IPA-Anhang

von

Andrei Mititelu

2024

Inhalt

Informationen

In diesem Anhang werden Codestücke dokumentiert, die selbst geschrieben wurden.

Selbsterstellte Codeabschnitte sind durch eine gelbe Markierung hervorgehoben.

Sollte eine Datei **vollständig selbst erstellt** oder umfassend überarbeitet worden sein, wird dies durch eine gelbe **Markierung des Headers** kenntlich gemacht.

Bei umfangreichen Dateien werden nicht eigenständig entwickelte Codeabschnitte, die vor oder nach dem dokumentierten Code liegen, durch die Platzhalter (...) angezeigt.

Backend

application.yml

```
(…)

error:
  whitelabel:
    enabled: false
  include-message: always

(…)
```

LifeInsuranceCalculationController.java

```
package com.generali.ovweb.controller;

import com.generali.fosoft.model.ProductCalculationResponse;
import com.generali.ovweb.model.dto.LifeInsuranceOfferDto;
import com.generali.ovweb.model.mapper.DtoMapper;
import com.generali.ovweb.service.LifeInsuranceCalculationService;
import java.util.List;
import java.util.stream.Collectors;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@Slf4j
@RestController
@RequestMapping("/get-offer-details")
public class LifeInsuranceCalculationController {

  private final LifeInsuranceCalculationService calculatorService;

  @Autowired
  public LifeInsuranceCalculationController(LifeInsuranceCalculationService
calculatorService) {
    this.calculatorService = calculatorService;
  }

  @GetMapping("/{id}")
  public List<LifeInsuranceOfferDto> getOfferDetails(@PathVariable Long id)
{
    log.atInfo()
        .setMessage("Getting offer details for request ID:" + id)
        .addKeyValue("requestId", id)
        .log();

    List<ProductCalculationResponse> responses =
calculatorService.calculateOffer(id);
    List<LifeInsuranceOfferDto> results =

responses.stream().map(DtoMapper.INSTANCE::mapToDto).collect(Collectors.toL
ist());
    return results;
  }
}
```

LifeInsuranceRequestController.java

```
package com.generali.ovweb.controller;

import com.generali.fosoft.model.ProductCalculationResponse;
import com.generali.fosoft.model.Status;
import com.generali.ovweb.model.dto.LifeInsuranceRequestPostDto;
import
com.generali.ovweb.model.dto.LifeInsuranceRequestWithCustomerInfoGetDto;
import
com.generali.ovweb.model.dto.LifeInsuranceRequestWithCustomerInfoPostDto;
import com.generali.ovweb.model.mapper.DtoMapper;
import com.generali.ovweb.service.LifeInsuranceCalculationService;
import com.generali.ovweb.service.LifeInsuranceRequestService;
import io.swagger.v3.oas.annotations.Operation;
import java.util.List;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import
org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.security.oauth2.jwt.Jwt;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.server.ResponseStatusException;

@Slf4j
@RestController
@RequestMapping("/lifeinsurance-request")
public class LifeInsuranceRequestController {

  private final LifeInsuranceRequestService service;
  private final LifeInsuranceCalculationService calculatorService;

  @Autowired
  public LifeInsuranceRequestController(
      LifeInsuranceRequestService service, LifeInsuranceCalculationService
calculatorService) {
    this.service = service;
    this.calculatorService = calculatorService;
  }

  @Operation(summary = "Get all LifeInsuranceRequests by auth ID")
  @GetMapping
  public List<LifeInsuranceRequestWithCustomerInfoGetDto>
getAllLifeInsuranceRequests(
      @AuthenticationPrincipal Jwt principal) {
    var userId = principal.getSubject();
    log.atInfo()
        .setMessage("Request to get all life insurance requests for user
ID:" + userId)
        .addKeyValue("userKeycloakId", userId)
        .log();
    return service.getAllInsuranceRequests(userId).stream()
        .map(DtoMapper.INSTANCE::mapToGetDto)
        .toList();
  }
@Operation(summary = "Create a LifeInsuranceRequest")
@PostMapping
public LifeInsuranceRequestWithCustomerInfoGetDto
```

```
createLifeInsuranceRequest(
    @RequestBody LifeInsuranceRequestPostDto request,
@AuthenticationPrincipal Jwt principal) {
  var keyCloakId = principal.getSubject();
  log.atInfo()
      .setMessage("Creating insurance request for userKeyCloakId:" +
keyCloakId)
      .addKeyValue("userKeycloakId", keyCloakId)
      .log();

  List<ProductCalculationResponse> validityResponses =
      calculatorService.checkOfferValidity(
          request.customerId(), DtoMapper.INSTANCE.mapToEntity(request));

  for (ProductCalculationResponse response : validityResponses) {
    if (response.getResult().getFirst().getStatus() != Status._0) {
      throw new ResponseStatusException(
          HttpStatus.BAD_REQUEST,

response.getResult().getFirst().getError().getFirst().getValue());
    }
  }
```

(…)

LifeInsuranceRequestWithCustomerInfo.java

```
(…)

  @Column(nullable = false)
  private String country;

  @Column(nullable = false)
  private Integer policyPeriod;

  @Column(nullable = false)
  private Integer vs1;

  @Column(nullable = true)
  private Integer vs2;

  @Column(nullable = true)
  private Integer vs3;

  @Enumerated(EnumType.ORDINAL)
  @Column(nullable = false)
  private PremiumInstallmentsYear pza;

  @Enumerated(EnumType.STRING)
  @Column(nullable = false)
  private PensionPillar pensionPillar;

  @Column(nullable = false)
  private String vsArt;

  @Column(nullable = false)
  private Boolean premiumWaiver;

  @ManyToOne(cascade = CascadeType.PERSIST)
  @JoinColumn(name = "sponsor_id", nullable = false)
  private User sponsor;
}
```

LifeInsuranceOfferDto.java

```
package com.generali.ovweb.model.dto;

import com.generali.fosoft.model.Status;
import com.generali.ovweb.model.enums.PensionPillar;
import com.generali.ovweb.model.enums.PremiumInstallmentsYear;
import io.swagger.v3.oas.annotations.media.Schema;
import jakarta.validation.constraints.NotNull;
import java.math.BigDecimal;
import java.util.List;
import lombok.Getter;

public record LifeInsuranceOfferDto(
    @NotNull Status status, OfferDetail offerDetail, @NotNull List<String>
error) {

  public record OfferDetail(
      @NotNull List<String> overview,
      @NotNull List<Calculation> berechnungsliste,
      String graphScaleMaxValue,
      @NotNull List<ItemDescription> praemienzahlartliste,
      @NotNull List<ItemDescription> vorsorgeartliste,
      @NotNull List<AVBItemDescription> avbliste) {}

  public record Calculation(
      @NotNull BigDecimal praemie,
      @NotNull Integer policyPeriod,
      @NotNull Integer vs,
      Integer vsr,
      @NotNull PremiumInstallmentsYear pza,
      PensionPillar pensionPillar,
      @NotNull String vsArt,
      String auspraegung,
      Integer prognoseGarantie,
      Integer prognoseTief,
      Integer prognoseMittel,
      Integer prognoseHoch,
      Integer gewuenschteSumme) {}

  public record ItemDescription(
      @NotNull String key, @NotNull String value, @NotNull String
description) {}

  public record AVBItemDescription(String name, String url) {}

(…)
```

LifeInsuranceRequestPostDto.java

```
package com.generali.ovweb.model.dto;

import com.generali.ovweb.model.enums.PensionPillar;
import com.generali.ovweb.model.enums.PremiumInstallmentsYear;
import jakarta.validation.constraints.NotNull;

public record LifeInsuranceRequestPostDto(
    @NotNull Long customerId,
    @NotNull Integer policyPeriod,
    @NotNull Integer vs1,
    Integer vs2,
    Integer vs3,
    @NotNull PremiumInstallmentsYear pza,
    @NotNull PensionPillar pensionPillar,
    @NotNull String vsArt,
    @NotNull Boolean premiumWaiver) {}
```

LifeInsuranceRequestWithCustomerInfoGetDto.java

```
package com.generali.ovweb.model.dto;

import com.generali.ovweb.model.enums.Gender;
import com.generali.ovweb.model.enums.PensionPillar;
import com.generali.ovweb.model.enums.PremiumInstallmentsYear;
import jakarta.validation.constraints.NotNull;
import java.time.LocalDate;

public record LifeInsuranceRequestWithCustomerInfoGetDto(
    @NotNull Long id,
    @NotNull String firstName,
    @NotNull String lastName,
    @NotNull LocalDate birthdate,
    @NotNull String phoneNumber,
    @NotNull String email,
    @NotNull Gender gender,
    @NotNull Boolean smoker,
    @NotNull String houseNumber,
    @NotNull String streetName,
    @NotNull String plz,
    @NotNull String city,
    @NotNull String country,
    @NotNull Integer policyPeriod,
    @NotNull Integer vs1,
    Integer vs2,
    Integer vs3,
    @NotNull PremiumInstallmentsYear pza,
    @NotNull PensionPillar pensionPillar,
    @NotNull String vsArt,
    @NotNull Boolean premiumWaiver,
    @NotNull Long sponsorId) {}
```

LifeInsuranceRequestWithCustomerInfoPostDto.java

```
package com.generali.ovweb.model.dto;

import com.generali.ovweb.model.enums.Gender;
import com.generali.ovweb.model.enums.PensionPillar;
import com.generali.ovweb.model.enums.PremiumInstallmentsYear;
import jakarta.validation.constraints.NotNull;
import java.time.LocalDate;

public record LifeInsuranceRequestWithCustomerInfoPostDto(
    @NotNull String firstName,
    @NotNull String lastName,
    @NotNull LocalDate birthdate,
    @NotNull String phoneNumber,
    @NotNull String email,
    @NotNull Gender gender,
    @NotNull Boolean smoker,
    @NotNull String houseNumber,
    @NotNull String streetName,
    @NotNull String plz,
    @NotNull String city,
    @NotNull String country,
    @NotNull Integer policyPeriod,
    @NotNull Integer vs1,
    Integer vs2,
    Integer vs3,
    @NotNull PremiumInstallmentsYear pza,
    @NotNull PensionPillar pensionPillar,
    @NotNull String vsArt,
    @NotNull Boolean premiumWaiver) {}
```

DtoMapper.java

```
(…)

@Mapping(source = "pza", target = "pza")
@Mapping(source = "vs1", target = "vs1")
@Mapping(source = "vs2", target = "vs2")
@Mapping(source = "vs3", target = "vs3")
public abstract LifeInsuranceRequestWithCustomerInfo mapToEntity(
    LifeInsuranceRequestWithCustomerInfoPostDto dto);

(…)


(…)

var offerDetail =
      new LifeInsuranceOfferDto.OfferDetail(
          calculation.getOverview(),
          berechnungsliste,
          calculation.getGraphScaleMaxValue(),
          praemienzahlartliste,
          vorsorgeartliste,
          avbListe);

  return new LifeInsuranceOfferDto(

LifeInsuranceOfferDto.Status.fromValue(calculation.getStatus().getValue()),
      offerDetail,

calculation.getError().stream().map(com.generali.fosoft.model.Error::getVal
ue).toList());
}

(…)
```

LifeInsuranceCalculationService.java

```
package com.generali.ovweb.service;

import com.generali.fosoft.model.*;
import com.generali.fosoft.model.Error;
import com.generali.ovweb.model.Customer;
import com.generali.ovweb.model.LifeInsuranceRequestWithCustomerInfo;
import com.generali.ovweb.model.enums.Gender;
import com.generali.ovweb.persistence.CustomerRepository;
import com.generali.ovweb.persistence.LifeInsuranceRequestRepository;
import com.generali.ovweb.rest.client.FoSoftClient;
import io.micrometer.tracing.annotation.NewSpan;
import jakarta.persistence.EntityNotFoundException;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Slf4j
@Service
public class LifeInsuranceCalculationService {

  private final LifeInsuranceRequestRepository
lifeInsuranceRequestRepository;
  private final FoSoftClient foSoftClient;
  private final CustomerRepository customerRepository;

  @Autowired
  public LifeInsuranceCalculationService(
      FoSoftClient foSoftClient,
      CustomerRepository customerRepository,
      LifeInsuranceRequestRepository lifeInsuranceRequestRepository) {
    this.foSoftClient = foSoftClient;
    this.lifeInsuranceRequestRepository = lifeInsuranceRequestRepository;
    this.customerRepository = customerRepository;
  }

  @NewSpan("calculateOffer")
  public List<ProductCalculationResponse> calculateOffer(Long id) {
    return lifeInsuranceRequestRepository
        .findById(id)
        .map(
            request -> {
              List<ProductCalculationResponse> responses = new
ArrayList<>();
              Integer[] vsValues = {request.getVs1(), request.getVs2(),
request.getVs3()};
              for (Integer vs : vsValues) {
                if (vs != null) {
                  log.atInfo()
                      .setMessage("Mapping and calculating offer with id :"
+ id)
                      .addKeyValue("requestId", id)
                      .log();
                  ProductCalculationRequest foSoftRequest =

LifeInsuranceCalculationService.mapToFoSoftRequest(request, vs);
                  ProductCalculationResponse response =
```

```
                        foSoftClient
                            .calculateLifeInsurance(Language.DE,
"01.02.2024", foSoftRequest)
                            .getBody();
                    responses.add(response);
                  }
                }
                return responses;
              })
          .orElseThrow(
              () -> {
                var message = "Cannot find request with id " + id + " to
calculate offer";
                log.atError().setMessage(message).addKeyValue("requestId",
id).log();
                return new RuntimeException(message);
              });
  }

  @NewSpan("checking offer validity")
  public List<ProductCalculationResponse> checkOfferValidity(
      Long customerId, LifeInsuranceRequestWithCustomerInfo request) {

    log.atInfo()
        .setMessage("Validating insurance request for customer with id " +
customerId)
        .addKeyValue("customerId", customerId)
        .log();

    Customer customer =
        customerRepository
            .findById(customerId)
            .orElseThrow(
                () -> new EntityNotFoundException("Customer not found with
ID: " + customerId));

    List<ProductCalculationResponse> responses = new ArrayList<>();
    Integer[] vsValues = {request.getVs1(), request.getVs2(),
request.getVs3()};

    for (Integer vs : vsValues) {
      if (vs != null) {
        try {
          ProductCalculationRequest foSoftRequest =
              LifeInsuranceCalculationService.mapToFoSoftRequest(
                  addCustomerInfoToRequest(request, customer), vs);
          ProductCalculationResponse response =
              foSoftClient
                  .calculateLifeInsurance(Language.DE, "01.02.2024",
foSoftRequest)
                  .getBody();
          log.atInfo()
              .setMessage("Validating offer with vs value :" + vs)
              .addKeyValue("vs", vs)
              .log();
          if (response == null
              || response.getResult() == null
              || response.getResult().get(0).getStatus() != Status._0) {
            Error error = new Error();
            String errorMessage = "VS Summe: " + vs + ":";
            var errorDescription =
```

```
                response != null
                        && response.getResult() != null
                        && response.getResult().getFirst() != null
                        && response.getResult().getFirst().getError() !=
null
                        && response.getResult().get(0).getError().get(0) !=
null
                    ?
response.getResult().get(0).getError().get(0).getValue()
                    : "no error info";
            errorMessage += " Error: " + errorDescription;
            error.setValue(errorMessage);
            ProductCalculation errorCalculation = new ProductCalculation();
            errorCalculation.addErrorItem(error);
            ProductCalculationResponse errorResponse = new
ProductCalculationResponse();
            errorResponse.addResultItem(errorCalculation);
            responses.add(errorResponse);
          } else {
            responses.add(response);
          }
        } catch (Exception e) {
          log.atInfo()
              .setMessage("Validation failed for vs with sum: " + vs)
              .addKeyValue("vs", vs)
              .log();
          Error error = new Error();
          error.setValue("Unexpected error for VS value: " + vs + ". Error:
" + e.getMessage());
          ProductCalculation errorCalculation = new ProductCalculation();
          errorCalculation.addErrorItem(error);
          ProductCalculationResponse errorResponse = new
ProductCalculationResponse();
          errorResponse.addResultItem(errorCalculation);
          responses.add(errorResponse);
        }
      }
    }
    return responses;
  }

  static LifeInsuranceRequestWithCustomerInfo addCustomerInfoToRequest(
      LifeInsuranceRequestWithCustomerInfo request, Customer customer) {
    request.setBirthdate(customer.getBirthdate());
    request.setGender(customer.getGender());
    request.setSmoker(customer.getSmoker());
    request.setPlz(customer.getAddress().getPlz());
    request.setCity(customer.getAddress().getCity());
    return request;
  }

  @NewSpan("mapping to Fosoft Request")
  static ProductCalculationRequest mapToFoSoftRequest(
      LifeInsuranceRequestWithCustomerInfo request, Integer vs) {
    var targetFormat = DateTimeFormatter.ofPattern("dd.MM.yyyy");
    var formattedDate = request.getBirthdate().format(targetFormat);
    var gender = request.getGender().equals(Gender.MALE) ? 1 : 2;

    var requestPayload = new ProductCalculationRequest();
    var personData = new ProductCalculationRequestPersonendaten();
    personData.setGeburtsdatum(formattedDate);
```

```
    personData.setGeschlecht(

ProductCalculationRequestPersonendaten.GeschlechtEnum.fromValue(gender));
    personData.setRaucher(request.getSmoker());
    personData.setPLZ(request.getPlz());
    personData.setWohnort(request.getCity());
    personData.setNationalitaet("CH");

    var insuranceData = new
ProductCalculationRequestBerechnungslisteInner();
    insuranceData.setDauer(request.getPolicyPeriod());
    insuranceData.setVS(vs);

insuranceData.setPZA(PremiumInstallmentsYear.fromValue(request.getPza().get
Value()));

insuranceData.set3a3b(PensionPillar.fromValue(request.getPensionPillar().ge
tValue()));
    insuranceData.setVsArt(request.getVsArt());
    insuranceData.setPraemienbefreiung(request.getPremiumWaiver());

    requestPayload.setPersonendaten(personData);
    requestPayload.setBerechnungsliste(List.of(insuranceData));
    return requestPayload;
  }
}
```

LifeInsuranceRequestService.java

```
(…)

@NewSpan("saving life insurance request to database")
public LifeInsuranceRequestWithCustomerInfo createLifeInsuranceRequest(
    String keyCloakId, Long customerId,
LifeInsuranceRequestWithCustomerInfo request) {

  log.atInfo()
      .addKeyValue("customerId", customerId)
      .log("Saving insurance request for customer with id: {}",
customerId);

  return customerRepository

(…)



(…)

@NewSpan("mapping to LifeInsuranceRequest")
static LifeInsuranceRequestWithCustomerInfo mapToLifeInsuranceRequest(
    Customer customer, User user, LifeInsuranceRequestWithCustomerInfo
request) {
  return request

(…)
```

V1_0_17__extend_life_insurance_request_entity.sql

```
ALTER TABLE if exists "life_insurance_request"
DROP COLUMN vs;

ALTER TABLE if exists "life_insurance_request"
  ADD COLUMN vs1 INT NOT NULL,
  ADD COLUMN vs2 INT,
  ADD COLUMN vs3 INT;
```

DataPopulationTest.java

(…)

```
var insuranceRequest1 =
    LifeInsuranceRequestWithCustomerInfo.builder()
        .sponsor(user1)
        .policyPeriod(26)
        .vs1(100000)
        .houseNumber("123")
        .streetName("Main Street")
        .plz("12345")
        .city("Example City 1")
        .country("Example Country 1")
        .gender(Gender.FEMALE)
        .pza(PremiumInstallmentsYear.MONTHLY)
        .pensionPillar(PensionPillar.PILLAR_3A)
        .vsArt("konstant")
        .premiumWaiver(true)
        .smoker(false)
        .firstName("John")
        .lastName("Doe")
        .birthdate(LocalDate.of(1990, 1, 15))
        .phoneNumber("+1234567890")
        .email("f@g")
        .build();

var insuranceRequest2 =
    LifeInsuranceRequestWithCustomerInfo.builder()
        .sponsor(user1)
        .policyPeriod(26)
        .vs1(100000)
        .houseNumber("123")
        .city("Example City 1")
        .country("Example Country 1")
        .streetName("Main Street")
        .plz("12345")
        .gender(Gender.MALE)
        .pza(PremiumInstallmentsYear.YEARLY)
        .pensionPillar(PensionPillar.PILLAR_3B)
        .vsArt("konstant")
        .premiumWaiver(false)
        .smoker(true)
        .firstName("John")
        .lastName("Doe")
        .birthdate(LocalDate.of(1990, 1, 15))
        .phoneNumber("+1234567890")
        .email("f@g")
        .build();
```

(…)

DtoMapperTest.java

(…)

```
@Test
void givenLifeInsuranceRequestWithCustomer_whenMapsToGetDto_thenCorrect() {
  var entity =
      LifeInsuranceRequestWithCustomerInfo.builder()
          .id(rndLong())
          .firstName("someFirstName")
          .lastName("someLastName")
          .phoneNumber("1234567890")
          .email("someEmail")
          .houseNumber("1")
          .streetName("street")
          .country("country")
          .sponsor(User.builder().id(rndLong()).build())
          .birthdate(LocalDate.of(2024, 2, 2))
          .gender(Gender.DIVERSE)
          .pensionPillar(PensionPillar.PILLAR_3A)
          .pza(PremiumInstallmentsYear.MONTHLY)
          .smoker(true)
          .plz("12345")
          .city("city")
          .policyPeriod(rndInt())
          .vs1(rndInt())
          .vsArt("vsArt")
          .premiumWaiver(true)
          .build();
  var dto = DtoMapper.INSTANCE.mapToGetDto(entity);
  assertEquals(entity.getId(), dto.id());
  assertEquals(entity.getFirstName(), dto.firstName());
  assertEquals(entity.getLastName(), dto.lastName());
  assertEquals(entity.getPhoneNumber(), dto.phoneNumber());
  assertEquals(entity.getEmail(), dto.email());
  assertEquals(entity.getHouseNumber(), dto.houseNumber());
  assertEquals(entity.getStreetName(), dto.streetName());
  assertEquals(entity.getCountry(), dto.country());
  assertEquals(entity.getSponsor().getId(), dto.sponsorId());
  assertEquals(entity.getBirthdate(), dto.birthdate());
  assertEquals(entity.getGender(), dto.gender());
  assertEquals(entity.getPensionPillar(), dto.pensionPillar());
  assertEquals(entity.getPza(), dto.pza());
  assertEquals(entity.getSmoker(), dto.smoker());
  assertEquals(entity.getPlz(), dto.plz());
  assertEquals(entity.getCity(), dto.city());
  assertEquals(entity.getPolicyPeriod(), dto.policyPeriod());
  assertEquals(entity.getVs1(), dto.vs1());
  assertEquals(entity.getVsArt(), dto.vsArt());
  assertEquals(entity.getPremiumWaiver(), dto.premiumWaiver());
}
```

(…)

(…)

```
@Test
void
givenLifeInsuranceRequestWithCustomerPostDto_whenMapsToEntity_thenCorrect()
```

```
{
  var dto =
      new LifeInsuranceRequestWithCustomerInfoPostDto(
          "someFirstName",
          "someLastName",
          LocalDate.of(2024, 2, 2),
          "1234567890",
          "someEmail",
          Gender.DIVERSE,
          true,
          "1",
          "street",
          "12345",
          "city",
          "country",
          rndInt(),
          rndInt(),
          rndInt(),
          rndInt(),
          PremiumInstallmentsYear.MONTHLY,
          PensionPillar.PILLAR_3B,
          "vsArt",
          false);

  var entity = DtoMapper.INSTANCE.mapToEntity(dto);
  assertNull(entity.getId());
  assertEquals(entity.getFirstName(), dto.firstName());
  assertEquals(entity.getLastName(), dto.lastName());
  assertEquals(entity.getPhoneNumber(), dto.phoneNumber());
  assertEquals(entity.getEmail(), dto.email());
  assertEquals(entity.getHouseNumber(), dto.houseNumber());
  assertEquals(entity.getStreetName(), dto.streetName());
  assertEquals(entity.getCountry(), dto.country());
  assertNull(entity.getSponsor());
  assertEquals(entity.getBirthdate(), dto.birthdate());
  assertEquals(entity.getGender(), dto.gender());
  assertEquals(entity.getPensionPillar(), dto.pensionPillar());
  assertEquals(entity.getPza(), dto.pza());
  assertEquals(entity.getSmoker(), dto.smoker());
  assertEquals(entity.getPlz(), dto.plz());
  assertEquals(entity.getCity(), dto.city());
  assertEquals(entity.getPolicyPeriod(), dto.policyPeriod());
  assertEquals(entity.getVs1(), dto.vs1());
  assertEquals(entity.getVs2(), dto.vs2());
  assertEquals(entity.getVs3(), dto.vs3());
  assertEquals(entity.getVsArt(), dto.vsArt());
  assertEquals(entity.getPremiumWaiver(), dto.premiumWaiver());
}
@Test
void givenLifeInsuranceRequestPostDto_whenMapsToEntity_thenCorrect() {
  var dto =
      new LifeInsuranceRequestPostDto(
          8L,
          10,
          1000,
          1100,
          1200,
          PremiumInstallmentsYear.MONTHLY,
          PensionPillar.PILLAR_3B,
          "vsArt",
          false);
```

```
  var entity = DtoMapper.INSTANCE.mapToEntity(dto);
  assertNull(entity.getId());
  assertNull(entity.getFirstName());
  assertNull(entity.getLastName());
  assertNull(entity.getPhoneNumber());
  assertNull(entity.getEmail());
  assertNull(entity.getHouseNumber());
  assertNull(entity.getStreetName());
  assertNull(entity.getCountry());
  assertNull(entity.getSponsor());
  assertNull(entity.getBirthdate());
  assertNull(entity.getGender());
  assertEquals(entity.getPensionPillar(), dto.pensionPillar());
  assertEquals(entity.getPza(), dto.pza());
  assertNull(entity.getSmoker());
  assertNull(entity.getPlz());
  assertNull(entity.getCity());
  assertEquals(entity.getPolicyPeriod(), dto.policyPeriod());
  assertEquals(entity.getVs1(), dto.vs1());
  assertEquals(entity.getVsArt(), dto.vsArt());
  assertEquals(entity.getPremiumWaiver(), dto.premiumWaiver());
}

(…)



(…)

var dto = DtoMapper.INSTANCE.mapToDto(response);
  assertEquals(dto.error().size(), calculation.getError().size());
  assertEquals(dto.status().getValue(), error.getCode().getValue());
  assertEquals(
      dto.offerDetail().berechnungsliste().getFirst().policyPeriod(),
berechnung.getDauer());
  assertEquals(
      dto.offerDetail().berechnungsliste().getFirst().praemie(),
berechnung.getPraemie());
  assertEquals(dto.offerDetail().berechnungsliste().getFirst().vs(),
berechnung.getVS());
  assertEquals(dto.offerDetail().berechnungsliste().getFirst().vsr(),
berechnung.getVsr());
  assertEquals(dto.offerDetail().berechnungsliste().getFirst().vsArt(),
berechnung.getVsArt());
  assertEquals(

dto.offerDetail().berechnungsliste().getFirst().pensionPillar().getValue(),
      berechnung.get3a3b().getValue());
  assertEquals(
      dto.offerDetail().berechnungsliste().getFirst().pza().getValue(),
      berechnung.getPZA().getValue());
  assertEquals(
      dto.offerDetail().berechnungsliste().getFirst().auspraegung(),
berechnung.getAuspraegung());
  assertEquals(
      dto.offerDetail().berechnungsliste().getFirst().prognoseGarantie(),
      berechnung.getPrognoseGarantie());
  assertEquals(
      dto.offerDetail().berechnungsliste().getFirst().prognoseTief(),
      berechnung.getPrognoseTief());
```

```
  assertEquals(
      dto.offerDetail().berechnungsliste().getFirst().prognoseMittel(),
      berechnung.getPrognoseMittel());
  assertEquals(
      dto.offerDetail().berechnungsliste().getFirst().prognoseHoch(),
      berechnung.getPrognoseHoch());
  assertEquals(
      dto.offerDetail().berechnungsliste().getFirst().gewuenschteSumme(),
      berechnung.getGewuenschteSumme());

  assertEquals(dto.offerDetail().avbliste().getFirst().name(),
aVB.getName());
  assertEquals(dto.offerDetail().avbliste().getFirst().url(),
aVB.getURL());
  assertEquals(
      dto.offerDetail().praemienzahlartliste().size(),
      calculation.getPraemienzahlartliste().size());
  assertEquals(dto.offerDetail().praemienzahlartliste().getFirst().key(),
paymentMode0.getKey());
  assertEquals(
      dto.offerDetail().praemienzahlartliste().getFirst().description(),
      paymentMode0.getDescription());
  assertEquals(
      dto.offerDetail().praemienzahlartliste().getFirst().value(),
paymentMode0.getValue());
  assertEquals(
      dto.offerDetail().vorsorgeartliste().size(),
calculation.getVorsorgeartliste().size());
  assertEquals(dto.offerDetail().vorsorgeartliste().getFirst().value(),
vorsorgeArt0.getValue());
  assertEquals(dto.offerDetail().vorsorgeartliste().getFirst().key(),
vorsorgeArt0.getKey());
  assertEquals(
      dto.offerDetail().vorsorgeartliste().getFirst().description(),
      vorsorgeArt0.getDescription());
}
```

(…)

LifeInsuranceCalculationServiceTest.java

```
package com.generali.ovweb.service;

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.ArgumentMatchers.any;
import static org.mockito.ArgumentMatchers.eq;
import static org.mockito.Mockito.*;

import com.generali.fosoft.model.*;
import com.generali.ovweb.model.Address;
import com.generali.ovweb.model.Customer;
import com.generali.ovweb.model.LifeInsuranceRequestWithCustomerInfo;
import com.generali.ovweb.model.enums.Gender;
import com.generali.ovweb.model.enums.PensionPillar;
import com.generali.ovweb.model.enums.PremiumInstallmentsYear;
import com.generali.ovweb.persistence.CustomerRepository;
import com.generali.ovweb.persistence.LifeInsuranceRequestRepository;
import com.generali.ovweb.rest.client.FoSoftClient;
import jakarta.persistence.EntityNotFoundException;
import java.time.LocalDate;
import java.util.List;
import java.util.Objects;
import java.util.Optional;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

class LifeInsuranceCalculationServiceTest {

  private AutoCloseable closeable;

  @Mock private LifeInsuranceRequestRepository
lifeInsuranceRequestRepository;

  @Mock private CustomerRepository customerRepository;

  @Mock private FoSoftClient foSoftClient;

  @InjectMocks private LifeInsuranceCalculationService
lifeInsuranceCalculationService;

  @BeforeEach
  void initService() {
    closeable = MockitoAnnotations.openMocks(this);
  }

  @AfterEach
  void closeService() throws Exception {
    closeable.close();
  }

  @Test
  void calculateOffer() {
```

```
    var response = new ProductCalculationResponse();
    var entity =
        LifeInsuranceRequestWithCustomerInfo.builder()
            .birthdate(LocalDate.now())
            .gender(Gender.MALE)
            .pensionPillar(PensionPillar.PILLAR_3A)
            .pza(PremiumInstallmentsYear.MONTHLY)
            .vs1(100)
            .build();

when(lifeInsuranceRequestRepository.findById(1L)).thenReturn(Optional.of(en
tity));
    when(foSoftClient.calculateLifeInsurance(
            eq(Language.DE), eq("01.02.2024"),
any(ProductCalculationRequest.class)))
        .thenReturn(new ResponseEntity<>(response, HttpStatus.OK));
    var actual = lifeInsuranceCalculationService.calculateOffer(1L);
    assertEquals(1, actual.size());
    assertEquals(response, actual.getFirst());
  }

  @Test
  void mapToFoSoftRequest() {
    var entity =
        LifeInsuranceRequestWithCustomerInfo.builder()
            .birthdate(LocalDate.of(2024, 2, 2))
            .gender(Gender.DIVERSE)
            .pensionPillar(PensionPillar.PILLAR_3A)
            .pza(PremiumInstallmentsYear.MONTHLY)
            .smoker(true)
            .plz("12345")
            .city("city")
            .policyPeriod(10)
            .vs1(1000)
            .vsArt("vsArt")
            .premiumWaiver(true)
            .build();

    var request =
LifeInsuranceCalculationService.mapToFoSoftRequest(entity,
entity.getVs1());
    assertEquals("02.02.2024",
request.getPersonendaten().getGeburtsdatum());
    assertEquals(
        ProductCalculationRequestPersonendaten.GeschlechtEnum.NUMBER_2,
        request.getPersonendaten().getGeschlecht());
    assertEquals(
        com.generali.fosoft.model.PensionPillar._3A,
        request.getBerechnungsliste().getFirst().get3a3b());
    assertEquals(
        com.generali.fosoft.model.PremiumInstallmentsYear.NUMBER_12,
        request.getBerechnungsliste().getFirst().getPZA());
    assertEquals(10, request.getBerechnungsliste().getFirst().getDauer());
    assertEquals(true, request.getPersonendaten().getRaucher());
    assertEquals("12345", request.getPersonendaten().getPLZ());
    assertEquals("city", request.getPersonendaten().getWohnort());
    assertEquals(1000, request.getBerechnungsliste().getFirst().getVS());
    assertEquals("vsArt",
request.getBerechnungsliste().getFirst().getVsArt());
    assertEquals(true,
request.getBerechnungsliste().getFirst().getPraemienbefreiung());
```

```
    assertEquals("CH", request.getPersonendaten().getNationalitaet());
  }

  @Test
  void addCustomerInfoToRequest() {
    Address address =
        Address.builder()
            .houseNumber("123")
            .streetName("Test Street")
            .plz("12345")
            .city("TestCity")
            .country("TestCountry")
            .build();

    Customer customer =
        Customer.builder()
            .birthdate(LocalDate.of(1990, 1, 1))
            .gender(Gender.MALE)
            .smoker(true)
            .address(address)
            .build();

    LifeInsuranceRequestWithCustomerInfo request =

LifeInsuranceRequestWithCustomerInfo.builder().policyPeriod(20).vs1(100000)
.build();

    LifeInsuranceRequestWithCustomerInfo updatedRequest =
        LifeInsuranceCalculationService.addCustomerInfoToRequest(request,
customer);

    assertEquals(customer.getBirthdate(), updatedRequest.getBirthdate());
    assertEquals(customer.getGender(), updatedRequest.getGender());
    assertEquals(customer.getSmoker(), updatedRequest.getSmoker());
    assertEquals(customer.getAddress().getPlz(), updatedRequest.getPlz());
    assertEquals(customer.getAddress().getCity(),
updatedRequest.getCity());
  }

  @Test
  void checkOfferValidity_Successful() {
    var customerId = 1L;
    var customer =
        Customer.builder()
            .address(Address.builder().build())
            .birthdate(LocalDate.now())
            .gender(Gender.MALE)
            .build();
    var request =
        LifeInsuranceRequestWithCustomerInfo.builder()
            .vs1(10000)
            .pza(PremiumInstallmentsYear.MONTHLY)
            .pensionPillar(PensionPillar.PILLAR_3A)
            .build();
    var expectedResponse = new ProductCalculationResponse();
    var calc = new ProductCalculation();
    calc.setStatus(Status._0);
    expectedResponse.addResultItem(calc);


when(customerRepository.findById(customerId)).thenReturn(Optional.of(custom
```

```
er));
    when(foSoftClient.calculateLifeInsurance(eq(Language.DE),
eq("01.02.2024"), any()))
            .thenReturn(new ResponseEntity<>(expectedResponse, HttpStatus.OK));

    var actualResponses =
lifeInsuranceCalculationService.checkOfferValidity(customerId, request);

    assertNotNull(actualResponses);
    assertFalse(actualResponses.isEmpty());
    var errors =
        actualResponses.stream()
            .map(ProductCalculationResponse::getResult)
            .flatMap(List::stream)
            .map(ProductCalculation::getError)
            .filter(Objects::nonNull)
            .flatMap(List::stream)
            .toList();
    assertTrue(errors.isEmpty());

    verify(customerRepository, times(1)).findById(customerId);
    verify(foSoftClient, atLeastOnce())
        .calculateLifeInsurance(
            eq(Language.DE), eq("01.02.2024"),
any(ProductCalculationRequest.class));
  }

  @Test
  void checkOfferValidity_CustomerNotFound() {
    Long customerId = 1L;
    LifeInsuranceRequestWithCustomerInfo request = new
LifeInsuranceRequestWithCustomerInfo();


when(customerRepository.findById(customerId)).thenReturn(Optional.empty());

    Exception exception =
        assertThrows(
            EntityNotFoundException.class,
            () -> {

lifeInsuranceCalculationService.checkOfferValidity(customerId, request);
            });
    assertTrue(exception.getMessage().contains("Customer not found with ID:
" + customerId));
    verify(customerRepository, times(1)).findById(customerId);
    verifyNoInteractions(foSoftClient);
  }
}
```

Frontend

create-offer.ts

```
'use server'
import {
    CreateLifeInsuranceRequestRequest,
    LifeInsuranceRequestControllerApi,
    ResponseError
} from '@it-apprentices/ovweb'
import {withApi} from '@/lib/with-api'
import {withSpan} from '@/lib/with-span'
import {getLogger} from '@/logging/log-util'
import {nonNullish} from '@/types/guards'

export const createOffer = async (
    request: CreateLifeInsuranceRequestRequest
) => {
    return await withSpan('createOffer', {}, async span => {
        const logger = getLogger('ovweb-frontend')
        const logContext: Record<string, unknown> = {}
        logger.info('creating Offer')
        try {
            return await withApi(
                LifeInsuranceRequestControllerApi
            ).createLifeInsuranceRequest(request)
        } catch (error) {
            const response = (error as ResponseError | undefined)?.response
            const responseBody = await response
                ?.text()
                .catch(() => 'failed to read response body')

            if (nonNullish(response)) {
                const responseStatus = response.status
                const responseStatusText = response.statusText
                span.setAttribute('responseStatus', responseStatus)
                span.setAttribute('responseStatusText', responseStatusText)
                span.setAttribute(
                    'responseBody',
                    responseBody ?? 'no response body'
                )
                logContext.responseStatus = responseStatus
                logContext.responseStatusText = responseStatusText
                logContext.responseBody = responseBody
            }
            logger.error(logContext, 'creating task failed')
            throw new Error(responseBody)
        }
    })
}
```

get-offers.ts

```
'use server'

import {
    LifeInsuranceRequestControllerApi,
    ResponseError
} from '@it-apprentices/ovweb'
import {withApi} from '@/lib/with-api'
import {withSpan} from '@/lib/with-span'
import {getLogger} from '@/logging/log-util'
import {nonNullish} from '@/types/guards'

export const getOffers = async () => {
    return await withSpan('getOffers', {}, async span => {
        const logger = getLogger('ovweb-frontend')
        const logContext: Record<string, unknown> = {}
        logger.info('getting Offers')
        try {
            const offers = await withApi(
                LifeInsuranceRequestControllerApi
            ).getAllLifeInsuranceRequests()
            logger.info('got Offers')
            return offers
        } catch (error) {
            const response = (error as ResponseError).response
            if (nonNullish(response)) {
                const responseStatus = response.status
                const responseStatusText = response.statusText
                const responseBody = await response
                    .text()
                    .catch(() => 'failed to read response body')
                span.setAttribute('responseStatus', responseStatus)
                span.setAttribute('responseStatusText', responseStatusText)
                span.setAttribute('responseBody', responseBody)
                logContext.responseStatus = responseStatus
                logContext.responseStatusText = responseStatusText
                logContext.responseBody = responseBody
            }
            logger.error(logContext, 'deleting task failed')
            throw error
        }
    })
}
```

offers/[id]/page.stories.data.ts

```
import {LifeInsuranceOfferDto} from '@it-apprentices/ovweb'

export const getOfferDetailsSuccess = [
    {
        status: 'SUCCESS',
        offerDetail: {
            overview: ['someOverview'],

            berechnungsliste: [
                {
                    praemie: 17.3,
                    policyPeriod: 26,
                    vs: 100000,
                    vsr: 0,
                    pza: 'MONTHLY',
                    pensionPillar: 'PILLAR_3A',
                    vsArt: 'konstant'
                }
            ],
            praemienzahlartliste: [
                {key: '1', value: '195.9', description: 'jährlich'},
                {key: '2', value: '99.9', description: 'halbjährlich'},
                {key: '4', value: '50.4', description: 'vierteljährlich'}
            ],
            vorsorgeartliste: [
                {
                    key: '3b',
                    value: 'Freie Vorsorge, Säule 3b',
                    description:
                        'Sie wollen frei entscheiden, welche Personen Sie
mit den Leistungen von PREVISTA begünstigen möchten. Für Sie ist wichtig,
dass Sie die Dauer Ihres Versicherungsvertrages selber bestimmen können.'
                },
                {
                    key: '3a',
                    value: 'Gebundene Vorsorge, Säule 3a',
                    description:
                        "Sie sind erwerbstätig und möchten mit den
einbezahlten Prämien Steuern sparen. Dafür sind Sie bereit, die
Versicherung bis zum offiziellen Pensionsalter abzuschliessen und nur
Familienangehörige in der gesetzlichen Erbreihenfolge als Begünstigte
einzusetzen. Die maximal zulässige Prämie für Angestellte beträgt CHF
7'056.-, für Selbstständige CHF 35'280.- pro Jahr."
                }
            ],
            avbliste: [
                {
                    name: 'Allgemeine Versicherungsbedingungen (AVB)
d2_d6',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/avb
_d2_d6_22_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Gebundene Vorsorge (Säule 3a)',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
```

```
_gebvor_21_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Prämienbefreiung',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_i_22_de.pdf'
                }
            ]
        },
        error: []
    }
] as LifeInsuranceOfferDto[]

export const getOfferDetailsSuccessWith2VS = [
    {
        status: 'SUCCESS',
        offerDetail: {
            overview: ['someOverview'],

            berechnungsliste: [
                {
                    praemie: 17.3,
                    policyPeriod: 26,
                    vs: 100000,
                    vsr: 0,
                    pza: 'MONTHLY',
                    pensionPillar: 'PILLAR_3A',
                    vsArt: 'konstant'
                }
            ],
            praemienzahlartliste: [
                {key: '1', value: '195.9', description: 'jährlich'},
                {key: '2', value: '99.9', description: 'halbjährlich'},
                {key: '4', value: '50.4', description: 'vierteljährlich'}
            ],
            vorsorgeartliste: [
                {
                    key: '3b',
                    value: 'Freie Vorsorge, Säule 3b',
                    description:
                        'Sie wollen frei entscheiden, welche Personen Sie
mit den Leistungen von PREVISTA begünstigen möchten. Für Sie ist wichtig,
dass Sie die Dauer Ihres Versicherungsvertrages selber bestimmen können.'
                },
                {
                    key: '3a',
                    value: 'Gebundene Vorsorge, Säule 3a',
                    description:
                        "Sie sind erwerbstätig und möchten mit den
einbezahlten Prämien Steuern sparen. Dafür sind Sie bereit, die
Versicherung bis zum offiziellen Pensionsalter abzuschliessen und nur
Familienangehörige in der gesetzlichen Erbreihenfolge als Begünstigte
einzusetzen. Die maximal zulässige Prämie für Angestellte beträgt CHF
7'056.-, für Selbstständige CHF 35'280.- pro Jahr."
                }
            ],
            avbliste: [
                {
                    name: 'Allgemeine Versicherungsbedingungen (AVB)
```

```
d2_d6',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/avb
_d2_d6_22_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Gebundene Vorsorge (Säule 3a)',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_gebvor_21_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Prämienbefreiung',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_i_22_de.pdf'
                }
            ]
        },
        error: []
    },
    {
        status: 'SUCCESS',
        offerDetail: {
            overview: ['someOverview'],

            berechnungsliste: [
                {
                    praemie: 17.3,
                    policyPeriod: 26,
                    vs: 100000,
                    vsr: 0,
                    pza: 'MONTHLY',
                    pensionPillar: 'PILLAR_3A',
                    vsArt: 'konstant'
                }
            ],
            praemienzahlartliste: [
                {key: '1', value: '195.9', description: 'jährlich'},
                {key: '2', value: '99.9', description: 'halbjährlich'},
                {key: '4', value: '50.4', description: 'vierteljährlich'}
            ],
            vorsorgeartliste: [
                {
                    key: '3b',
                    value: 'Freie Vorsorge, Säule 3b',
                    description:
                        'Sie wollen frei entscheiden, welche Personen Sie
mit den Leistungen von PREVISTA begünstigen möchten. Für Sie ist wichtig,
dass Sie die Dauer Ihres Versicherungsvertrages selber bestimmen können.'
                },
                {
                    key: '3a',
                    value: 'Gebundene Vorsorge, Säule 3a',
                    description:
                        "Sie sind erwerbstätig und möchten mit den
einbezahlten Prämien Steuern sparen. Dafür sind Sie bereit, die
Versicherung bis zum offiziellen Pensionsalter abzuschliessen und nur
Familienangehörige in der gesetzlichen Erbreihenfolge als Begünstigte
```

```
einzusetzen. Die maximal zulässige Prämie für Angestellte beträgt CHF
7'056.-, für Selbstständige CHF 35'280.- pro Jahr."
                }
            ],
            avbliste: [
                {
                    name: 'Allgemeine Versicherungsbedingungen (AVB)
d2_d6',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/avb
_d2_d6_22_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Gebundene Vorsorge (Säule 3a)',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_gebvor_21_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Prämienbefreiung',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_i_22_de.pdf'
                }
            ]
        },
        error: []
    }
] as LifeInsuranceOfferDto[]

export const getOfferDetailsSuccessWith3VS = [
    {
        status: 'SUCCESS',
        offerDetail: {
            overview: ['someOverview'],

            berechnungsliste: [
                {
                    praemie: 17.3,
                    policyPeriod: 26,
                    vs: 100000,
                    vsr: 0,
                    pza: 'MONTHLY',
                    pensionPillar: 'PILLAR_3A',
                    vsArt: 'konstant'
                }
            ],
            praemienzahlartliste: [
                {key: '1', value: '195.9', description: 'jährlich'},
                {key: '2', value: '99.9', description: 'halbjährlich'},
                {key: '4', value: '50.4', description: 'vierteljährlich'}
            ],
            vorsorgeartliste: [
                {
                    key: '3b',
                    value: 'Freie Vorsorge, Säule 3b',
                    description:
                        'Sie wollen frei entscheiden, welche Personen Sie
mit den Leistungen von PREVISTA begünstigen möchten. Für Sie ist wichtig,
```

```
dass Sie die Dauer Ihres Versicherungsvertrages selber bestimmen können.'
                },
                {
                    key: '3a',
                    value: 'Gebundene Vorsorge, Säule 3a',
                    description:
                        "Sie sind erwerbstätig und möchten mit den
einbezahlten Prämien Steuern sparen. Dafür sind Sie bereit, die
Versicherung bis zum offiziellen Pensionsalter abzuschliessen und nur
Familienangehörige in der gesetzlichen Erbreihenfolge als Begünstigte
einzusetzen. Die maximal zulässige Prämie für Angestellte beträgt CHF
7'056.-, für Selbstständige CHF 35'280.- pro Jahr."
                }
            ],
            avbliste: [
                {
                    name: 'Allgemeine Versicherungsbedingungen (AVB)
d2_d6',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/avb
_d2_d6_22_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Gebundene Vorsorge (Säule 3a)',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_gebvor_21_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Prämienbefreiung',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_i_22_de.pdf'
                }
            ]
        },
        error: []
    },
    {
        status: 'SUCCESS',
        offerDetail: {
            overview: ['someOverview'],

            berechnungsliste: [
                {
                    praemie: 34.6,
                    policyPeriod: 26,
                    vs: 100000,
                    vsr: 0,
                    pza: 'MONTHLY',
                    pensionPillar: 'PILLAR_3A',
                    vsArt: 'konstant'
                }
            ],
            praemienzahlartliste: [
                {key: '1', value: '195.9', description: 'jährlich'},
                {key: '2', value: '99.9', description: 'halbjährlich'},
                {key: '4', value: '50.4', description: 'vierteljährlich'}
            ],
```

```
            vorsorgeartliste: [
                {
                    key: '3b',
                    value: 'Freie Vorsorge, Säule 3b',
                    description:
                        'Sie wollen frei entscheiden, welche Personen Sie
mit den Leistungen von PREVISTA begünstigen möchten. Für Sie ist wichtig,
dass Sie die Dauer Ihres Versicherungsvertrages selber bestimmen können.'
                },
                {
                    key: '3a',
                    value: 'Gebundene Vorsorge, Säule 3a',
                    description:
                        "Sie sind erwerbstätig und möchten mit den
einbezahlten Prämien Steuern sparen. Dafür sind Sie bereit, die
Versicherung bis zum offiziellen Pensionsalter abzuschliessen und nur
Familienangehörige in der gesetzlichen Erbreihenfolge als Begünstigte
einzusetzen. Die maximal zulässige Prämie für Angestellte beträgt CHF
7'056.-, für Selbstständige CHF 35'280.- pro Jahr."
                }
            ],
            avbliste: [
                {
                    name: 'Allgemeine Versicherungsbedingungen (AVB)
d2_d6',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/avb
_d2_d6_22_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Gebundene Vorsorge (Säule 3a)',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_gebvor_21_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Prämienbefreiung',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_i_22_de.pdf'
                }
            ]
        },
        error: []
    },
    {
        status: 'SUCCESS',
        offerDetail: {
            overview: ['someOverview'],

            berechnungsliste: [
                {
                    praemie: 69.2,
                    policyPeriod: 26,
                    vs: 100000,
                    vsr: 0,
                    pza: 'MONTHLY',
                    pensionPillar: 'PILLAR_3A',
                    vsArt: 'konstant'
```

```
                }
            ],
            praemienzahlartliste: [
                {key: '1', value: '195.9', description: 'jährlich'},
                {key: '2', value: '99.9', description: 'halbjährlich'},
                {key: '4', value: '50.4', description: 'vierteljährlich'}
            ],
            vorsorgeartliste: [
                {
                    key: '3b',
                    value: 'Freie Vorsorge, Säule 3b',
                    description:
                        'Sie wollen frei entscheiden, welche Personen Sie
mit den Leistungen von PREVISTA begünstigen möchten. Für Sie ist wichtig,
dass Sie die Dauer Ihres Versicherungsvertrages selber bestimmen können.'
                },
                {
                    key: '3a',
                    value: 'Gebundene Vorsorge, Säule 3a',
                    description:
                        "Sie sind erwerbstätig und möchten mit den
einbezahlten Prämien Steuern sparen. Dafür sind Sie bereit, die
Versicherung bis zum offiziellen Pensionsalter abzuschliessen und nur
Familienangehörige in der gesetzlichen Erbreihenfolge als Begünstigte
einzusetzen. Die maximal zulässige Prämie für Angestellte beträgt CHF
7'056.-, für Selbstständige CHF 35'280.- pro Jahr."
                }
            ],
            avbliste: [
                {
                    name: 'Allgemeine Versicherungsbedingungen (AVB)
d2_d6',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/avb
_d2_d6_22_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Gebundene Vorsorge (Säule 3a)',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_gebvor_21_de.pdf'
                },
                {
                    name: 'Ergänzende Versicherungsbedingungen (EVB)
Prämienbefreiung',
                    url:
'https://www.devl.gch.generali.ch/formular_extern/download010/Dokumente/evb
_i_22_de.pdf'
                }
            ]
        },
        error: []
    }
] as LifeInsuranceOfferDto[]

export const getOfferDetailsError = [
    {
        status: 'ERROR',
        error: [
            'Die Endalterregel für die gebundene Vorsorge 3a ist verletzt,
```

```
bitte wählen Sie eine Dauer von mindestens  26 Jahren'
        ]
    }
]
```

offers/[id]/page.stories.ts

```
import type {Meta, StoryObj} from '@storybook/react'

import OfferDetail from './page'
import {
    getOfferDetailsError,
    getOfferDetailsSuccess,
    getOfferDetailsSuccessWith2VS,
    getOfferDetailsSuccessWith3VS
} from '@/app/offers/[id]/page.stories.data'

const meta = {
    title: 'Pages/OfferDetail',
    component: OfferDetail
} satisfies Meta<typeof OfferDetail>

export default meta
type Story = StoryObj<typeof meta>

export const Loading: Story = {
    args: {
        params: {id: 1}
    },
    parameters: {
        actions: {
            getOfferDetails: Promise.resolve()
        }
    }
}
export const CalculationSuccess: Story = {
    args: {
        params: {id: 1}
    },
    parameters: {
        actions: {
            getOfferDetails: Promise.resolve(getOfferDetailsSuccess)
        }
    }
}
export const CalculationSuccessWith2VS: Story = {
    args: {
        params: {id: 1}
    },
    parameters: {
        actions: {
            getOfferDetails: Promise.resolve(getOfferDetailsSuccessWith2VS)
        }
    }
}
export const CalculationSuccessWith3VS: Story = {
    args: {
        params: {id: 1}
    },
    parameters: {
        actions: {
            getOfferDetails: Promise.resolve(getOfferDetailsSuccessWith3VS)
        }
```

```
    }
}
export const CalculationBusinessError: Story = {
    args: {
        params: {id: 1}
    },
    parameters: {
        actions: {
            getOfferDetails: Promise.resolve(getOfferDetailsError)
        }
    }
}
export const DataFetchError: Story = {
    args: {
        params: {id: 1}
    },
    parameters: {
        actions: {
            getOfferDetails: Promise.reject(
                new Error('Response returned an error code')
            )
        }
    }
}
```

offers/[id]/page.tsx

```
'use client'
import React, {FC, useState} from 'react'

import {isNullish, nonNullish} from '@/types/guards'
import {getOfferDetails} from '@/actions'
import {useAsyncEffect} from 'use-async-effect'
import {LifeInsuranceOfferDto} from '@it-apprentices/ovweb'
import {ProductCalculationDisplay} from '@/components/ui/product-
calculation-props'

interface PageProps {
    params: {id: number}
}

const OfferDetails: FC<PageProps> = ({params}) => {
    const [offerData, setOfferData] = useState<LifeInsuranceOfferDto[] |
null>(
        null
    )

    useAsyncEffect(async () => {
        try {
            const response = await getOfferDetails(params.id)
            console.log(response)
            setOfferData(response)
        } catch (e) {
            setOfferData(null)
        }
    }, [])

    if (isNullish(offerData)) {
        return <p>Loading offer details...</p>
    }

    const offers = offerData
        .map(f => f.offerDetail)
        .filter(nonNullish)
        .filter(f => f.berechnungsliste.length > 0)
        .map(({berechnungsliste: [calculation], ...rest}) => ({
            ...rest,
            calculation
        }))
    if (offers.length !== offerData.length) {
        return <div>Failed to calculate offer</div>
    }

    return (
        <div className="flex justify-center w-full py-6">
            <div className="w-10/12 min-w-3xl max-w-8xl p-8 text-center ">
                <h1 className="text-3xl font-bold mb-10">
                    Todesfallversicherung D2
                </h1>
                <div className="flex justify-around w-full">
                    {offers.map((calculation, index) => {
                        const {praemie, policyPeriod, vs} =
                            calculation.calculation
```

```
                            return (
                                <ProductCalculationDisplay
                                    key={index}
                                    praemie={praemie}

pzaDescription={calculation.calculation.pza}
                                    policyPeriod={policyPeriod}
                                    vs={vs}
                                    praemienzahlartliste={
                                        calculation.praemienzahlartliste
                                    }

vorsorgeartliste={calculation.vorsorgeartliste}
                                />
                            )
                        })}
                    </div>
                </div>
            </div>
        )
}
export default OfferDetails
```

create-insurance-form.stories.tsx

```
import type {Meta, StoryObj} from '@storybook/react'
import {
    InsuranceFormWithContext,
    FormModel,
    FieldResolver
} from '@/components/create-insurance-form'
import {FormProvider, useForm} from 'react-hook-form'
import React, {FC} from 'react'
import {useAsyncEffect} from 'use-async-effect'
import {nonNullable} from 'next/dist/lib/non-nullable'
import {PensionPillar, PremiumInstallmentsYear} from '@it-
apprentices/ovweb'

const InsuranceFormWrapper: FC<{
    initModel?: FormModel
    validate?: boolean
}> = ({initModel, validate = false}) => {
    const form = useForm<FormModel>({
        resolver: FieldResolver,
        mode: 'onBlur',
        defaultValues: {
            customerId: '',
            dauer: '',
            vs1: '',
            pza: undefined,
            vsart: '',
            praemienbefreiung: false
        }
    })

    useAsyncEffect(async () => {
        if (validate) {
            await form.trigger()
        }
    }, [form.trigger])
    useAsyncEffect(async () => {
        if (nonNullable(initModel)) {
            form.reset(initModel)

            form.setValue('vs1', '200000')

            if ('vs2' in initModel) {
                form.setValue('vs2', '200000')
            }

            if ('vs3' in initModel) {
                form.setValue('vs2', '200000')
                form.setValue('vs3', '200000')
            }
        }
    }, [form.reset, form.setValue, initModel])

    return (
        <FormProvider {...form}>
            <InsuranceFormWithContext />
        </FormProvider>
    )
}
```

```
const meta = {
    title: 'Components/CreateInsuranceForm',
    component: InsuranceFormWrapper
} satisfies Meta<typeof InsuranceFormWrapper>

export default meta
type Story = StoryObj<typeof meta>

export const Filled: Story = {
    args: {
        initModel: {
            customerId: '123',
            dauer: '20',
            vs1: '100000',
            pza: PremiumInstallmentsYear.Monthly,
            vsart: 'konstant',
            praemienbefreiung: true,
            dreiAdreiB: PensionPillar._3A
        }
    },
    parameters: {
        nextjs: {
            appDirectory: true
        }
    }
}

export const FilledWith2VS: Story = {
    args: {
        initModel: {
            customerId: '123',
            dauer: '20',
            vs1: '100000',
            vs2: '100000',
            pza: PremiumInstallmentsYear.Monthly,
            vsart: 'konstant',
            praemienbefreiung: true,
            dreiAdreiB: PensionPillar._3A
        }
    },
    parameters: {
        nextjs: {
            appDirectory: true
        }
    }
}

export const FilledWith3VS: Story = {
    args: {
        initModel: {
            customerId: '123',
            dauer: '20',
            vs1: '100000',
            vs2: '100000',
            vs3: '100000',
            pza: PremiumInstallmentsYear.Monthly,
            vsart: 'konstant',
            praemienbefreiung: true,
            dreiAdreiB: PensionPillar._3A
        }
    },
```

```
    parameters: {
        nextjs: {
            appDirectory: true
        }
    }
}

export const Empty: Story = {
    args: {},
    parameters: {
        nextjs: {
            appDirectory: true
        }
    }
}
export const FailedValidation: Story = {
    args: {
        validate: true
    },
    parameters: {
        nextjs: {
            appDirectory: true
        }
    }
}

export const FailedCreation: Story = {
    args: {},
    parameters: {
        actions: {
            createOffer: Promise.reject(
                new Error('Response returned an error code')
            )
        },
        nextjs: {
            appDirectory: true
        }
    }
}
```

create-insurance-form.tsx

```
'use client'

import {zodResolver} from '@hookform/resolvers/zod'
import {FormProvider, useForm, useFormContext} from 'react-hook-form'
import * as z from 'zod'
import {Checkbox} from '@/components/ui/checkbox'
import {useToast} from '@/components/ui/use-toast'
import {useRouter} from 'next/navigation'

import {Button} from '@/components/ui/button'
import {
    Form,
    FormControl,
    FormField,
    FormItem,
    FormLabel,
    FormMessage
} from '@/components/ui/form'

import React, {useEffect, useState} from 'react'

import {Input} from '@/components/ui/input'
import {
    Select,
    SelectContent,
    SelectItem,
    SelectTrigger,
    SelectValue
} from '@/components/ui/select'
import {createOffer, getCustomers} from '@/actions'
import {useAsyncEffect} from 'use-async-effect'
import {
    CustomerGetDto as Customer,
    PensionPillar,
    PremiumInstallmentsYear
} from '@it-apprentices/ovweb'
import {nonNullish} from '@/types/guards'

const insuranceFormSchema = z.object({
    customerId: z
        .string()
        .nonempty({message: 'Ein Kunde muss ausgewählt werden.'}),

    dauer: z.string().refine(
        value => {
            const parsedValue = parseInt(value, 10)
            return !isNaN(parsedValue) && parsedValue >= 5 && parsedValue
<= 45
        },
        {
            message: 'Die Mindestdauer muss 5-45 Jahre betragen'
        }
    ),

    vs1: z.string().refine(
        value => {
            const parsedValue = parseInt(value, 10)
            return (
                !isNaN(parsedValue) &&
```

```
                parsedValue >= 40000 &&
                parsedValue <= 1000000
            )
        },
        {
            message: "VS muss mindestens 40.000 und hoechstens 1'000'000
sein."
        }
    ),

    vs2: z
        .string()
        .optional()
        .refine(
            value => {
                if (value === undefined || value === '') return true

                const parsedValue = parseInt(value, 10)
                return (
                    !isNaN(parsedValue) &&
                    parsedValue >= 40000 &&
                    parsedValue <= 1000000
                )
            },
            {
                message:
                    "VS muss mindestens 40.000 und hoechstens 1'000'000
sein."
            }
        ),

    vs3: z
        .string()
        .optional()
        .refine(
            value => {
                if (value === undefined || value === '') return true

                const parsedValue = parseInt(value, 10)
                return (
                    !isNaN(parsedValue) &&
                    parsedValue >= 40000 &&
                    parsedValue <= 1000000
                )
            },
            {
                message:
                    "VS muss mindestens 40.000 und hoechstens 1'000'000
sein."
            }
        ),

    pza: z.nativeEnum(PremiumInstallmentsYear),
    dreiAdreiB: z.nativeEnum(PensionPillar),

    vsart: z.string(),
    praemienbefreiung: z.boolean()
})

export type FormModel = z.infer<typeof insuranceFormSchema>
export const FieldResolver = zodResolver(insuranceFormSchema)
```

```
export function InsuranceFormWithContext() {
    const [customers, setCustomers] = useState<Customer[]>([])
    const [loading, setLoading] = useState(false)
    const [error, setError] = useState<Error | null>(null)
    // const [submitError, setSubmitError] = useState('')
    const {toast} = useToast()
    const router = useRouter()
    const form = useFormContext<FormModel>()
    const [vs2Enabled, setvs2Enabled] = useState(false)
    const [vs3Enabled, setvs3Enabled] = useState(false)

    useAsyncEffect(async () => {
        try {
            const response = await getCustomers()
            setCustomers(response)
        } catch (e) {
            setError(e as Error)
        }
    }, [])

    useEffect(() => {
        const subscription = form.watch((value, {name}) => {
            if (name === 'vs1') {
                setvs2Enabled(value.vs1 !== undefined && value.vs1 !== '')
            }
            if (name === 'vs2') {
                setvs3Enabled(value.vs2 !== undefined && value.vs2 !== '')
            }
        })
        return () => subscription.unsubscribe()
    }, [form.watch])

    async function onSubmit(values: z.infer<typeof insuranceFormSchema>) {
        try {
            setLoading(true)
            const offer = await createOffer({
                lifeInsuranceRequestPostDto: {
                    ...values,
                    policyPeriod: parseInt(values.dauer, 10),
                    vs1: parseInt(values.vs1, 10),
                    vs2:
                        values.vs2 !== undefined
                            ? parseInt(values.vs2, 10)
                            : undefined,
                    vs3:
                        values.vs3 !== undefined
                            ? parseInt(values.vs3, 10)
                            : undefined,
                    pza: values.pza,
                    pensionPillar: values.dreiAdreiB,
                    vsArt: values.vsart,
                    premiumWaiver: values.praemienbefreiung,
                    customerId: parseInt(values.customerId, 10)
                }
            })
            const offerId = offer.id
            router.push(`/offers/${offerId}`)

            toast({
                title: 'Offerte erfolgreich erstellt',
                variant: 'success'
```

```
            })
        } catch (e) {
            console.error('Error while creating offer request:', e)
            form.setError('root.formValidation', {
                message: JSON.parse((e as Error).message).message
            })
        } finally {
            setLoading(false)
        }
    }

    return (
        <div className="flex justify-center w-full py-6">
            <div className="flex flex-col w-2/5 h-4/5 min-h-[63vh] bg-white
shadow-lg rounded-lg overflow-hidden border-indigo-500/100">
                <div className="flex flex-col m-10">
                    <Form {...form}>
                        <form
                            onSubmit={form.handleSubmit(onSubmit)}
                            className="space-y-8"
                        >
                            <FormField
                                control={form.control}
                                name="customerId"
                                render={({field}) => (
                                    <FormItem>
                                        <FormLabel>Kunde</FormLabel>
                                        <FormControl>
                                            <Select
                                                value={field.value}
                                                name={field.name}

onValueChange={field.onChange}
                                                >
                                                <SelectTrigger
                                                    className={`border-2 ${

form.formState.errors
                                                        .customerId
                                                        ? 'border-red-
500'
                                                        : 'border-gray-
300'
                                                    } rounded-md shadow-
sm`}
                                                >
                                                    <SelectValue
                                                        placeholder="Select
customer"

onBlur={field.onBlur}
                                                        ref={field.ref}
                                                    />
                                                </SelectTrigger>
                                                <SelectContent>
                                                    {loading ? (
                                                        <SelectItem
                                                            value="loading"
                                                            disabled
                                                        >
                                                            Loading
```

```
customers...
                                                    </SelectItem>
                                                ) : error !== null ? (
                                                    <SelectItem
                                                        value="error"
                                                        disabled
                                                    >
                                                        Error loading
                                                        customers
                                                    </SelectItem>
                                                ) : (
                                                    customers.map(
                                                        customer => (
                                                            <SelectItem
                                                                key={

customer.id
                                                                }

value={customer.id.toString()}
                                                                >

{`${customer.firstName} ${customer.lastName}`}

</SelectItem>
                                                        )
                                                    )
                                                )}
                                            </SelectContent>
                                        </Select>
                                    </FormControl>
                                    <FormMessage>
                                        {form.formState.errors
                                            .customerId && (
                                            <p className="text-red-
500">
                                                {

form.formState.errors

.customerId.message
                                                }
                                            </p>
                                        )}
                                    </FormMessage>
                                </FormItem>
                            )}
                        />
                        <div className="flex justify-between space-x-
4">
                            <div className="flex justify-between w-
1/3">
                                <FormField
                                    control={form.control}
                                    name="vs1"
                                    render={({field}) => (
                                        <FormItem>
                                            <FormLabel>VS1</FormLabel>
                                            <Input
                                                {...field}
                                                type="text"
```

```
                                                placeholder="VS1
eingeben"
                                            />
                                            <FormMessage>
                                                {form.formState.errors
                                                    .vs1 && (
                                                    <p className="text-
red-500">
                                                        {
form.formState
.errors.vs1
.message
                                                        }
                                                    </p>
                                                )}
                                            </FormMessage>
                                        </FormItem>
                                    )}
                                />
                            </div>
                            <div className="flex justify-between w-
1/3">
                                <FormField
                                    control={form.control}
                                    name="vs2"
                                    render={({field}) => (
                                        <FormItem>
                                            <FormLabel>VS2</FormLabel>
                                            <Input
                                                {...field}
                                                type="text"
                                                placeholder="VS2
eingeben"
                                                disabled={!vs2Enabled}
                                                className={`${
                                                    !vs2Enabled
                                                        ? 'bg-gray-200'
                                                        : ''
                                                } flex-1`}
                                            />
                                            <FormMessage>
                                                {form.formState.errors
                                                    .vs2 && (
                                                    <p className="text-
red-500">
                                                        {
form.formState
.errors.vs2
.message
                                                        }
                                                    </p>
                                                )}
                                            </FormMessage>
                                        </FormItem>
                                    )}
```

```
                                    />
                                </div>
                                <div className="flex justify-between w-
1/3">
                                    <FormField
                                        control={form.control}
                                        name="vs3"
                                        render={({field}) => (
                                            <FormItem>
                                                <FormLabel>VS3</FormLabel>
                                                <Input
                                                    {...field}
                                                    type="text"
                                                    placeholder="VS3
eingeben"
                                                    disabled={!vs3Enabled}
                                                    className={`${
                                                        !vs3Enabled
                                                            ? 'bg-gray-200'
                                                            : ''
                                                    } flex-1`}
                                                />
                                                <FormMessage>
                                                    {form.formState.errors
                                                        .vs3 && (
                                                        <p className="text-
red-500">
                                                            {

form.formState

.errors.vs3

.message
                                                            }
                                                        </p>
                                                    )}
                                                </FormMessage>
                                            </FormItem>
                                        )}
                                    />
                                </div>
                            </div>

                            <FormField
                                control={form.control}
                                name="dauer"
                                render={({field}) => (
                                    <FormItem>
                                        <FormLabel>Dauer</FormLabel>
                                        <Input
                                            {...field}
                                            type="text"
                                            placeholder="Dauer der
Versicherung in Jahren eingeben"
                                        />
                                        <FormMessage>
                                            {form.formState.errors.dauer &&
(
                                                <p className="text-red-
500">
```

```
                                                {
form.formState.errors
                                                                .dauer.message
                                                }
                                            </p>
                                        )}
                                    </FormMessage>
                                </FormItem>
                            )}
                        />
                        <FormField
                            control={form.control}
                            name="pza"
                            render={({field}) => (
                                <FormItem>
                                    <FormLabel>PZA</FormLabel>

                                    <Select
                                        value={field.value}
                                        name={field.name}
                                        onValueChange={field.onChange}
                                    >
                                        <SelectTrigger>
                                            <SelectValue
                                                onBlur={field.onBlur}
                                                ref={field.ref}
                                            />
                                        </SelectTrigger>
                                        <SelectContent>
                                            {Object.entries(
                                                PremiumInstallmentsYear
                                            ).map(([k, v]) => (
                                                <SelectItem
                                                    key={k}
                                                    value={v}
                                                >
                                                    {k}
                                                </SelectItem>
                                            ))}
                                        </SelectContent>
                                    </Select>

                                    <FormMessage>
                                        {form.formState.errors.pza && (
                                            <p className="text-red-
500">
                                                {
form.formState.errors
                                                                .pza.message
                                                }
                                            </p>
                                        )}
                                    </FormMessage>
                                </FormItem>
                            )}
                        />

                        <FormField
                            control={form.control}
```

```
                                name="vsart"
                                render={({field}) => (
                                    <FormItem>
                                        <FormLabel>VSArt</FormLabel>
                                        <Input
                                            {...field}
                                            type="text"
                                            placeholder="Versicherungsart
eingeben"
                                        />
                                        <FormMessage>
                                            {form.formState.errors.vsart &&
(
                                                <p className="text-red-
500">
                                                    {
form.formState.errors
                                                        .vsart.message
                                                    }
                                                </p>
                                            )}
                                        </FormMessage>
                                    </FormItem>
                                )}
                            />

                            <FormField
                                control={form.control}
                                name="praemienbefreiung"
                                render={({field}) => (
                                    <FormItem className="flex flex-row
items-start space-x-3 space-y-0 rounded-md border p-4">
                                        <FormControl>
                                            <Checkbox
                                                checked={field.value}
onCheckedChange={field.onChange}
                                            />
                                        </FormControl>
                                        <div className="space-y-1 leading-
none">
                                            <FormLabel>
                                                Praemienbefreiung
                                            </FormLabel>
                                        </div>
                                    </FormItem>
                                )}
                            />

                            <FormField
                                control={form.control}
                                name="dreiAdreiB"
                                render={({field}) => (
                                    <FormItem>
                                        <FormLabel>3a 3b</FormLabel>
                                        <Select
                                            value={field.value}
                                            name={field.name}
                                            onValueChange={field.onChange}
                                        >
```

```
                                                <SelectTrigger>
                                                    <SelectValue
                                                        placeholder="Wählen Sie
3a oder 3b"
                                                        onBlur={field.onBlur}
                                                        ref={field.ref}
                                                    />
                                                </SelectTrigger>
                                                <SelectContent>
                                                    {Object.entries(
                                                        PensionPillar
                                                    ).map(([k, v]) => (
                                                        <SelectItem
                                                            key={k}
                                                            value={v}
                                                        >
                                                            {v}
                                                        </SelectItem>
                                                    ))}
                                                </SelectContent>
                                            </Select>
                                            <FormMessage>
                                                {form.formState.errors
                                                    .dreiAdreiB && (
                                                    <p className="text-red-
500">
                                                        {

form.formState.errors

.dreiAdreiB.message
                                                        }
                                                    </p>
                                                )}
                                            </FormMessage>
                                        </FormItem>
                                    )}
                                />

                                {nonNullish(form.formState.errors.root) && (
                                    <div className="text-red-500 mb-4">
                                        {
                                            form.formState.errors.root
                                                .formValidation.message
                                        }
                                    </div>
                                )}

                                <Button type="submit">Speichern</Button>
                            </form>
                        </Form>
                    </div>
                </div>
            </div>
        )
}

export default function InsuranceForm() {
    const form = useForm<FormModel>({
        resolver: FieldResolver,
        mode: 'onBlur',
```

```
        defaultValues: {
            customerId: '',
            dauer: '',
            vs1: '',
            vs2: '',
            vs3: '',
            pza: undefined,
            vsart: '',
            praemienbefreiung: false
        }
    })

    return (
        <FormProvider {...form}>
            <InsuranceFormWithContext />
        </FormProvider>
    )
}
```

product-calculation-props.spec.tsx

```
import {render, screen, waitFor} from '@testing-library/react'
import '@testing-library/jest-dom'
import {ProductCalculationDisplay} from './product-calculation-props'
import {PremiumInstallmentsYear} from '@it-apprentices/ovweb'

jest.mock('@/actions', () => ({
    getOfferDetails: () => Promise.resolve()
}))
jest.mock('next/navigation', () => ({
    useRouter: jest.fn(() => ({
        push: jest.fn()
    }))
}))

describe('ProductCalculationDisplay', () => {
    it('renders the component with provided props', async () => {
        const args = {
            praemie: '500',
            policyPeriod: 20,
            vs: 100000,
            pzaDescription: PremiumInstallmentsYear.Monthly,
            praemienzahlartliste: [
                {key: 'monthly', value: '50', description: 'Monatlich'},
                {key: 'yearly', value: '600', description: 'Jährlich'}
            ],
            vorsorgeartliste: [
                {key: 'd2', value: 'Todesfallversicherung D2'},
                {key: 'd3', value: 'Todesfallversicherung D3'}
            ]
        }

        render(<ProductCalculationDisplay {...args} />)
        await waitFor(() => {
            expect(screen.getByText('500 Fr.')).toBeInTheDocument()
            expect(screen.getByText('MONTHLY')).toBeInTheDocument()
        })
    })
})
```

product-calculation-props.stories.tsx

```
import {Meta, StoryObj} from '@storybook/react'
import {ProductCalculationDisplay} from './product-calculation-props'
import {PremiumInstallmentsYear} from '@it-apprentices/ovweb'

const meta = {
    title: 'Components/ProductCalculationDisplayProp',
    component: ProductCalculationDisplay
} satisfies Meta<typeof ProductCalculationDisplay>
export default meta
type Story = StoryObj<typeof meta>
export const Filled: Story = {
    args: {
        praemie: '500',
        policyPeriod: 20,
        vs: 100000,
        pzaDescription: PremiumInstallmentsYear.Monthly,
        praemienzahlartliste: [
            {key: 'monthly', value: '50', description: 'Monatlich'},
            {key: 'yearly', value: '600', description: 'Jährlich'}
        ],
        vorsorgeartliste: [
            {key: 'd2', value: 'Todesfallversicherung D2'},
            {key: 'd3', value: 'Todesfallversicherung D3'}
        ]
    }
}
```

product-calculation-props.tsx

```
import {PremiumInstallmentsYear} from '@it-apprentices/ovweb'
export interface ProductCalculationDisplayProps {
    praemie: number | string
    policyPeriod: number
    vs: number
    pzaDescription: PremiumInstallmentsYear
    praemienzahlartliste: {key: string; value: string; description:
string}[]
    vorsorgeartliste: {key: string; value: string}[]
}

const ProductCalculationDisplay: React.FC<ProductCalculationDisplayProps> =
({
    praemie,
    pzaDescription,
    policyPeriod,
    vs,
    praemienzahlartliste,
    vorsorgeartliste
}) => {
    return (
        <div className="border border-gray-100 shadow-lg rounded-2xl
overflow-hidden mx-4 flex flex-col min-h-[550px] max-w-[437px] w-full">
            <div className="bg-white p-4 flex justify-center min-h-[160px]
items-center flex-col">
                <span className="text-3xl font-bold text-red-
600">{`${praemie} Fr.`}</span>
                <span className="text-base font-bold text-gray-500">
                    {pzaDescription}
                </span>
            </div>
            <div className="bg-gray-200 p-4 rounded-lg flex-grow">
                <div className="mb-4">
                    <div className="flex justify-between items-center my-
5">
                        <span className="font-
bold">Versicherungssumme:</span>
                        <span className="font-bold">{vs} Fr.</span>
                    </div>
                    <hr className="border-t border-gray-300 mx-auto w-
11/12" />
                </div>

                <div className="mb-4">
                    <div className="flex justify-between items-center my-
5">
                        <span className="font-bold">Dauer:</span>
                        <span className="font-bold">{policyPeriod}
Jahre</span>
                    </div>
                    <hr className="border-t border-gray-300 mx-auto w-
11/12" />
                </div>

                <div className="mb-4">
                    <div className="flex">
                        <span className="font-bold self-center mr-2">
```

```
                            Zahlart:
                        </span>
                        <div className="flex-grow flex">
                            <div className="w-3/4">
                                {praemienzahlartliste.map((item, index) =>
(
                                    <div
                                        key={index}
                                        className="mb-2 ml-20 text-left"
                                    >
                                        <span className="text-gray-500
font-bold text-left content-start">
                                            {item.description}:
                                        </span>
                                    </div>
                                ))}
                            </div>
                            <div className="w-1/4 text-right">
                                {praemienzahlartliste.map((item, index) =>
(
                                    <div key={index} className="mb-2">
                                        <span className="font-bold">
                                            {item.value} Fr.
                                        </span>
                                    </div>
                                ))}
                            </div>
                        </div>
                    </div>
                    <hr className="border-t border-gray-300 mx-auto w-11/12
mt-4" />
                </div>

                <div className="mb-4">
                    <div className="py-5">
                        <span className="font-bold text-left block mb-2">
                            Vorsorgeliste:
                        </span>
                        <ul className="list-disc pl-5 mt-2 text-left">
                            {vorsorgeartliste.map((item, index) => (
                                <li
                                    key={index}
                                    className="ml-4 text-gray-500 font-bold
mb-2"
                                >
                                    {item.value}
                                </li>
                            ))}
                        </ul>
                    </div>
                </div>
            </div>
        </div>
    )
}

export {ProductCalculationDisplay}
```