Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №9

Теорія Розробки Програмного Забезпечення

Шаблон «SOA»

Виконав

студент групи ІА-14:

Фіалківський І.О.

Перевірив:

Мягкий М.Ю.

Київ 2023

Мета: метою даної лабораторної роботи є вивчення та реалізація принципів архітектури Службової Орієнтованої Архітектури (SOA) у розробці програмних систем. Лабораторна спрямована на освоєння концепцій сервісів, їх взаємодії та інтеграції в рамках SOA, а також на практичний досвід розробки сервісно-орієнтованих застосунків.

Виконання

Завданням на лобораторну було реалізувати шаблон ***SOA***.

Службово-орієнтована архітектура (SOA) є стратегією розробки програмного забезпечення, яка покладається на створення і використання незалежних, взаємодіючих між собою служб. У порівнянні з традиційними монолітними системами, SOA пропонує гнучкішу та масштабовану архітектуру, що дозволяє покращити розподілені та розширювані характеристики програмних систем.

Однією з ключових переваг SOA є здатність до створення розподілених застосунків з використанням незалежних компонентів, які можуть бути легко взаємодіючими та перевикористовуваними. Це сприяє полегшенню розвитку, утримання та розширенню системи, а також забезпечує більшу гнучкість у відповіді на зміни в бізнес-вимогах. В даній лабораторній роботі ми дослідимо ці переваги та навчимося розробляти програмні системи відповідно до принципів SOA.

Я вирішив винести в мікросервіс все що взаємодіє з файловою системою. На цьому етапі це один сервіс який генерує статистичний вайл.

Контроллер:

```
package com.example.dbmicroservice.controller;

@RestController
public class StatisticFileController {

    private final StatisticsService statisticsService;

    private final AccumulationDtoConverter accumulationDtoConverter;
    private final FinancialArrangementDtoConverter
financialArrangementDtoConverter;
    private final TransactionDTOConverter transactionDTOConverter;

    @Autowired
    public StatisticFileController(StatisticsService statisticsService,
AccumulationDtoConverter accumulationDtoConverter,
FinancialArrangementDtoConverter financialArrangementDtoConverter,
TransactionDTOConverter transactionDTOConverter) {
        this.statisticsService = statisticsService;
        this.accumulationDtoConverter = accumulationDtoConverter;
        this.financialArrangementDtoConverter =
financialArrangementDtoConverter;
        this.transactionDTOConverter = transactionDTOConverter;
    }
```

```
    @GetMapping
    public ResponseEntity<Resource> statisticsFile(@RequestBody
StatisticFileRequest statisticFileRequest) {

        List<Transaction> transactions =
statisticFileRequest.getTransactionDTOList().stream()
                .map(transactionDTOConverter::convertToTransaction).toList();
        List<Accumulation> accumulations =
statisticFileRequest.getAccumulationDTOList().stream()

.map(accumulationDtoConverter::convertToAccumulation).toList();
        List<FinancialArrangement> faList =
statisticFileRequest.getFinancialArrangementDTOList().stream()
                .map(financialArrangementDtoConverter::convertToFA).toList();


        File file = statisticsService.getStatisticsInFile(transactions,
accumulations, faList, statisticFileRequest.getEmail());

        try {
            InputStreamResource resource = new InputStreamResource(new
FileInputStream(file));

            HttpHeaders headers = new HttpHeaders();
            headers.add(HttpHeaders.CONTENT_DISPOSITION, "attachment;
filename=Statistics."
                    + statisticFileRequest.getFormat());

            return ResponseEntity.ok()
                    .headers(headers)
                    .contentLength(file.length())
                    .contentType(MediaType.APPLICATION_OCTET_STREAM)
                    .body(resource);

        } catch (FileNotFoundException e) {
            throw new RuntimeException("Something went wrong. Try later.");
        }
    }


    @ExceptionHandler
    public ResponseEntity<ErrorResponse> handleException(RuntimeException e)
{
        ErrorResponse errorResponse = new ErrorResponse(e.getMessage(),
System.currentTimeMillis());
        return new ResponseEntity<>(errorResponse, HttpStatus.BAD_REQUEST);
    }

}
```

DTO:

```
package com.example.dbmicroservice.dto;

@Getter
@Setter
@NoArgsConstructor
public class StatisticFileRequest {

    @NotNull
    private List<TransactionDTO> transactionDTOList;
    @NotNull
    private List<FinancialArrangementDTO> financialArrangementDTOList;
    @NotNull
    private List<AccumulationDTO> accumulationDTOList;
```

```
    @Email
    private String email;

    @NotBlank
    private String format;
}
```

```
package com.example.dbmicroservice.dto;

@NoArgsConstructor
@Getter
@Setter
public class TransactionDTO {

    private int id;

    @Min(value = 0, message = "Transaction sum can't be negative")
    private int sum;

    @Length(max = 150, message = "Comment should be less then 150
characters")
    private String comment;

    @NotNull(message = "Category can't be empty")
    private TransactionCategory category;

    @NotNull
    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", shape =
JsonFormat.Shape.STRING)
    private LocalDateTime dateTime;

    @Column(name = "refill")
    private boolean refill;

    @Column(name = "periodic")
    private boolean periodic;
}
```

```
package com.example.dbmicroservice.dto;

@NoArgsConstructor
@Setter
@Getter
public class AccumulationDTO {

    private int id;

    @NotBlank
    @Length(min = 1, max = 50, message = "Name should contains less than 50
characters")
    private String name;

    @Length(max = 150, message = "Comment should be less then 150
characters")
    private String comment;

    @Min(value = 0, message = "Current sum should be greater then 0")
    private int currentSum;

    @Min(value = 0, message = "Goal sum should be greater then 0")
    private int goalSum;
```

```
    @JsonFormat(pattern = "yyyy-MM-dd", shape = JsonFormat.Shape.STRING)
    private LocalDate startDate;

    @Future
    @NotNull
    @JsonFormat(pattern = "yyyy-MM-dd", shape = JsonFormat.Shape.STRING)
    private LocalDate endDate;

    @JsonFormat(pattern = "yyyy-MM-dd", shape = JsonFormat.Shape.STRING)
    private LocalDate lastPaymentDate;

    private Status status;
}
```

```
package com.example.dbmicroservice.dto;

@NoArgsConstructor
@Getter
@Setter
public class FinancialArrangementDTO {

    private int id;

    @NotBlank
    @Length(min = 1, max = 50, message = "Name should contains less than 50
characters")
    private String name;

    @Min(value = 0, message = "Rate should be greater then 0")
    private int percent;

    @Min(value = 0, message = "Current sum should be greater then 0")
    private int startSum;

    @Min(value = 0, message = "Current sum should be greater then 0")
    private int currentSum;

    private int refundSum;

    @JsonFormat(pattern = "yyyy-MM-dd", shape = JsonFormat.Shape.STRING)
    private LocalDate startDate;

    @Future
    @NotNull
    @JsonFormat(pattern = "yyyy-MM-dd", shape = JsonFormat.Shape.STRING)
    private LocalDate endDate;

    private boolean fromToUserFunds;

    @NotNull
    private FinancialArrangementState state;

    private Status status;
}
```

Важливим моментом є атентифікація користувача. Я використовував JWT.

JWT filter:

```
@Component
public class JWTFilter extends OncePerRequestFilter {

    private final JWTUtil jwtUtil;
    private final PersonDetailsService personDetailsService;

    @Autowired
    public JWTFilter(JWTUtil jwtUtil, PersonDetailsService
personDetailsService) {
        this.jwtUtil = jwtUtil;
        this.personDetailsService = personDetailsService;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response, FilterChain filterChain) throws
ServletException, IOException {
        String authHeader = request.getHeader("Authorization");

        if(authHeader != null && !authHeader.isBlank() &&
authHeader.startsWith("Bearer ")) {
            String jwt = authHeader.substring(7);

            if(jwt.isBlank()) {
                response.sendError(HttpServletResponse.SC_BAD_REQUEST,
"Invalid JWT Token in Bearer Header");
            }else {
                try {
                    String username =
jwtUtil.validateTokenAndRetrieveClaim(jwt);
                    UserDetails userDetails =
personDetailsService.loadUserByUsername(username);

                    UsernamePasswordAuthenticationToken authToken =
                            new
UsernamePasswordAuthenticationToken(userDetails,
                                    userDetails.getPassword(),
userDetails.getAuthorities());

                    if
(SecurityContextHolder.getContext().getAuthentication() == null) {

SecurityContextHolder.getContext().setAuthentication(authToken);
                    }
                }catch (JWTVerificationException exc) {
                    response.sendError(HttpServletResponse.SC_BAD_REQUEST,
"Invalid JWT Token");
                }
            }
        }
        filterChain.doFilter(request, response);
        }
}
```

JWT util:

```
@Component
public class JWTUtil {

    @Value("${jwt_secret}")
    private String secret;

    public String generateToken(String username) {
        Date expirationDate =
Date.from(ZonedDateTime.now().plusMinutes(60).toInstant());

        return JWT.create()
                .withSubject("User details")
                .withClaim("username", username)
                .withIssuedAt(new Date())
                .withIssuer("server")
                .withExpiresAt(expirationDate)
                .sign(Algorithm.HMAC256(secret));
    }

    public String validateTokenAndRetrieveClaim(String token) throws
JWTVerificationException {
        JWTVerifier verifier = JWT.require(Algorithm.HMAC256(secret))
                .withSubject("User details")
                .withIssuer("server")
                .build();

        DecodedJWT jwt = verifier.verify(token);
        return jwt.getClaim("username").asString();
    }
}
```

Сервіс:

```
@Service
@PropertySource("classpath:application.properties")
public class StatisticsService {


    //TODO: maybe set default value for handler
    private StatisticsFileHandler fileHandler;


    @Transactional(readOnly = true)
    public File getStatisticsInFile(List<Transaction> transactions,
List<Accumulation> accumulations,
                                    List<FinancialArrangement> arrangements,
String email) {
        if(fileHandler == null)
            throw new NullPointerException("Can't generate statistics file
without fileGenerator");

        if(transactions.isEmpty() || accumulations.isEmpty() ||
arrangements.isEmpty())
            throw new IllegalArgumentException("No data for statistics");

        return fileHandler.generateStatisticsFile(transactions,
accumulations, arrangements, email);
    }

    @Scheduled(cron = "${statistics.file.cron.daily}")
```

```
    public void deleteDailyStatisticsFile() throws IOException {
        StatisticsFileHandler.deleteAllFile();
    }

    public StatisticsFileHandler getFileHandler() {
        return fileHandler;
    }

    public void setFileHandler(StatisticsFileHandler fileGenerator) {
        this.fileHandler = fileGenerator;
    }
}
```

Інтерфейс файл хендлеру:

```
package com.example.dbmicroservice.service;

import com.example.dbmicroservice.entity.Accumulation;
import com.example.dbmicroservice.entity.FinancialArrangement;
import com.example.dbmicroservice.entity.Transaction;
import org.apache.commons.io.FileUtils;

import java.io.File;
import java.io.IOException;
import java.util.List;

public abstract class StatisticsFileHandler {

    private static final String FOLDER = "statistics_file/";

    protected static final String FILE_PATH = "statistics_file/%s.%s";

    protected static final List<String> TRANSACTION_TABLE_COLUMNS_NAME =
            List.of("№", "Sum", "Refill", "Comment", "Category", "Date and
time");

    protected static final List<String> ACCUMULATIONS_TABLE_COLUMNS_NAME =
            List.of("№", "Name", "Comment", "Current sum", "Goal sum", "Start
date", "End date",
                    "Last payment date", "Status");

    protected static final List<String>
FINANCIAL_ARRANGEMENT_TABLE_COLUMNS_NAME =
            List.of("№", "Name", "Type", "Percent", "StartSum", "Current
Sum",
                    "Refund Sum", "Start date", "End date", "From to user
funds", "Status");


    public abstract File generateStatisticsFile(List<Transaction>
transactions, List<Accumulation> accumulations,
                                                List<FinancialArrangement>
financialArrangements, String userEmail);

    public static void deleteAllFile() {
        try {
            FileUtils.cleanDirectory(new File(FOLDER));
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage(), e);
        }
    }
}
```

Одна з реалізацій:

```
package com.example.dbmicroservice.service;


public class WordStatisticsFileHandler extends StatisticsFileHandler {

    //TODO: make another date time format

    @Override
    public File generateStatisticsFile(List<Transaction> transactions,
List<Accumulation> accumulations,
                                       List<FinancialArrangement>
financialArrangements, String userEmail){
        File statisticsFile = new File(String.format(FILE_PATH, userEmail,
"docx"));

        try(FileOutputStream output = new FileOutputStream(statisticsFile)) {
            XWPFDocument document = new XWPFDocument();

            createDocumentStructure(document, transactions, accumulations,
financialArrangements);

            List<XWPFTable> tables = document.getTables();

            fillTransactionTable(tables.get(0), transactions);
            fillAccumulationsTable(tables.get(1), accumulations);
            fillFinancialArrangementTable(tables.get(2),
financialArrangements);

            document.write(output);
            document.close();
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage(), e);
        }

        return statisticsFile;
    }

    protected void createDocumentStructure(XWPFDocument document,
List<Transaction> transactions,
                                           List<Accumulation> accumulations,
List<FinancialArrangement> financialArrangements){
        document.createTable(transactions.size() + 1, 6);
        addNewLine(document);
        document.createTable(accumulations.size() + 1, 9);
        addNewLine(document);
        document.createTable(financialArrangements.size() + 1, 11);
        addNewLine(document);
    }

    private void fillTransactionTable(XWPFTable table, List<Transaction>
transactions) {
        fillTableHead(table.getRow(0).getTableCells(),
TRANSACTION_TABLE_COLUMNS_NAME);

        for (int i = 1; i < table.getNumberOfRows(); i++) {
            Transaction transaction = transactions.get(i - 1);
            List<XWPFTableCell> cells = table.getRow(i).getTableCells();

            cells.get(0).setText(i + "");
            cells.get(1).setText(transaction.getSum() + "");
            cells.get(2).setText(transaction.isRefill() + "");
            cells.get(3).setText(transaction.getComment());
```

```
            cells.get(4).setText(transaction.getCategory() + "");
            cells.get(5).setText(transaction.getDateTime() + "");
        }
    }

    private void fillAccumulationsTable(XWPFTable table, List<Accumulation>
accumulations) {
        fillTableHead(table.getRow(0).getTableCells(),
ACCUMULATIONS_TABLE_COLUMNS_NAME);

        for(int i = 1; i < table.getNumberOfRows(); i++) {
            Accumulation accumulation = accumulations.get(i - 1);
            List<XWPFTableCell> cells = table.getRow(i).getTableCells();

            cells.get(0).setText(i + "");
            cells.get(1).setText(accumulation.getName());
            cells.get(2).setText(accumulation.getComment());
            cells.get(3).setText(accumulation.getCurrentSum() + "");
            cells.get(4).setText(accumulation.getGoalSum() + "");
            cells.get(5).setText(accumulation.getStartDate() + "");
            cells.get(6).setText(accumulation.getEndDate() + "");
            cells.get(7).setText(accumulation.getLastPaymentDate() + "");
            cells.get(8).setText(accumulation.getStatus() + "");
        }
    }

    private void fillFinancialArrangementTable(XWPFTable table,
List<FinancialArrangement> arrangements) {
        fillTableHead(table.getRow(0).getTableCells(),
FINANCIAL_ARRANGEMENT_TABLE_COLUMNS_NAME);

        for(int i = 1; i < table.getNumberOfRows(); i++) {
            FinancialArrangement arrangement = arrangements.get(i - 1);
            List<XWPFTableCell> cells = table.getRow(i).getTableCells();

            cells.get(0).setText(i + "");
            cells.get(1).setText(arrangement.getName());
            cells.get(2).setText(arrangement.getState() + "");
            cells.get(3).setText(arrangement.getPercent() + "%");
            cells.get(4).setText(arrangement.getStartSum() + "");
            cells.get(5).setText(arrangement.getCurrentSum() + "");
            cells.get(6).setText(arrangement.getRefundSum() + "");
            cells.get(7).setText(arrangement.getStartDate() + "");
            cells.get(8).setText(arrangement.getEndDate() + "");
            cells.get(9).setText(arrangement.isFromToUserFunds() + "");
            cells.get(10).setText(arrangement.getStatus() + "");
        }
    }

    private void fillTableHead(List<XWPFTableCell> cells, List<String>
columnName){
        for(int i = 0; i < cells.size(); i++) {
            cells.get(i).setText(columnName.get(i));
        }
    }

    private void addNewLine(XWPFDocument document) {
        var paragraph = document.createParagraph();
        var run = paragraph.createRun();
        run.addBreak();
    }
}
```

Тепер у мене є можливість звертатися до цього url з інших мікросервісів та отримувати файл.

Приклад використання:

```
String statisticFileUrl = "http://db-microservice-url/statisticsFile";

StatisticFileRequest request = new StatisticFileRequest();
//Потрібно додати логіку заповнення


HttpHeaders headers = new HttpHeaders();
HttpEntity<StatisticFileRequest> httpEntity = new HttpEntity<>(request,
headers);


ResponseEntity<byte[]> responseEntity = new RestTemplate().exchange(
        statisticFileUrl,
        HttpMethod.GET,
        httpEntity,
        byte[].class
);
```

Висновок: У результаті виконання лабораторної роботи було вивчено основні принципи та концепції Службової Орієнтованої Архітектури (SOA). Реалізація сервісно-орієнтованої архітектури дозволяє створювати гнучкі та розширювані програмні системи, що легко піддаються змінам та розвитку. Використання незалежних служб сприяє покращенню масштабованості, перевикористання коду та полегшенню інтеграції між компонентами.