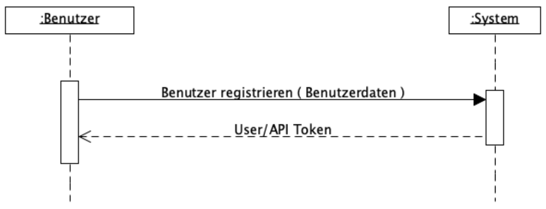


Figure 40-3: *Sequence Diagram: User registration (german)*

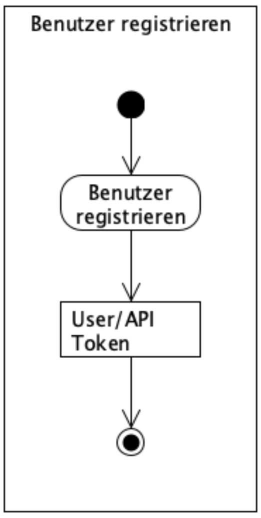


Figure 40-4: *Activity Diagram: User registration (german)*

Get API Definition

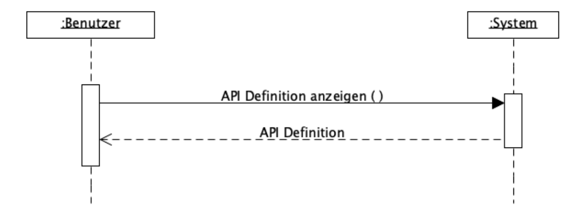


Figure 40-5: *Sequence Diagram: Get API definition (german)*

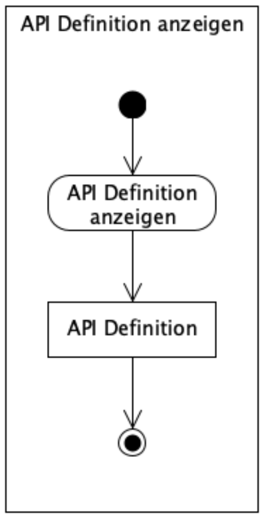


Figure 40-6: *Activity Diagram: Get API definition (german)*

API Registration

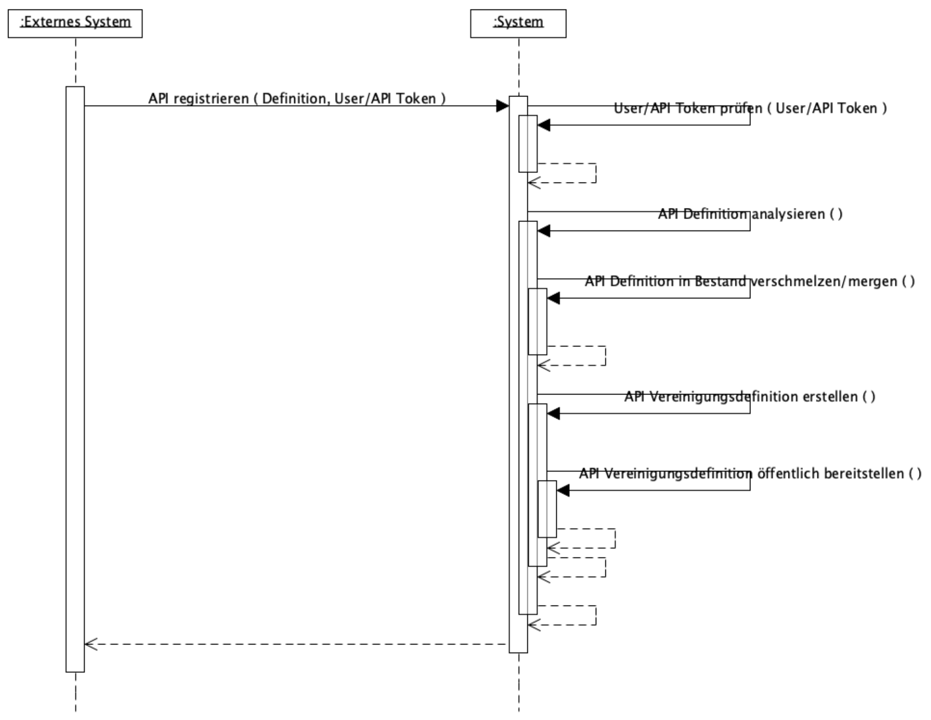

:Externes System
:System
API registrieren (Definition, User/API Token)
User/API Token prüfen (User/API Token)
API Definition analysieren ( )
API Definition in Bestand verschmelzen/mergen ( )
API Vereinigungsdefinition erstellen ( )
API Vereinigungsdefinition öffentlich bereitstellen ( )

Figure 40-7: *Sequence Diagram: API registration (german)*

Delete API

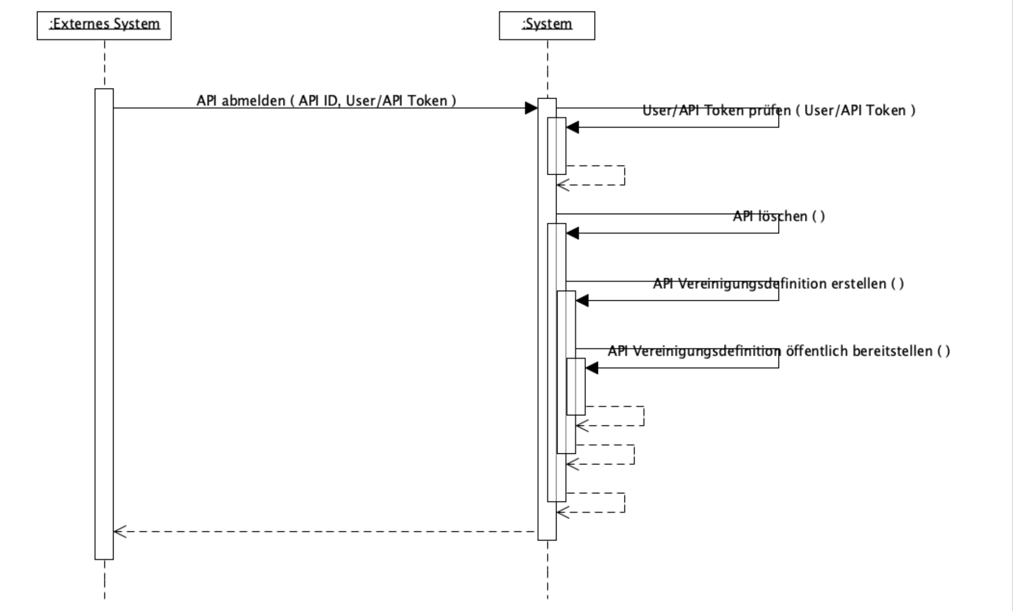

:Externes System
:System
API abmelden (API ID, User/API Token)
User/API Token prüfen (User/API Token)
API löschen ( )
API Vereinigungsdefinition erstellen ( )
API Vereinigungsdefinition öffentlich bereitstellen ( )

Figure 40-8: *Sequence Diagram: Delete API (german)*

API Request

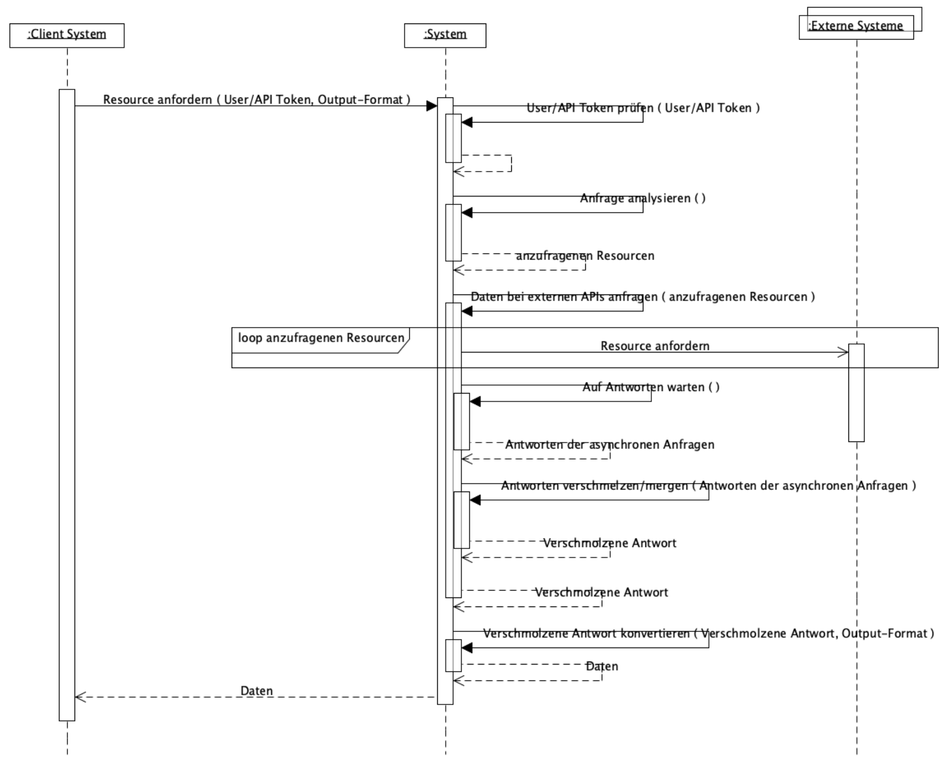

:Client System
:System
:Externe Systeme
Resource anfordern (User/API Token, Output-Format)
User/API Token prüfen (User/API Token)
Anfrage analysieren ( )
anzufragenen Resourcen
Daten bei externen APIs anfragen (anzufragenen Resourcen)
loop anzufragenen Resourcen
Resource anfordern
Auf Antworten warten ( )
Antworten der asynchronen Anfragen
Antworten verschmelzen/mergen (Antworten der asynchronen Anfragen)
Verschmolzene Antwort
Verschmolzene Antwort
Verschmolzene Antwort konvertieren (Verschmolzene Antwort, Output-Format)
Daten
Daten

Figure 40-9: *Sequence Diagram: API request (german)*

Application Structure

In the following you can see the app structure visulized with class diagramms.

The system is realized with PHP and the framework Laravel.
PHP is a language that optionally looks at typitization. For this reason the data type Mixed is used in this diagram. Behind this data type can hide one or more arbitrary data types.

"Packages" which only occupy a dashed frame are not present in the source code due to given structures, e.g. by a framework, and should only represent the affiliation of their contents.

Classes and packages that are gray are external modules that are only available for better understanding.

Table of Contents

Overview

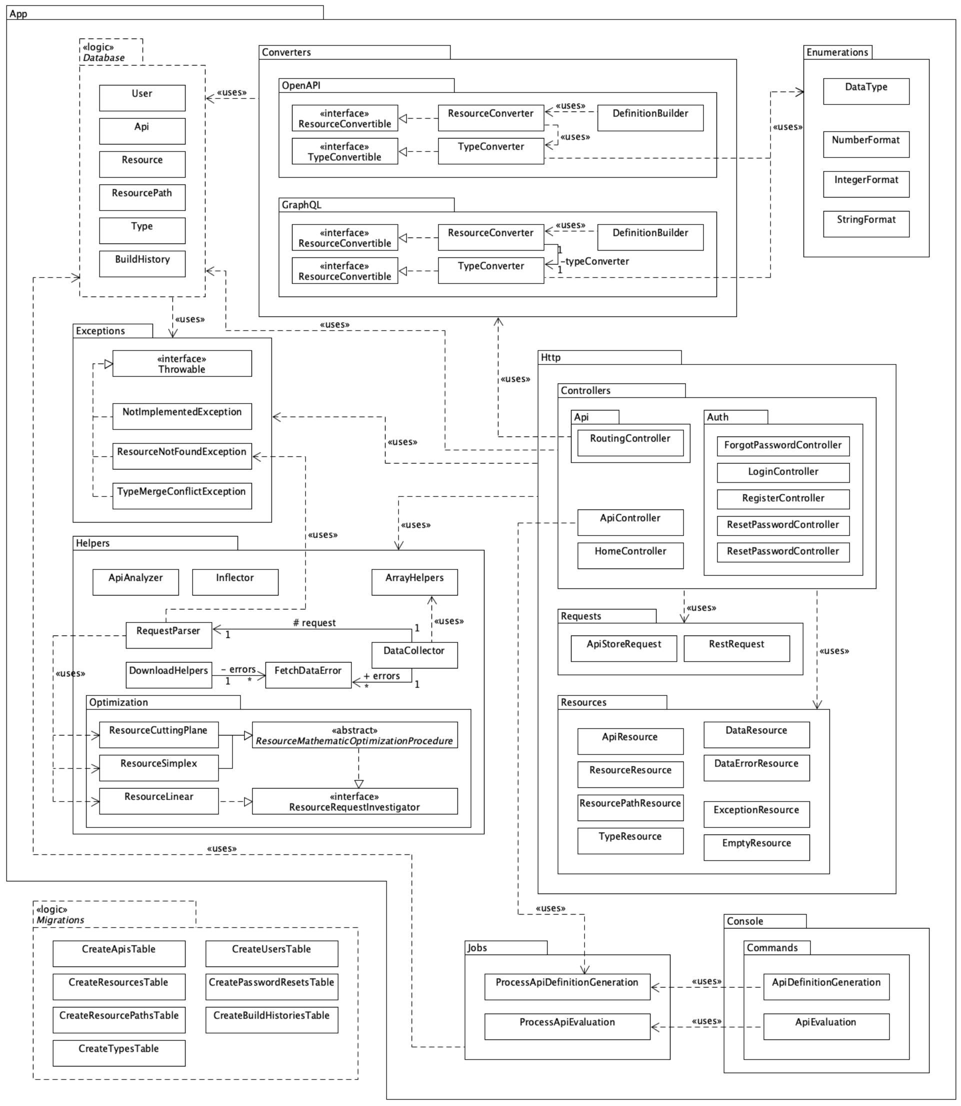


Figure 50-01: *Class Diagram: Overview*

Database

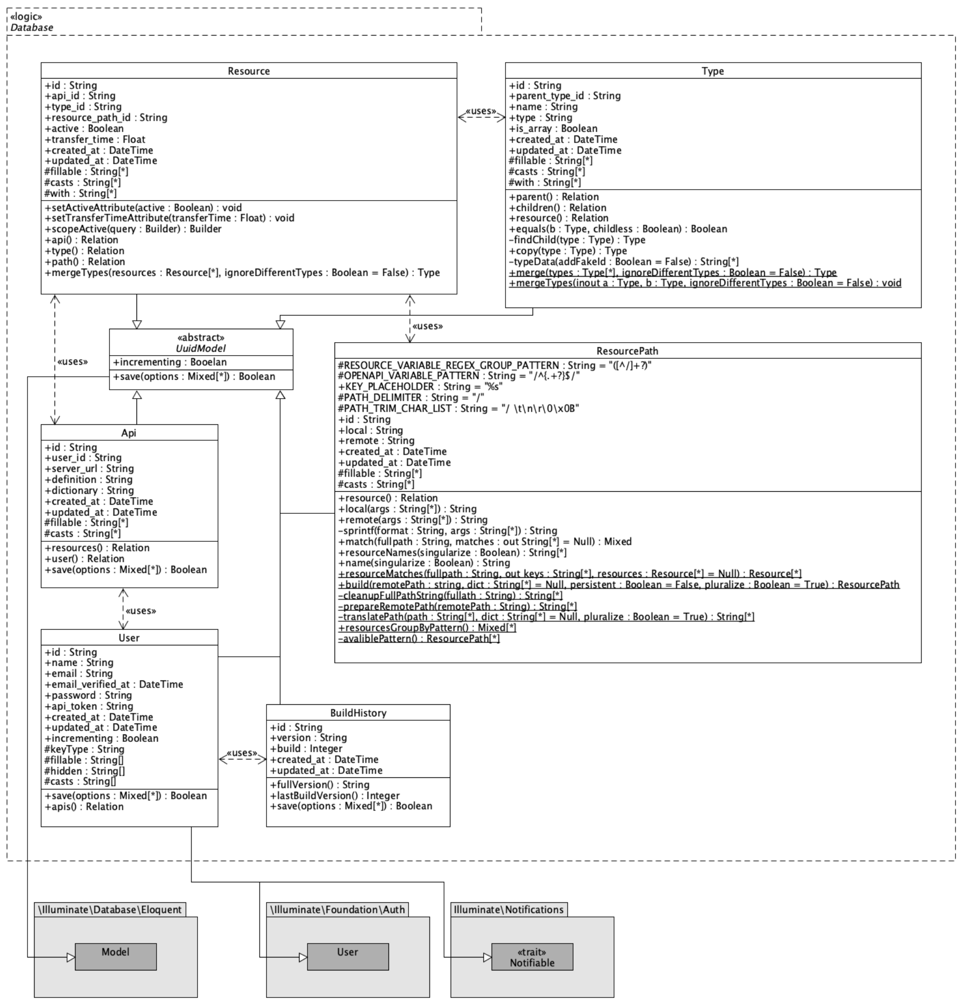

«logic»
Database
Resource
+id : String
+api_id : String
+type_id : String
+resource_path_id : String
+active : Boolean
+transfer_time : Float
+created_at : DateTime
+updated_at : DateTime
#fillable : String[*]
#casts : String[*]
#with : String[*]
+setActiveAttribute(active : Boolean) : void
+setTransferTimeAttribute(transferTime : Float) : void
+scopeActive(query : Builder) : Builder
+api() : Relation
+type() : Relation
+path() : Relation
+mergeTypes(resources : Resource[*], ignoreDifferentTypes : Boolean = False) : Type
«uses»
Type
+id : String
+parent_type_id : String
+name : String
+type : String
+is_array : Boolean
+created_at : DateTime
+updated_at : DateTime
#fillable : String[*]
#casts : String[*]
#with : String[*]
+parent() : Relation
+children() : Relation
+resource() : Relation
+equals(b : Type, childless : Boolean) : Boolean
-findChild(type : Type) : Type
+copy(type : Type) : Type
-typeData(addFakeId : Boolean = False) : String[*]
+merge(types : Type[*], ignoreDifferentTypes : Boolean = False) : Type
+mergeTypes(inout a : Type, b : Type, ignoreDifferentTypes : Boolean = False) : void
«uses»
«abstract»
UuidModel
+incrementing : Booelan
+save(options : Mixed[*]) : Boolean
«uses»
ResourcePath
#RESOURCE_VARIABLE_REGEX_GROUP_PATTERN : String = "([^/]+?)"
#OPENAPI_VARIABLE_PATTERN : String = "/^{.+?}$/"
+KEY_PLACEHOLDER : String = "%s"
#PATH_DELIMITER : String = "/"
#PATH_TRIM_CHAR_LIST : String = "/ \t\n\r\0\x0B"
+id : String
+local : String
+remote : String
+created_at : DateTime
+updated_at : DateTime
#fillable : String[*]
#casts : String[*]
+resource() : Relation
+local(args : String[*]) : String
+remote(args : String[*]) : String
-sprintf(format : String, args : String[*]) : String
+match(fullpath : String, matches : out String[*] = Null) : Mixed
+resourceNames(singularize : Boolean) : String[*]
+name(singularize : Boolean) : String
+resourceMatches(fullpath : String, out keys : String[*], resources : Resource[*] = Null) : Resource[*]
+build(remotePath : string, dict : String[*] = Null, persistent : Boolean = False, pluralize : Boolean = True) : ResourcePath
-cleanupFullPathString(fullath : String) : String[*]
-prepareRemotePath(remotePath : String) : String[*]
-translatePath(path : String[*], dict : String[*] = Null, pluralize : Boolean = True) : String[*]
+resourcesGroupByPattern() : Mixed[*]
-avaliblePattern() : ResourcePath[*]
Api
+id : String
+user_id : String
+server_url : String
+definition : String
+dictionary : String
+created_at : DateTime
+updated_at : DateTime
#fillable : String[*]
#casts : String[*]
+resources() : Relation
+user() : Relation
+save(options : Mixed[*]) : Boolean
«uses»
User
+id : String
+name : String
+email : String
+email_verified_at : DateTime
+password : String
+api_token : String
+created_at : DateTime
+updated_at : DateTime
+incrementing : Boolean
#keyType : String
#fillable : String[]
#hidden : String[]
#casts : String[]
+save(options : Mixed[*]) : Boolean
+apis() : Relation
«uses»
BuildHistory
+id : String
+version : String
+build : Integer
+created_at : DateTime
+updated_at : DateTime
+fullVersion() : String
+lastBuildVersion() : Integer
+save(options : Mixed[*]) : Boolean
\Illuminate\Database\Eloquent
Model
\Illuminate\Foundation\Auth
User
Illuminate\Notifications
«trait»
Notifiable

Figure 50-02: *Class Diagram: Database*

Http

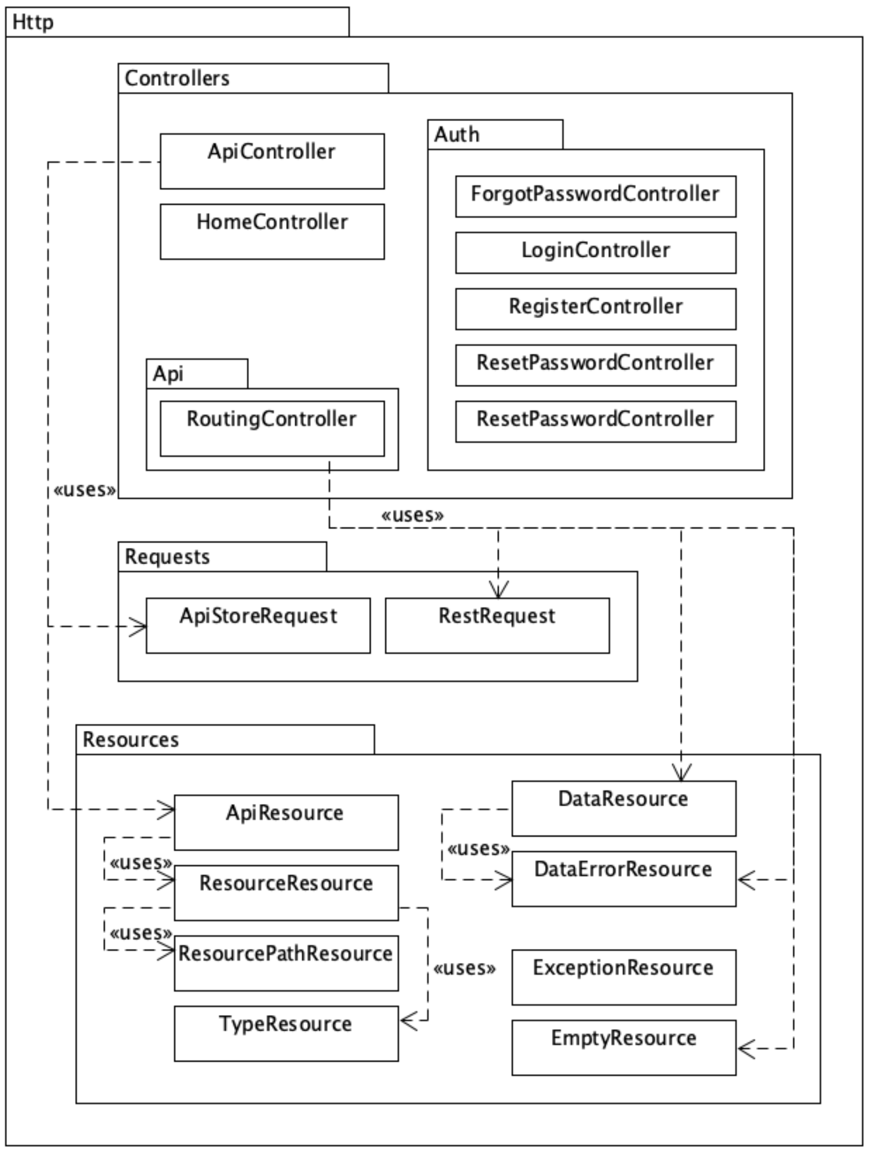


Figure 50-03: *Class Diagram: Http*

Controller

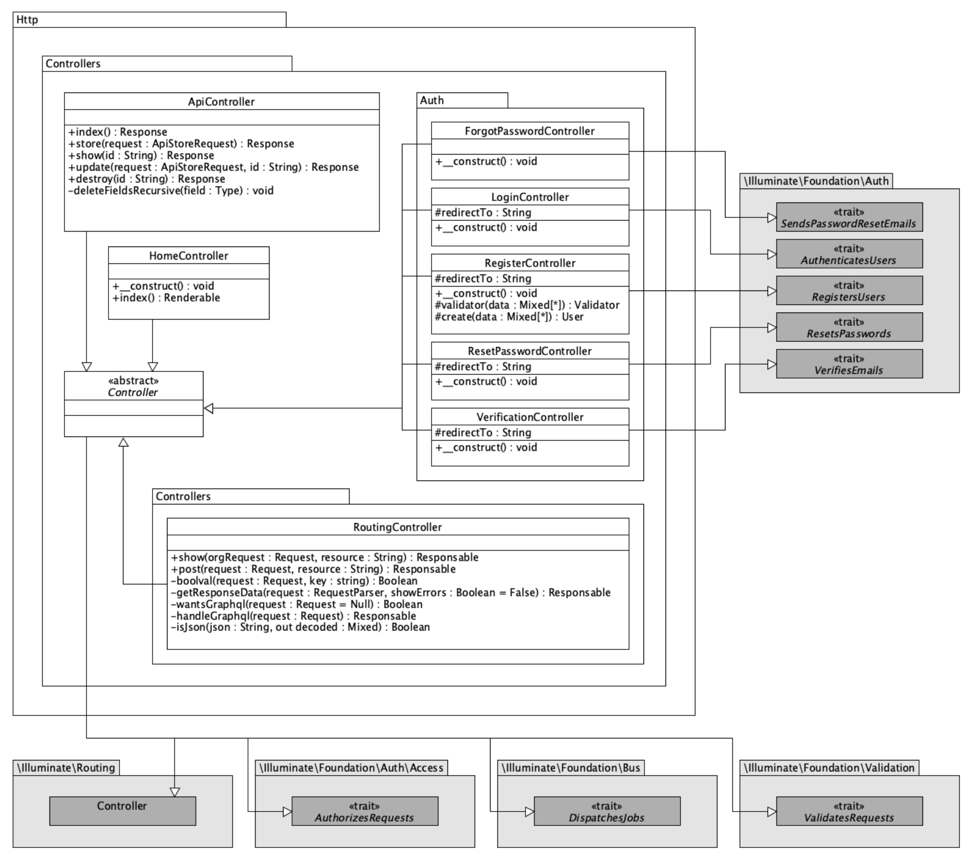

Http
Controllers
ApiController
+index() : Response
+store(request : ApiStoreRequest) : Response
+show(id : String) : Response
+update(request : ApiStoreRequest, id : String) : Response
+destroy(id : String) : Response
−deleteFieldsRecursive(field : Type) : void
HomeController
+__construct() : void
+index() : Renderable
«abstract»
Controller
Auth
ForgotPasswordController
+__construct() : void
LoginController
#redirectTo : String
+__construct() : void
RegisterController
#redirectTo : String
+__construct() : void
#validator(data : Mixed[*]) : Validator
#create(data : Mixed[*]) : User
ResetPasswordController
#redirectTo : String
+__construct() : void
VerificationController
#redirectTo : String
+__construct() : void
\Illuminate\Foundation\Auth
«trait»
SendsPasswordResetEmails
«trait»
AuthenticatesUsers
«trait»
RegistersUsers
«trait»
ResetsPasswords
«trait»
VerifiesEmails
Controllers
RoutingController
+show(orgRequest : Request, resource : String) : Responsable
+post(request : Request, resource : String) : Responsable
−boolval(request : Request, key : string) : Boolean
−getResponseData(request : RequestParser, showErrors : Boolean = False) : Responsable
−wantsGraphql(request : Request = Null) : Boolean
−handleGraphql(request : Request) : Responsable
−isJson(json : String, out decoded : Mixed) : Boolean
\Illuminate\Routing
Controller
\Illuminate\Foundation\Auth\Access
«trait»
AuthorizesRequests
\Illuminate\Foundation\Bus
«trait»
DispatchesJobs
\Illuminate\Foundation\Validation
«trait»
ValidatesRequests

Figure 50-04: *Class Diagram: Controller*

Requests

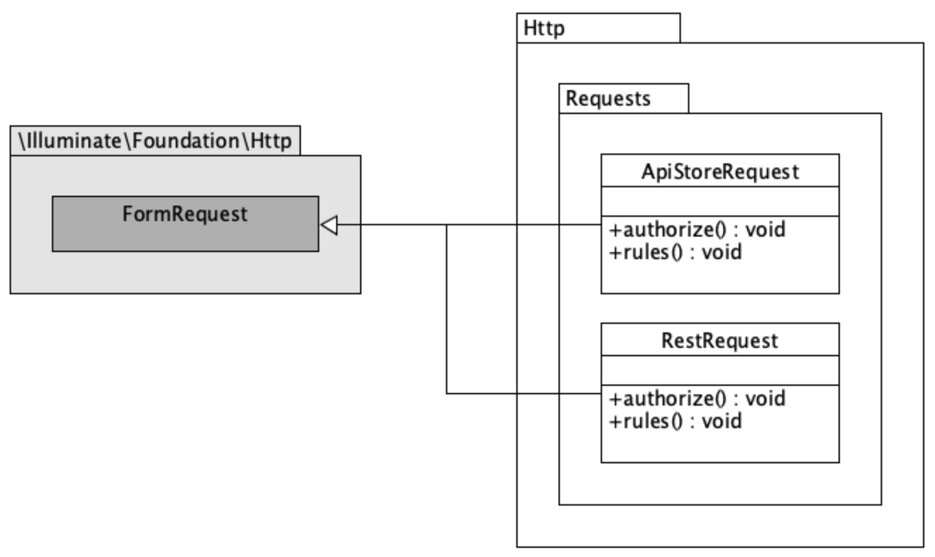


Figure 50-05: *Class Diagram: Requests*

Resources

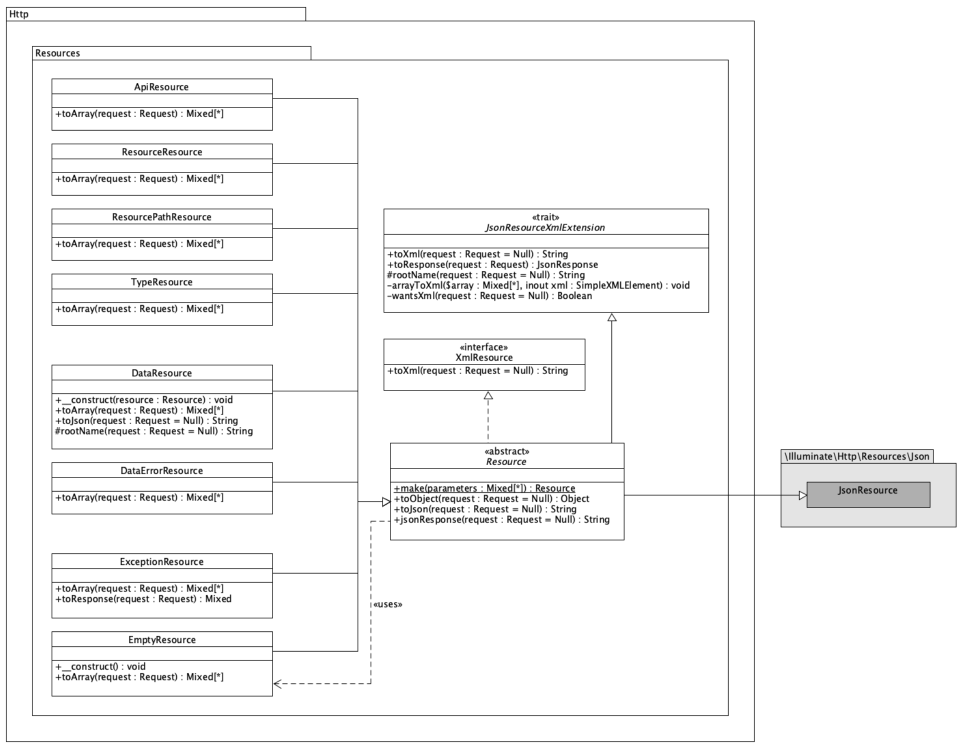

Http
Resources
ApiResource
+toArray(request : Request) : Mixed[*]
ResourceResource
+toArray(request : Request) : Mixed[*]
ResourcePathResource
+toArray(request : Request) : Mixed[*]
TypeResource
+toArray(request : Request) : Mixed[*]
DataResource
+__construct(resource : Resource) : void
+toArray(request : Request) : Mixed[*]
+toJson(request : Request = Null) : String
#rootName(request : Request = Null) : String
DataErrorResource
+toArray(request : Request) : Mixed[*]
ExceptionResource
+toArray(request : Request) : Mixed[*]
+toResponse(request : Request) : Mixed
EmptyResource
+__construct() : void
+toArray(request : Request) : Mixed[*]
«trait»
JsonResourceXmlExtension
+toXml(request : Request = Null) : String
+toResponse(request : Request) : JsonResponse
#rootName(request : Request = Null) : String
-arrayToXml($array : Mixed[*], inout xml : SimpleXMLElement) : void
-wantsXml(request : Request = Null) : Boolean
«interface»
XmlResource
+toXml(request : Request = Null) : String
«abstract»
Resource
+make(parameters : Mixed[*]) : Resource
+toObject(request : Request = Null) : Object
+toJson(request : Request = Null) : String
+jsonResponse(request : Request = Null) : String
«uses»
\Illuminate\Http\Resources\Json
JsonResource

Figure 50-06: *Class Diagram: Resources*

Converters

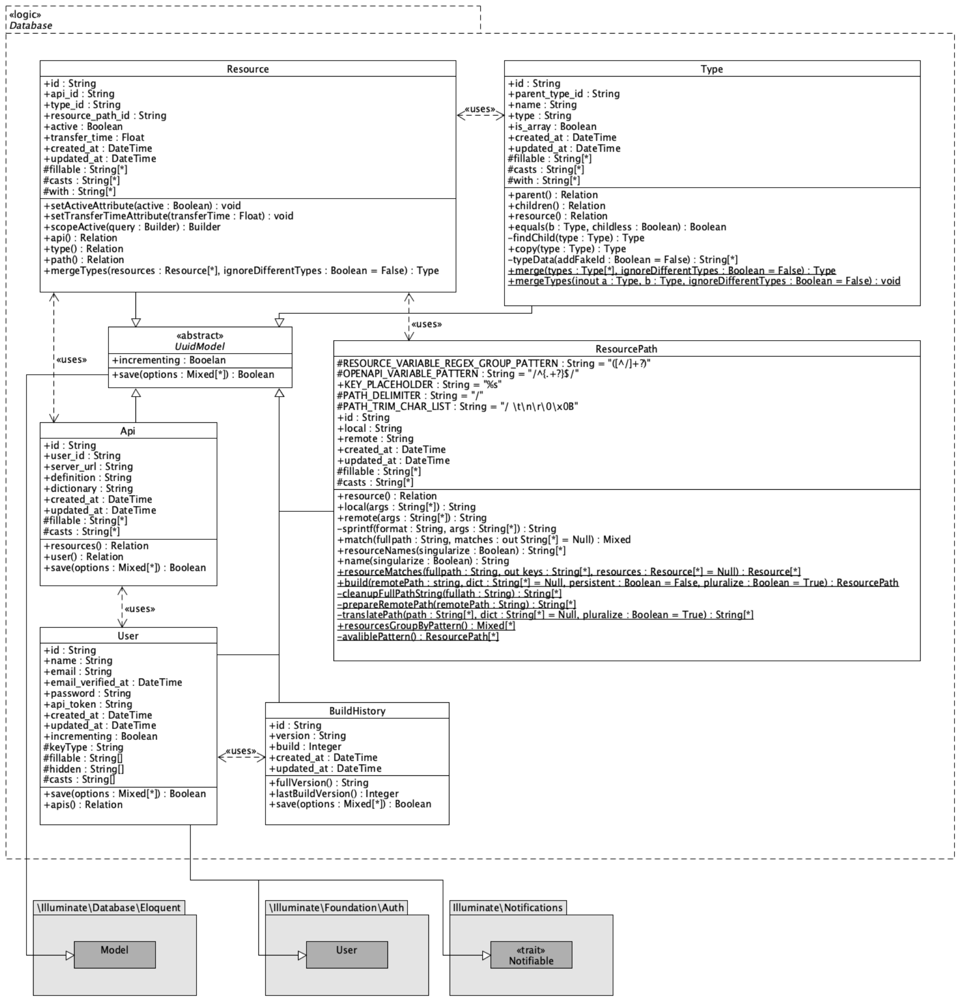

«logic»
Database
Resource
+id : String
+api_id : String
+type_id : String
+resource_path_id : String
+active : Boolean
+transfer_time : Float
+created_at : DateTime
+updated_at : DateTime
#fillable : String[*]
#casts : String[*]
#with : String[*]
+setActiveAttribute(active : Boolean) : void
+setTransferTimeAttribute(transferTime : Float) : void
+scopeActive(query : Builder) : Builder
+api() : Relation
+type() : Relation
+path() : Relation
+mergeTypes(resources : Resource[*], ignoreDifferentTypes : Boolean = False) : Type
«uses»
Type
+id : String
+parent_type_id : String
+name : String
+type : String
+is_array : Boolean
+created_at : DateTime
+updated_at : DateTime
#fillable : String[*]
#casts : String[*]
#with : String[*]
+parent() : Relation
+children() : Relation
+resource() : Relation
+equals(b : Type, childless : Boolean) : Boolean
-findChild(type : Type) : Type
+copy(type : Type) : Type
-typeData(addFakeId : Boolean = False) : String[*]
+merge(types : Type[*], ignoreDifferentTypes : Boolean = False) : Type
+mergeTypes(inout a : Type, b : Type, ignoreDifferentTypes : Boolean = False) : void
«uses»
«abstract»
UuidModel
+incrementing : Booelan
+save(options : Mixed[*]) : Boolean
«uses»
ResourcePath
#RESOURCE_VARIABLE_REGEX_GROUP_PATTERN : String = "([^/]+?)"
#OPENAPI_VARIABLE_PATTERN : String = "/^{.+?}$/"
+KEY_PLACEHOLDER : String = "%s"
#PATH_DELIMITER : String = "/"
#PATH_TRIM_CHAR_LIST : String = "/ \t\n\r\0\x0B"
+id : String
+local : String
+remote : String
+created_at : DateTime
+updated_at : DateTime
#fillable : String[*]
#casts : String[*]
+resource() : Relation
+local(args : String[*]) : String
+remote(args : String[*]) : String
-sprintf(format : String, args : String[*]) : String
+match(fullpath : String, matches : out String[*] = Null) : Mixed
+resourceNames(singularize : Boolean) : String[*]
+name(singularize : Boolean) : String
+resourceMatches(fullpath : String, out keys : String[*], resources : Resource[*] = Null) : Resource[*]
+build(remotePath : string, dict : String[*] = Null, persistent : Boolean = False, pluralize : Boolean = True) : ResourcePath
-cleanupFullPathString(fullath : String) : String[*]
-prepareRemotePath(remotePath : String) : String[*]
-translatePath(path : String[*], dict : String[*] = Null, pluralize : Boolean = True) : String[*]
+resourcesGroupByPattern() : Mixed[*]
-avaliblePattern() : ResourcePath[*]
Api
+id : String
+user_id : String
+server_url : String
+definition : String
+dictionary : String
+created_at : DateTime
+updated_at : DateTime
#fillable : String[*]
#casts : String[*]
+resources() : Relation
+user() : Relation
+save(options : Mixed[*]) : Boolean
«uses»
User
+id : String
+name : String
+email : String
+email_verified_at : DateTime
+password : String
+api_token : String
+created_at : DateTime
+updated_at : DateTime
+incrementing : Boolean
#keyType : String
#fillable : String[]
#hidden : String[]
#casts : String[]
+save(options : Mixed[*]) : Boolean
+apis() : Relation
«uses»
BuildHistory
+id : String
+version : String
+build : Integer
+created_at : DateTime
+updated_at : DateTime
+fullVersion() : String
+lastBuildVersion() : Integer
+save(options : Mixed[*]) : Boolean
\Illuminate\Database\Eloquent
Model
\Illuminate\Foundation\Auth
User
Illuminate\Notifications
«trait»
Notifiable

Figure 50-07: *Class Diagram: Converters*

OpenAPI

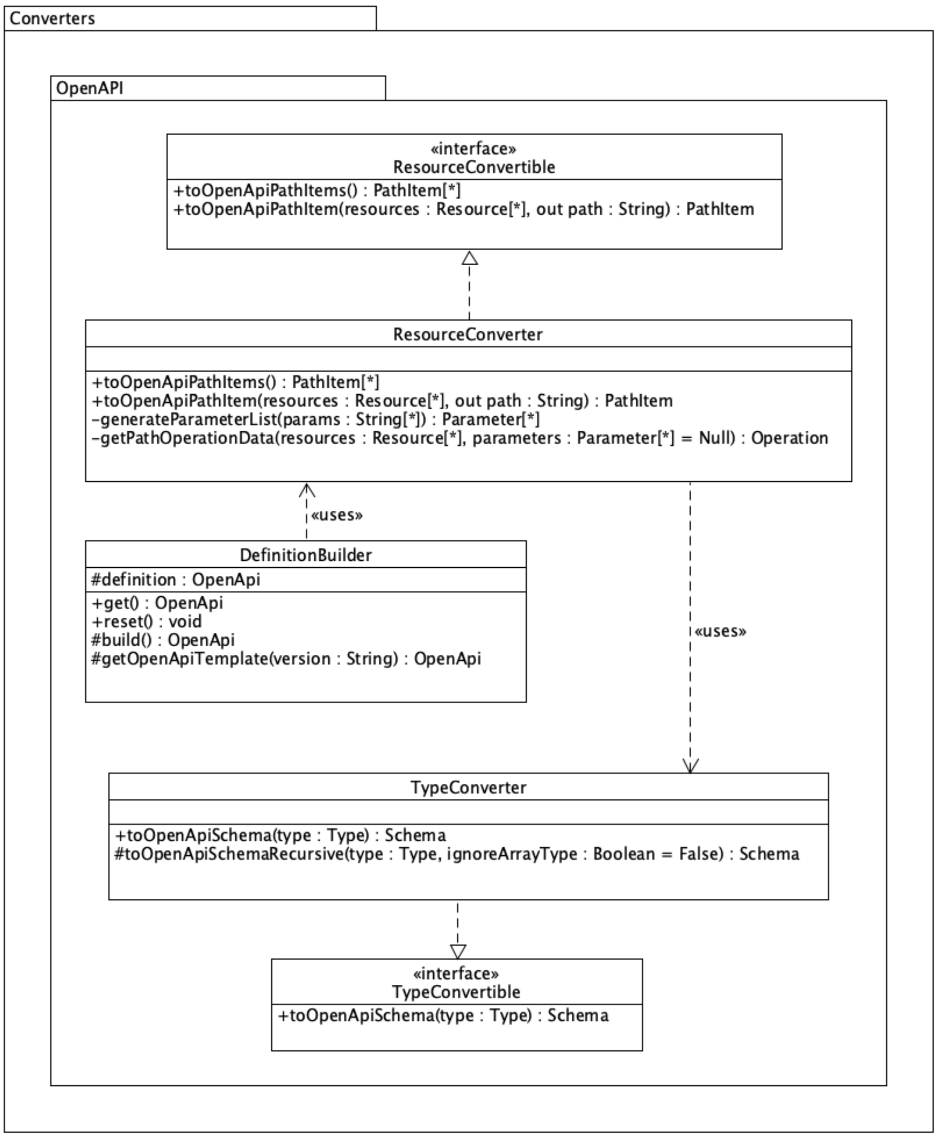

Converters
OpenAPI
«interface»
ResourceConvertible
+toOpenApiPathItems() : PathItem[*]
+toOpenApiPathItem(resources : Resource[*], out path : String) : PathItem
ResourceConverter
+toOpenApiPathItems() : PathItem[*]
+toOpenApiPathItem(resources : Resource[*], out path : String) : PathItem
-generateParameterList(params : String[*]) : Parameter[*]
-getPathOperationData(resources : Resource[*], parameters : Parameter[*] = Null) : Operation
«uses»
DefinitionBuilder
#definition : OpenApi
+get() : OpenApi
+reset() : void
#build() : OpenApi
#getOpenApiTemplate(version : String) : OpenApi
«uses»
TypeConverter
+toOpenApiSchema(type : Type) : Schema
#toOpenApiSchemaRecursive(type : Type, ignoreArrayType : Boolean = False) : Schema
«interface»
TypeConvertible
+toOpenApiSchema(type : Type) : Schema

Figure 50-08: *Class Diagram: OpenAPI*

GraphQL

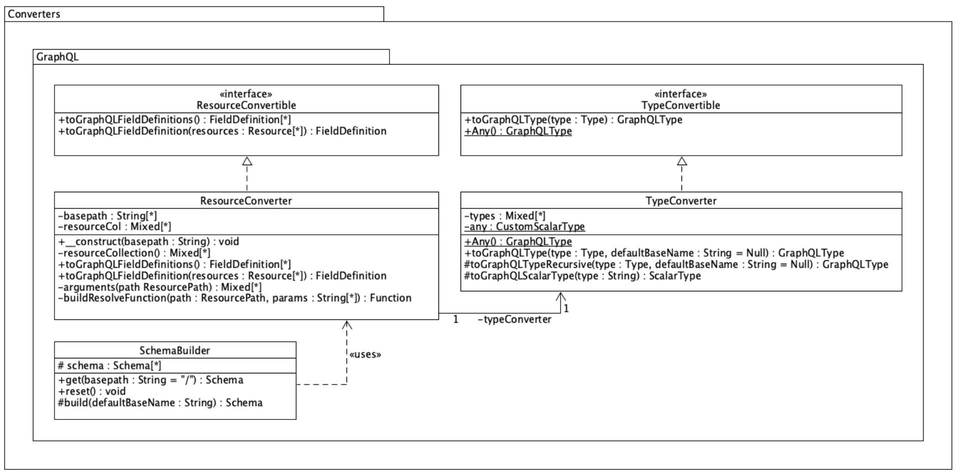

Converters
GraphQL
«interface»
ResourceConvertible
+toGraphQLFieldDefinitions() : FieldDefinition[*]
+toGraphQLFieldDefinition(resources : Resource[*]) : FieldDefinition
«interface»
TypeConvertible
+toGraphQLType(type : Type) : GraphQLType
+Any() : GraphQLType
ResourceConverter
-basepath : String[*]
-resourceCol : Mixed[*]
+__construct(basepath : String) : void
-resourceCollection() : Mixed[*]
+toGraphQLFieldDefinitions() : FieldDefinition[*]
+toGraphQLFieldDefinition(resources : Resource[*]) : FieldDefinition
-arguments(path ResourcePath) : Mixed[*]
-buildResolveFunction(path : ResourcePath, params : String[*]) : Function
TypeConverter
-types : Mixed[*]
-any : CustomScalarType
+Any() : GraphQLType
+toGraphQLType(type : Type, defaultBaseName : String = Null) : GraphQLType
#toGraphQLTypeRecursive(type : Type, defaultBaseName : String = Null) : GraphQLType
#toGraphQLScalarType(type : String) : ScalarType
1
1
-typeConverter
«uses»
SchemaBuilder
# schema : Schema[*]
+get(basepath : String = "/") : Schema
+reset() : void
#build(defaultBaseName : String) : Schema

Figure 50-09: *Class Diagram: GraphQL*

Enumerations

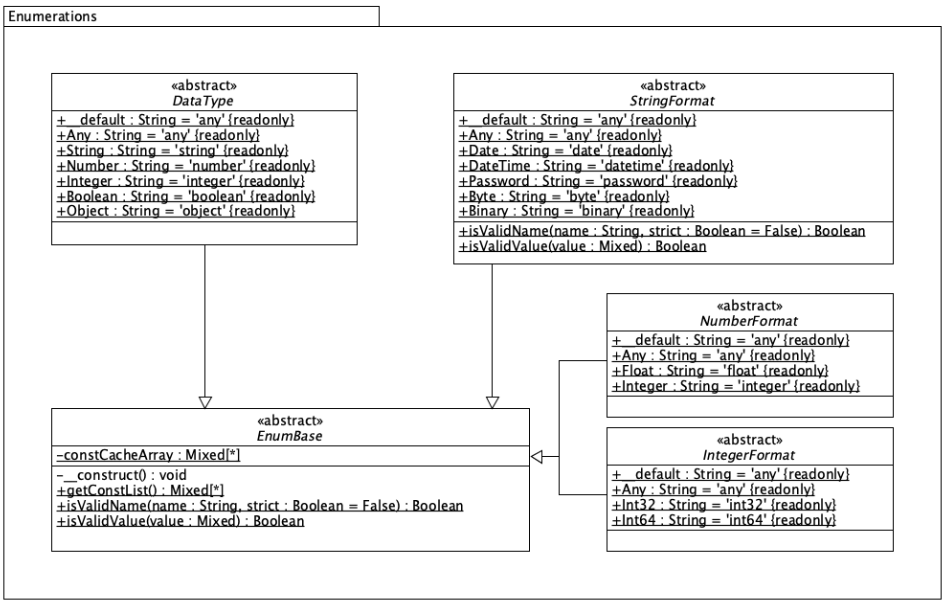

Enumerations
«abstract»
DataType
+__default : String = 'any' {readonly}
+Any : String = 'any' {readonly}
+String : String = 'string' {readonly}
+Number : String = 'number' {readonly}
+Integer : String = 'integer' {readonly}
+Boolean : String = 'boolean' {readonly}
+Object : String = 'object' {readonly}
«abstract»
StringFormat
+__default : String = 'any' {readonly}
+Any : String = 'any' {readonly}
+Date : String = 'date' {readonly}
+DateTime : String = 'datetime' {readonly}
+Password : String = 'password' {readonly}
+Byte : String = 'byte' {readonly}
+Binary : String = 'binary' {readonly}
+isValidName(name : String, strict : Boolean = False) : Boolean
+isValidValue(value : Mixed) : Boolean
«abstract»
NumberFormat
+__default : String = 'any' {readonly}
+Any : String = 'any' {readonly}
+Float : String = 'float' {readonly}
+Integer : String = 'integer' {readonly}
«abstract»
EnumBase
-constCacheArray : Mixed[*]
-__construct() : void
+getConstList() : Mixed[*]
+isValidName(name : String, strict : Boolean = False) : Boolean
+isValidValue(value : Mixed) : Boolean
«abstract»
IntegerFormat
+__default : String = 'any' {readonly}
+Any : String = 'any' {readonly}
+Int32 : String = 'int32' {readonly}
+Int64 : String = 'int64' {readonly}

Figure 50-10: *Class Diagram: Enumerations*

Exceptions

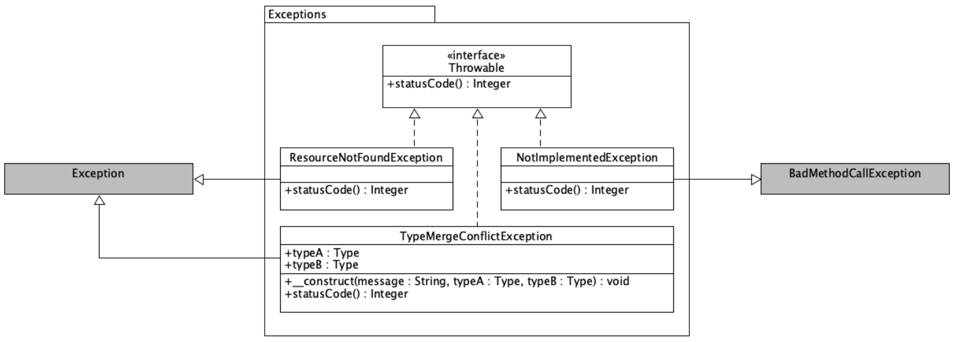


Figure 50-11: *Class Diagram: Exceptions*

Helpers

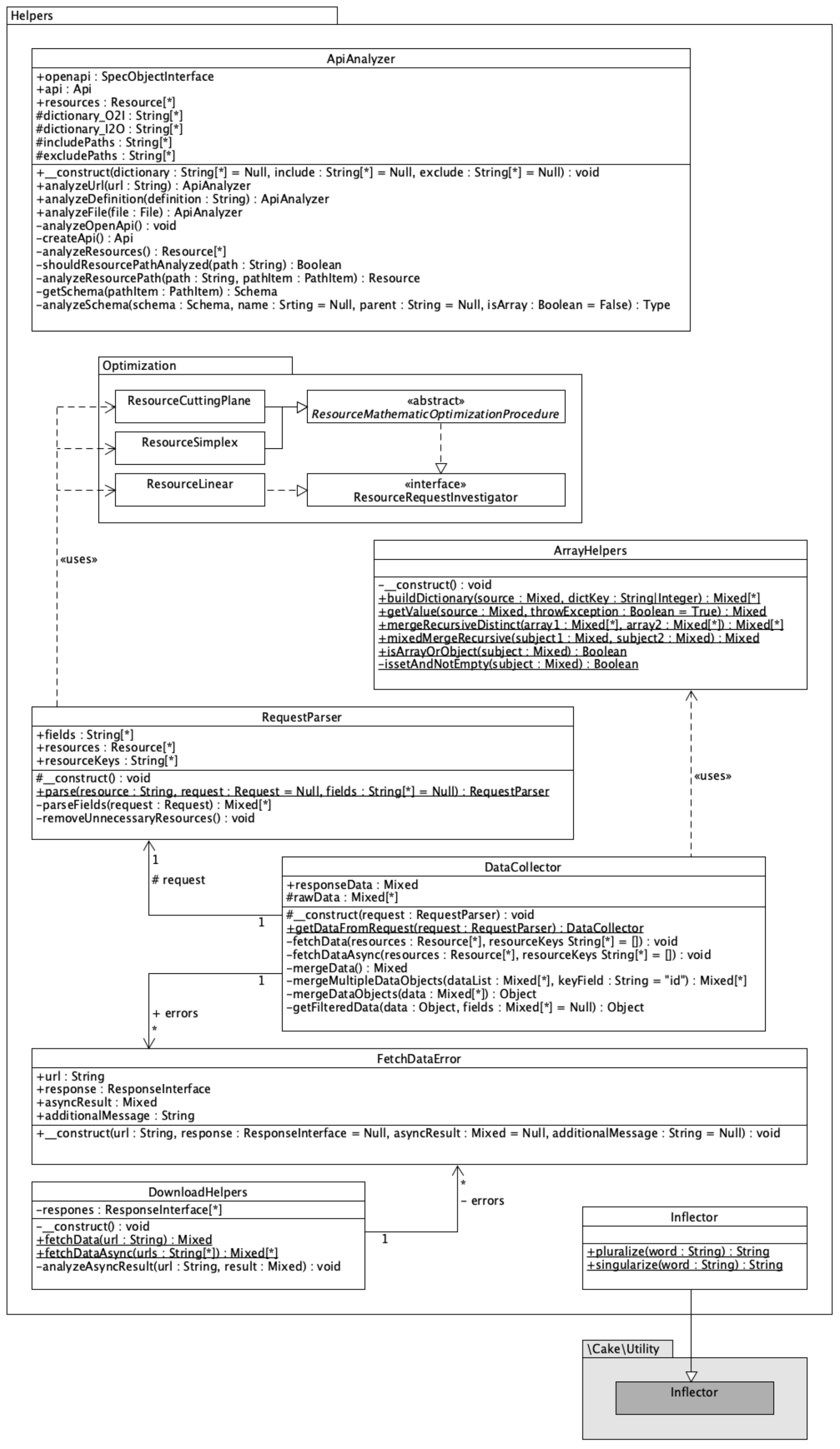

Helpers
ApiAnalyzer
+openapi : SpecObjectInterface
+api : Api
+resources : Resource[*]
#dictionary_O2I : String[*]
#dictionary_I2O : String[*]
#includePaths : String[*]
#excludePaths : String[*]
+__construct(dictionary : String[*] = Null, include : String[*] = Null, exclude : String[*] = Null) : void
+analyzeUrl(url : String) : ApiAnalyzer
+analyzeDefinition(definition : String) : ApiAnalyzer
+analyzeFile(file : File) : ApiAnalyzer
-analyzeOpenApi() : void
-createApi() : Api
-analyzeResources() : Resource[*]
-shouldResourcePathAnalyzed(path : String) : Boolean
-analyzeResourcePath(path : String, pathItem : PathItem) : Resource
-getSchema(pathItem : PathItem) : Schema
-analyzeSchema(schema : Schema, name : Srting = Null, parent : String = Null, isArray : Boolean = False) : Type
Optimization
ResourceCuttingPlane
ResourceSimplex
ResourceLinear
«abstract»
ResourceMathematicOptimizationProcedure
«interface»
ResourceRequestInvestigator
«uses»
ArrayHelpers
-__construct() : void
+buildDictionary(source : Mixed, dictKey : String|Integer) : Mixed[*]
+getValue(source : Mixed, throwException : Boolean = True) : Mixed
+mergeRecursiveDistinct(array1 : Mixed[*], array2 : Mixed[*]) : Mixed[*]
+mixedMergeRecursive(subject1 : Mixed, subject2 : Mixed) : Mixed
+isArrayOrObject(subject : Mixed) : Boolean
-issetAndNotEmpty(subject : Mixed) : Boolean
RequestParser
+fields : String[*]
+resources : Resource[*]
+resourceKeys : String[*]
#__construct() : void
+parse(resource : String, request : Request = Null, fields : String[*] = Null) : RequestParser
-parseFields(request : Request) : Mixed[*]
-removeUnnecessaryResources() : void
«uses»
1
# request
1
DataCollector
+responseData : Mixed
#rawData : Mixed[*]
#__construct(request : RequestParser) : void
+getDataFromRequest(request : RequestParser) : DataCollector
-fetchData(resources : Resource[*], resourceKeys String[*] = []) : void
-fetchDataAsync(resources : Resource[*], resourceKeys String[*] = []) : void
-mergeData() : Mixed
-mergeMultipleDataObjects(dataList : Mixed[*], keyField : String = "id") : Mixed[*]
-mergeDataObjects(data : Mixed[*]) : Object
-getFilteredData(data : Object, fields : Mixed[*] = Null) : Object
1
+ errors
*
FetchDataError
+url : String
+response : ResponseInterface
+asyncResult : Mixed
+additionalMessage : String
+__construct(url : String, response : ResponseInterface = Null, asyncResult : Mixed = Null, additionalMessage : String = Null) : void
DownloadHelpers
-respones : ResponseInterface[*]
-__construct() : void
+fetchData(url : String) : Mixed
+fetchDataAsync(urls : String[*]) : Mixed[*]
-analyzeAsyncResult(url : String, result : Mixed) : void
*
- errors
1
Inflector
+pluralize(word : String) : String
+singularize(word : String) : String
\Cake\Utility
Inflector

Figure 50-12: *Class Diagram: Helpers*

Optimization

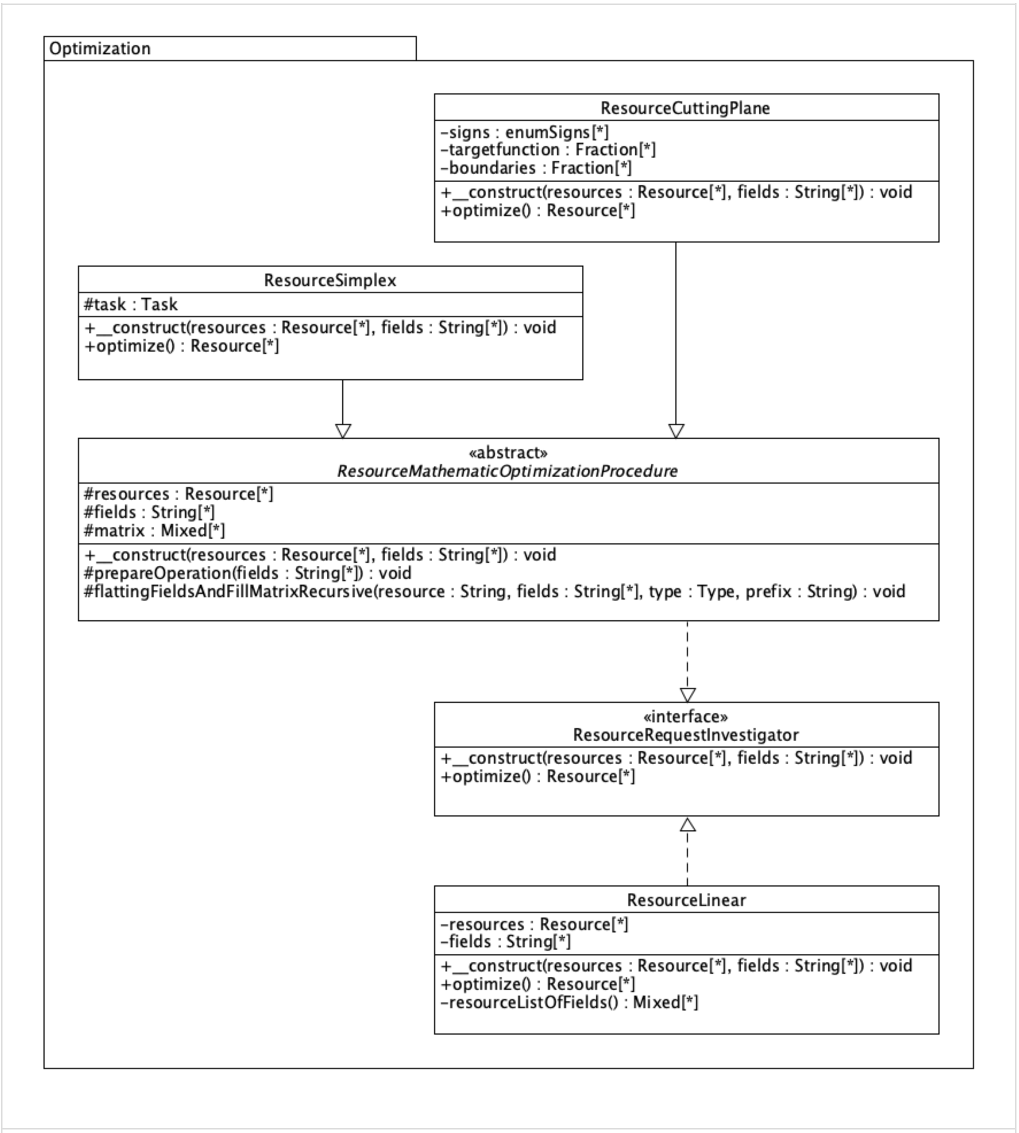

Optimization
ResourceCuttingPlane
-signs : enumSigns[*]
-targetfunction : Fraction[*]
-boundaries : Fraction[*]
+__construct(resources : Resource[*], fields : String[*]) : void
+optimize() : Resource[*]
ResourceSimplex
#task : Task
+__construct(resources : Resource[*], fields : String[*]) : void
+optimize() : Resource[*]
«abstract»
ResourceMathematicOptimizationProcedure
#resources : Resource[*]
#fields : String[*]
#matrix : Mixed[*]
+__construct(resources : Resource[*], fields : String[*]) : void
#prepareOperation(fields : String[*]) : void
#flattingFieldsAndFillMatrixRecursive(resource : String, fields : String[*], type : Type, prefix : String) : void
«interface»
ResourceRequestInvestigator
+__construct(resources : Resource[*], fields : String[*]) : void
+optimize() : Resource[*]
ResourceLinear
-resources : Resource[*]
-fields : String[*]
+__construct(resources : Resource[*], fields : String[*]) : void
+optimize() : Resource[*]
-resourceListOfFields() : Mixed[*]

Figure 50-13: *Class Diagram: Optimization*

Jobs

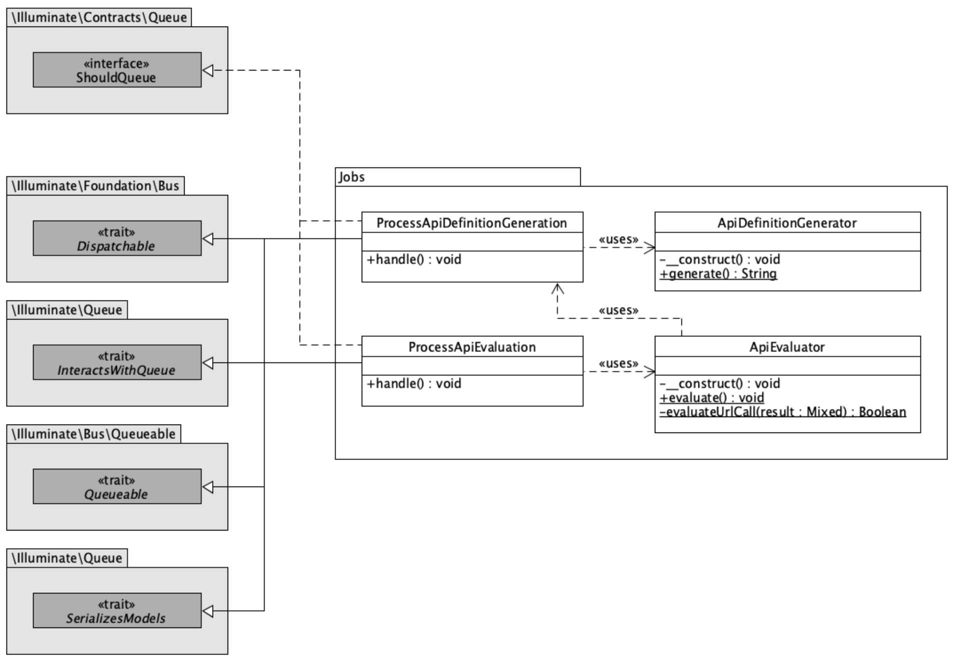


Figure 50-14: *Class Diagram: Jobs*

Commands

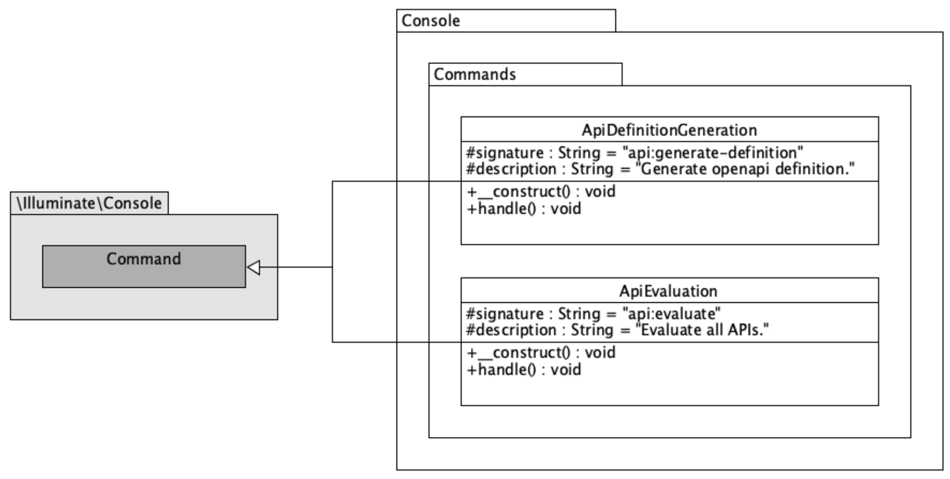

Console
Commands
ApiDefinitionGeneration
#signature : String = "api:generate-definition"
#description : String = "Generate openapi definition."
+__construct() : void
+handle() : void
ApiEvaluation
#signature : String = "api:evaluate"
#description : String = "Evaluate all APIs."
+__construct() : void
+handle() : void
\Illuminate\Console
Command

Figure 50-15: *Class Diagram: Commands*

Migrations

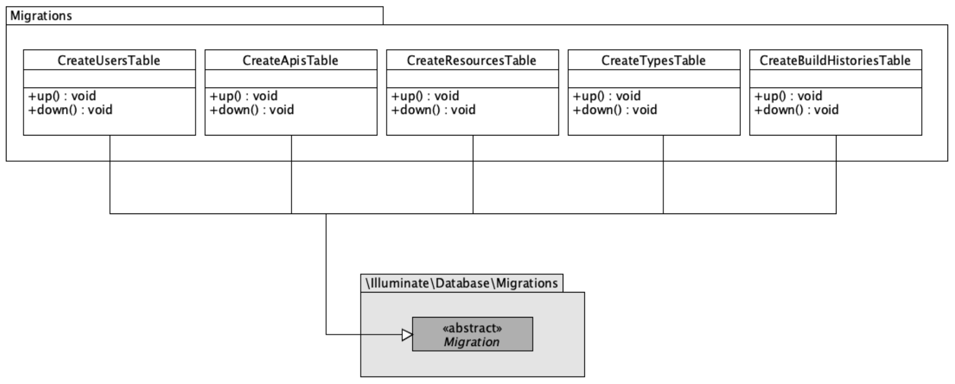


Figure 50-16: *Class Diagram: Migrations*

Implementation

Table of Contents

OpenAPI

> The OpenAPI Specification was donated to the Linux Foundation under the OpenAPI Initiative in 2015. The specification creates a RESTful interface for easily developing and consuming an API by effectively mapping all the resources and operations associated with it.
>
> OpenAPI Specification (formerly Swagger Specification) is an API description format for REST APIs. An OpenAPI file allows you to describe your entire API.

For more information about OpenAPI visit therehomepage.

OpenAPI vs. RAML

Why do we use OpenAPI and not something similar like RAML?

Even if the basic description of RAML fits the project more than OpenAPI does, OpenAPI is more common and has a bigger community.

To see some commons and some differences look as this german article:OpenAPI vs. RAML (german article).

Tools

One of the biggest advantages of OpenAPI is the variety of free tools. Some tools for example converts RAML API specifications to OpenAPI specifications. Some of these tools are listed here.

OpenAPI also has a nice UI to design your API. The providedOnline Editor or its source code on GitHub are public available.

The PHP basic support to read and write OpenAPI definitions / specifications is also free available.

> Read and write OpenAPI yaml/json files and make the content accessible in PHP objects.

It's basicaly a json/yaml parser.

GraphQL

Some of the additional feature requests were to supportGraphQL.

> GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data.
>
> **graphql-php** is a feature-complete implementation of GraphQL specification in PHP (5.5+, 7.0+).

The documentation could be found here. The GitHub repository could be found here.

UML

Unified Modeling Language is a "specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems."

Tools

The tool used to visualize the application / the system isUMLet. Its GitHub repository could be found here. UMLet has also an Online Editor available.

PHP Traits

As Mojmír Němeček wrote in his answer about how to correctly visualize PHPTraits in UML class diagrams, traits will be handled like inheritances.

Logic packages

It seems natural to understand a package in class diagrams as its own namespace. However logic packages are packages which do not have its own namespace. To keep a visual difference between these package types we create a new package symbol for logic packages.

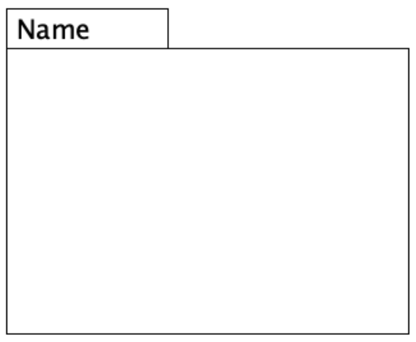
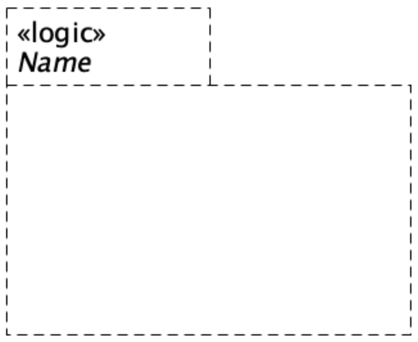
	

Figure 60-1: *Class Diagram: Package*	Figure 60-2: *Class Diagram: Logic Package*

Plural <-> Singular

To convert a word from singular to plural or vice versa we use a slightly modified version of the CakePHP Inflector.

Design and implementation of a RESTful API gateway for the consolidation of existing distributed resources.

Table of Contents

RESTful API Gateway

This system was developed as part of a bachelor thesis.

> Design and implementation of a RESTful API gateway for the consolidation of existing distributed resources.

Table of Contents

Situation

Nowadays, software development is moving away from building one large monolith. Many small self-contained micro services are preferred. In some constellations, the problem now arises that an information object is distributed over several microservices. In other cases, the required information is distributed over several systems.
This bachelor thesis topic deals with this problem limited to the distributed retrieval of information via RESTful APIs.

Objective

A central unit is to be created where RESTful APIs can be registered. All requests, in particular GET requests, are to be made via this central unit. In the following part this central unit is called RESTful API Gateway.

Delimitation

Within the scope of this work, the central unit, RESTful API Gateway, should only support queries, i.e. `GET` requests. Thus no data can be manipulated (`PUT` / `PATCH`), created (`POST`) or deleted (`DELETE`) via the central unit RESTful API Gateway.
The RESTful API Gateway also does not check or ensure the data consistency of the various APIs.

Diagrams

Unfortunately, all diagrams were created in German. Hopefully the diagrams will be translated in the future.

Example

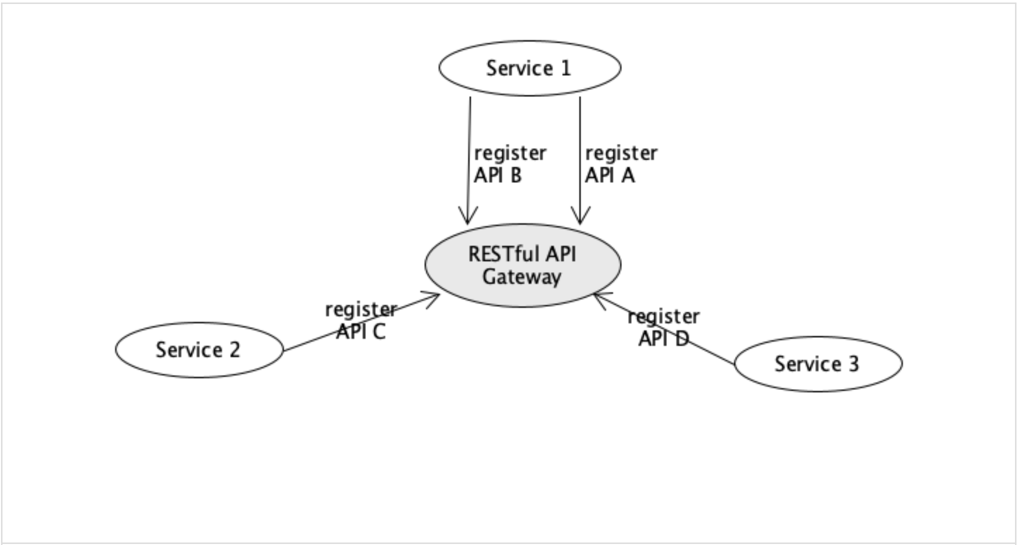


Figure 01: *Example Constellation of the RESTful API Gateway (german)*

The following table illustrates the different services of the different APIs.

Service 1		Service 2	Service 3

`User: {`
`name,`
`address`
`}`
	`Car: {`
	`name,`
	`image`
	`}`
		`User: {`
		`name,`
		`email`
		`}`
			`User: {`
			`birthday,`
			`image`
			`}`

Table 01: *Information about services, their RESTful APIs and data in this example*

Via the RESTful API gateway user data can now be queried centrally. For example, if you want to query the name, e-mail and address of a user the following query should be possible:

`GET /users/{user_id}?fields=name|email|address` (1)

To get all user data:

`GET /users` (2)

With the help of the fragments in the request (1): name, e-mail and address, the RESTful API can find out which APIs have to be requested with the help of the registered APIs and their descriptions. In this case these would be API A and API C. Then the responses are returned in combination.

```
User: {
    name,
    address,
    email
}
```

The request (2) would address API A, API C and API D accordingly and return its combined response.

```
User: {
    name,
    address,
    email,
    birthday,
    image
}
```

Use-Case Diagram

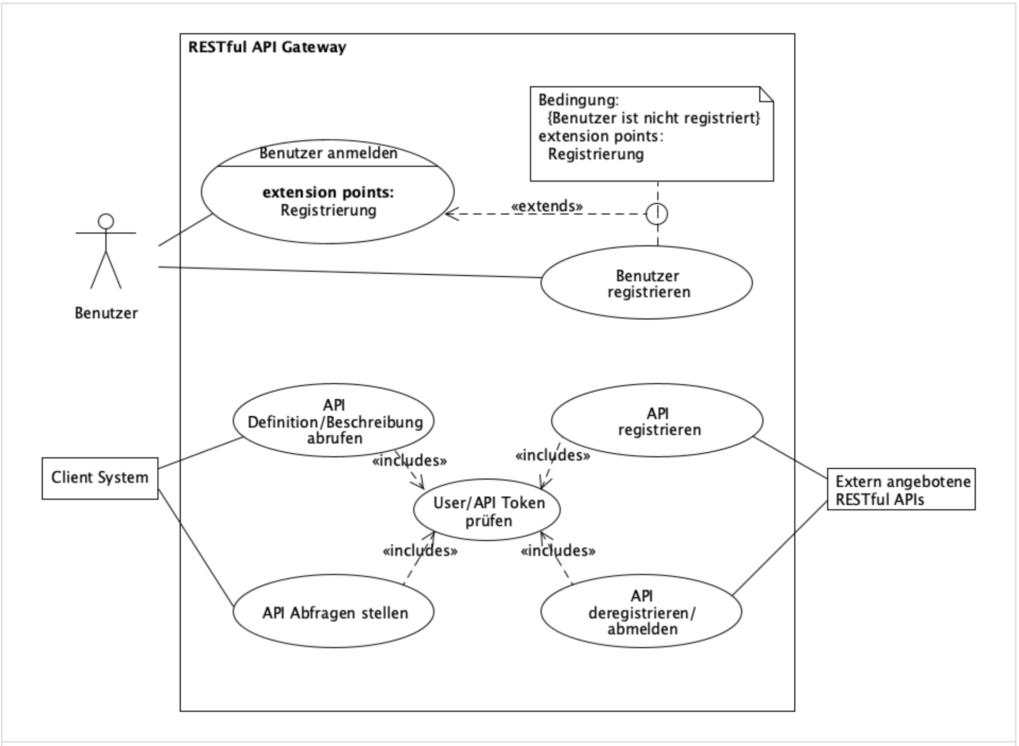


Figure 02: *Use-Case Diagram (german)*