Concept

Roman Richert

Version 1.0.0, 2025-05-19

Table of Contents

Project Overview . . . 1
- Technology Stack . . . 2
  - Backend . . . 2
  - Frontend . . . 2
- Features . . . 3
- Premium Formula . . . 4
- Architecture . . . 5
- Default Ports and Access . . . 6
- Development Notes . . . 7

Insurance Request API . . . 8
- Overview . . . 9
- POST `/api/insurance-request` . . . 10
  - Request . . . 10
  - Response `200 OK` . . . 10
  - Response `400 Bad Request` . . . 10
- GET `/api/insurance-request/{id}` . . . 11
  - Example . . . 11
  - Response `200 OK` . . . 11
- GET `/api/insurance-request?page=0&size=20` . . . 12
  - Response . . . 12
- DELETE `/api/insurance-request/{id}` . . . 13
  - Example . . . 13
  - Response . . . 13
- Notes . . . 14

Postcode Registry API . . . 15
- Overview . . . 16
- GET `/api/postcodes` . . . 17
  - Query Parameters . . . 17
  - Example Request . . . 17
  - Example Response `200 OK` . . . 17
- Notes . . . 18

Region Factors API . . . 19
- Overview . . . 20
- GET `/api/region-factors` . . . 21
  - Response `200 OK` . . . 21
- Notes . . . 22

Vehicle Factor API . . . 23
- Overview . . . 24

GET `/api/vehicle-factors` . . . 25
Example Request . . . 25
Example Response `200 OK` . . . 25
Notes . . . 26

Project Overview

Technology Stack

Backend

- Quarkus 3 (Jakarta EE, RESTEasy Reactive)
- Kotlin
- Hibernate ORM + Panache
- PostgreSQL
- Flyway (database migration)
- CSV import via Apache Commons CSV
- OpenAPI (Swagger UI)
- CORS and Exception Handling support
- SSL via self-signed keystore

Frontend

- Nuxt 3
- Vue 3
- PrimeVue
- TypeScript + Pinia for state management

Features

- Premium calculation via input form or API
- Historical log of all requests with full data
- Dynamic CSV import for postcode mapping
- Factor configuration via application config
- Global error handling and JSON error structure
- Request logging with interceptor
- Full HTTPS support with keystore (self-signed)
- Manual keystore renewal script:

```
cd ../../../../../
chmod +x ./update-keystore.sh
./update-keystore.sh
```

- Full-stack startup with `docker-compose` via

```
cd ../../../../../
chmod +x ./build-and-run.sh
./build-and-run.sh
```

- Database migrations via Flyway
- Modular and extensible architecture

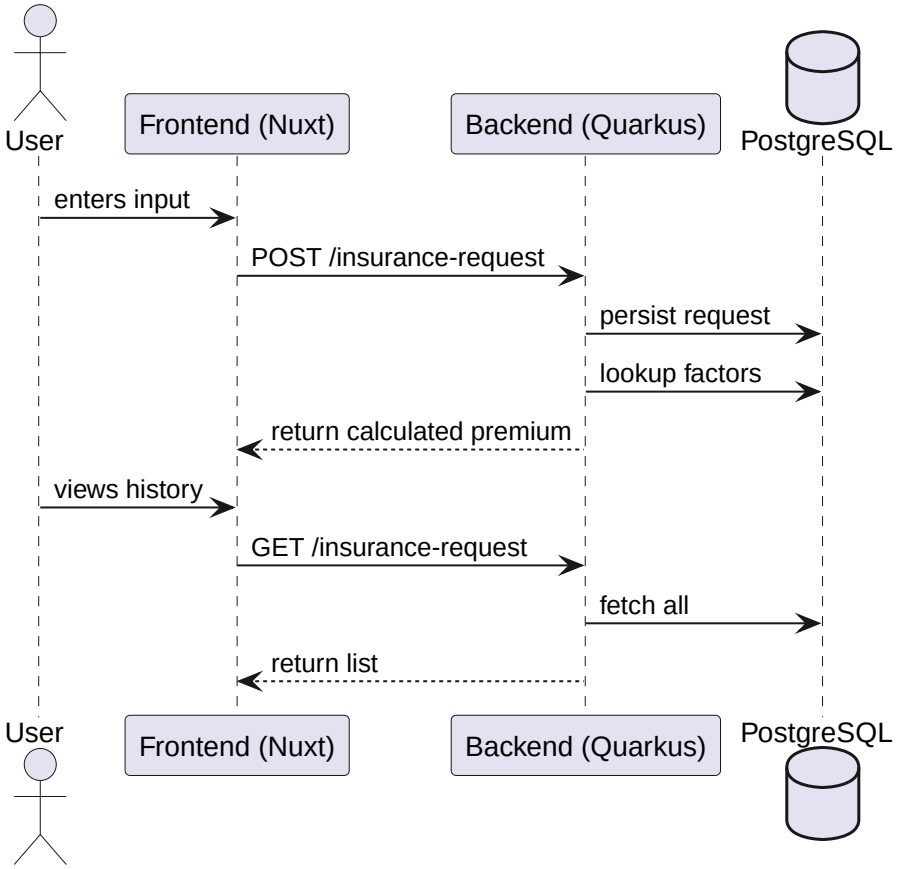
Premium Formula

The premium is calculated using:

```
premium = mileageFactor * vehicleFactor * regionFactor
```

Each component is resolved based on user input and imported configuration data.

Architecture



Default Ports and Access

When running the application with the default configuration:

Service	URL / Host
Frontend (Nuxt)	http://localhost:3000
Backend (HTTP)	http://localhost:8080
Backend (HTTPS)	https://localhost:8443
Swagger	/q/swagger
PostgreSQL (in container)	db:5432
PostgreSQL (host access)	localhost:15432

Development Notes

- CORS support is enabled via a JAX-RS response filter (`CorsFilter.kt`)
- CSV files for postcode data are placed in `src/main/resources/data`
- All user-facing errors follow a uniform structure (`ErrorResponse`)
- Vehicle and region factors are injected via config mapping
- SSL enabled by default; replace `keystore.p12` manually if needed

Insurance Request API

Overview

This endpoint allows submitting insurance calculation requests, retrieving past entries, and deleting records.

Base path: `/api/insurance-request`

POST `/api/insurance-request`

Field	Description
postalCode	German postal code (e.g. `10115`)
vehicleType	Vehicle type (`CAR`, `SUV`, `TRUCK`, `MOTORCYCLE`)
annualMileage	Annual mileage in kilometers (positive integer)

Request

```
{
  "postalCode": "50667",
  "vehicleType": "SUV",
  "annualMileage": 15000
}
```

Response `200 OK`

```
{
  "id": "d01a4e35-01c9-4343-b09f-b9bd6eadbbdf",
  "postalCode": "50667",
  "vehicleType": "SUV",
  "annualMileage": 15000,
  "calculatedPremium": 1.8,
  "createdAt": "2025-05-18T12:00:00"
}
```

Response `400 Bad Request`

Invalid postal code or data format.

GET `/api/insurance-request/{id}`

Returns a single request by its unique ID.

Example

```
GET /api/insurance-request/d01a4e35-01c9-4343-b09f-b9bd6eadbbdf
Accept: application/json
```

Response `200 OK`

Returns the full insurance request data (same structure as POST response).

GET `/api/insurance-request?page=0&size=20`

Returns a paginated list of requests. The default size is 20, page indexing starts at 0.

Response

```
[
  {
    "id": "d01a4e35-01c9-4343-b09f-b9bd6eadbbdf",
    "postalCode": "50667",
    "vehicleType": "CAR",
    "annualMileage": 12000,
    "calculatedPremium": 2.2,
    "createdAt": "2025-05-18T12:00:00"
  }
]
```

DELETE `/api/insurance-request/{id}`

Deletes a request by its ID.

Example

```
DELETE /api/insurance-request/d01a4e35-01c9-4343-b09f-b9bd6eadbbdf
```

Response

```
true
```

Notes

- All endpoints accept and return `application/json`.
- Errors follow the standard `ErrorResponse` format.
- Validation is enforced on input fields (e.g., mileage must be positive).

Postcode Registry API

Overview

Provides filtered, paginated access to postcode records imported via CSV. Each postcode may include regional metadata like city, district, and state.

Base path: `/api/postcodes`

GET `/api/postcodes`

Returns postcodes that match specified filters. All filters are optional; results are paginated.

Query Parameters

Name	Type	Description
`postalCode`	String	Exact match of the postal code (e.g. `50667`)
`state`	String	Two-letter federal state code (e.g. `NRW`)
`region`	String	Optional region name
`district`	String	Optional district name
`city`	String	Optional city name
`quarter`	String	Optional city quarter
`page`	Int	Page number (zero-based), default `0`
`size`	Int	Page size, default `20`

Example Request

```
GET /api/postcodes?postalCode=50667&state=NRW&page=0&size=10
```

Example Response `200 OK`

```
[
  {
    "postalCode": "50667",
    "state": "NRW",
    "region": "Köln",
    "district": "Innenstadt",
    "city": "Köln",
    "quarter": "Altstadt-Nord"
  }
]
```

Notes

- Missing filters will be ignored.
- Results are paged.
- All data is read-only and populated via CSV import at startup.

Region Factors API

Overview

Region factors influence the final premium and are linked to German states. This endpoint provides read-only access to all region factors in the system.

Base path: `/api/region-factors`

GET `/api/region-factors`

Returns all configured region-based premium factors.

Response `200 OK`

```
[
  {
    "state": "NRW",
    "factor": 1.2
  },
  {
    "state": "BE",
    "factor": 0.9
  }
]
```

Notes

- The list is static and reflects the current configuration / state in the database.
- Each entry maps a two-letter federal state code (`state`) to a numeric factor (`factor`).
- This endpoint is useful for diagnostics or UI configuration.

Vehicle Factor API

Overview

Returns the list of all configured vehicle types and their associated factors. These factors are defined via the application config and used during premium calculation.

Base path: `/api/vehicle-factors`

GET `/api/vehicle-factors`

Returns the static list of all supported vehicle types and their factors.

Example Request

```
GET /api/vehicle-factors
```

Example Response `200 OK`

```
[
  { "vehicleType": "CAR", "factor": 1.0 },
  { "vehicleType": "SUV", "factor": 1.2 },
  { "vehicleType": "TRUCK", "factor": 1.5 },
  { "vehicleType": "MOTORCYCLE", "factor": 0.8 }
]
```

Notes

- This endpoint is read-only.
- Vehicle factors are loaded from configuration (`application.yml`) on startup.
- Used internally by the premium calculation service.