Pemrograman Platform Khusus

Layanan Web Service Peminjaman Laboratorium Komputer Polstat STIS

Dosen : Ibnu Santoso, SST, MT

DISUSUN OLEH:

Agape Bagus Rega Anggara (222212455)

Kelas : 3SI2

POLITEKNIK STATISTIKA STIS

Jl. Otto Iskandardinata No. 64C, Jakarta 13330

Telp. (021) 8508812, 8191473, Fax. 8197577

Website : www.stis.ac.id, E-Mail : info@stis.ac.id

Tahun Ajaran 2024/2025

A. LATAR BELAKANG DAN TUJUAN

Politeknik Statistika STIS adalah institusi pendidikan tinggi yang fokus pada pendidikan dan pengembangan ilmu statistik di Indonesia. Dalam mendukung proses pembelajaran yang optimal, Polstat STIS menyediakan berbagai fasilitas penunjang, termasuk laboratorium komputer yang dilengkapi perangkat lunak dan perangkat keras terkini. Fasilitas laboratorium komputer ini digunakan untuk berbagai kegiatan akademik, seperti praktikum, penelitian, dan pelatihan teknis, yang melibatkan banyak mahasiswa dari berbagai program studi. Namun, dengan tingginya permintaan penggunaan laboratorium komputer, diperlukan sistem yang mampu mengelola peminjaman laboratorium secara efisien dan terstruktur agar pemanfaatan laboratorium dapat maksimal dan tidak terjadi benturan jadwal.

Seiring dengan perkembangan teknologi dan semakin kompleksnya kebutuhan pengelolaan peminjaman fasilitas, Polstat STIS membutuhkan web servis yang dapat mengelola peminjaman laboratorium komputer secara efektif. Sistem ini diharapkan dapat memfasilitasi proses reservasi, pemantauan jadwal, serta pencatatan peminjaman laboratorium komputer secara otomatis dan transparan. Dengan adanya web servis ini, mahasiswa dan staf dapat mengakses informasi ketersediaan laboratorium dengan mudah dan melakukan reservasi sesuai kebutuhan, sehingga kegiatan akademik yang melibatkan laboratorium dapat berjalan dengan lancar dan terorganisir.

Web servis peminjaman laboratorium komputer Polstat STIS dibentuk dengan tujuan:

a) Mempermudah Proses Reservasi Laboratorium Komputer
b) Mengoptimalkan Penggunaan Laboratorium Komputer
c) Meningkatkan Transparansi dan Akurasi Data Peminjaman.

B. PROSES BISNIS

Pada kali ini, saya akan membuat layanan web service peminjaman laboratorium komputer Polstat STIS yang akan terdiri dari kurang lebih 5 bisnis utama, diantaranya adalah registrasi akun, peminjaman, pengubahan status komputer, penggantian password serta profil, dan penghapusan akun. Berikut adalah tahapan dari proses bisnis registrasi akun:

- Dengan *method post*, pengguna dapat mengakses http://localhost:8088/register untuk membuat akun baru.
- Lalu pengguna mengisi *body request* dengan tipe *json* yang berisi nim, nama, kelas, email, *password*, dan *role*, lalu kirim.
- Jika berhasil, maka pengguna akan mendapatkan *response* 200 dan akan terdaftar mahasiswa jika mengisi *role* mahasiswa, dan sebagai admin jika mengisi *role* admin.

Berikut adalah tahapan proses bisnis peminjaman komputer:

- Peminjaman komputer dapat dilakukan oleh pengguna dengan *role* mahasiswa.
- Pengguna mengakses http://localhost:8088/peminjaman dengan *method post* lalu mengisi *request body* bertipe *json* dengan waktu peminjaman.
- Pengguna dapat mulai mengisi data diri melalui http://localhost:8088/peminjaman/{id}/user dengan *method put* lalu mengisi *request body* dengan header bertipe *text/uri-list* dengan isian http://localhost:8088/user/{id} dan mengisi *end point* /{id} dengan id yang di kehendaki.
- Pengguna mengisi *komputer* yang ingin dipinjam melalui http://localhost:8088/peminjaman/{id}/komputer dengan *method put* lalu mengisi *request body* dengan header bertipe *text/uri-list* dengan isian http://localhost:8088/komputer/{id} dan mengisi *end point* /{id} dengan id yang dikehendaki.
- Jika pengguna sudah selesai meminjam, pengguna dapat mengakses http://localhost:8088/peminjaman/{id} dengan *method post* lalu mengisi *body request* bertipe *json* dengan isian waktu pengembalian.

Berikut adalah tahapan proses bisnis pengubahan status komputer:

- Pengguna dengan *role* admin memiliki peran untuk mengatur status komputer.
- Admin mengakses http://localhost:8088/komputer/{id} dengan *method patch*, lalu memilih komputer mana yang ingin diubah statusnya. Admin dapat mengisi *end point* /{id} dengan id komputer yang diinginkan.
- Admin mengisi *request body* bertipe *json* dengan isian status "Tidak Tersedia", "Baru", "Siap Digunakan", "Sedang Diperbaiki".
- Admin mengirim *body request* dan status komputer akan langsung berubah.

Berikut adalah tahapan proses bisnis penggantian profil dan *password*:

- Untuk mengganti profil, maka pengguna dapat mengakses http://localhost:8088/user/{id} dengan *method patch.*
- Pengguna mengisi *body request* bertipe json dengan isian profil yang ingin di edit, lalu mengirimnya. *Response* 200 akan keluar jika telah berhasil mengedit profil.
- Untuk mengganti password, maka pengguna dapat mengakses http://localhost:8088/changePassword dengan *method post.*
- Pengguna mengisi *body request* bertipe json dengan isian NIM dan *password* yang baru. *Response* 200 akan keluar jika telah berhasil mengubah *password*.

Berikut adalah tahapan proses bisnis penghapusan akun:

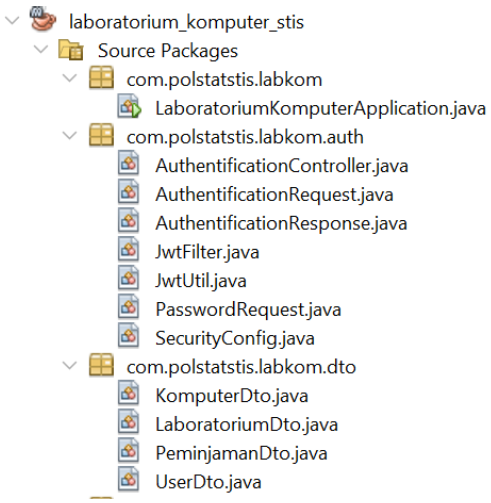
- Untuk menghapus akun, maka pengguna dapat mengakses http://localhost:8088/user/{id} dengan menggunakan *method delete*.
- Jika berhasil menghapus akun, maka pengguna akan mendapat *response* 200.

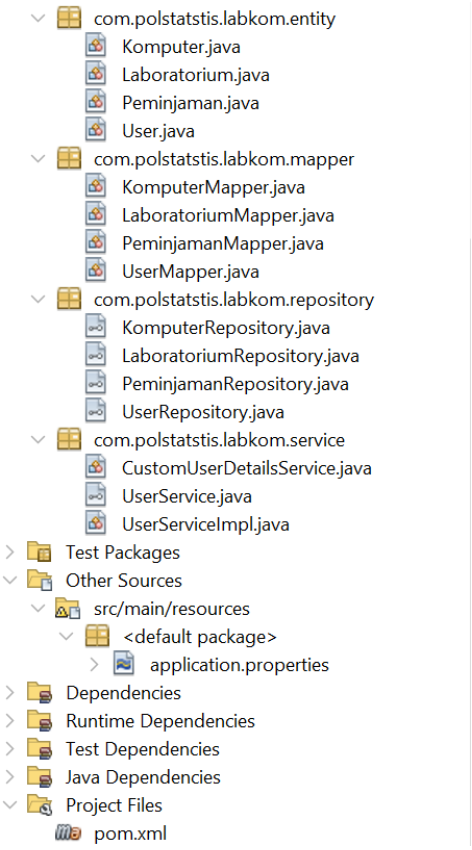
C. KODE PROGRAM LAYANAN WEB SERVICE PEMINJAMAN LABORATORIUM KOMPUTER POLSTAT STIS

Kode program yang digunakan pada Layanan Web Servis Peminjaman Laboratorium Komputer Polstat STIS dapat diakses pada link berikut https://git.stis.ac.id/222212455/UTS_PPK_3SI2_Agape-Bagus-Rega-Anggara_222212455.git Lalu untuk penjelasannya, akan saya jelaskan dibawah ini.

- Struktur Program

  Berikut adalah tampak dari struktur program yang saya gunakan untuk membuat Layanan Web Servis Peminjaman Laboratorium Komputer Polstat STIS.





- *Package* com.polstatstis.labkom

  Package ini menyimpan semua *package* yang ada, namun di sini terdapat kelas utama yang berfungsi untuk dapat menjalankan program layanan web service ini.

  - Labkom.Application.java

```
package com.polstatstis.labkom;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LaboratoriumKomputerApplication {

    public static void main(String[] args) {
        SpringApplication.run(LaboratoriumKomputerApplication.
class, args);
    }
```

```
}
```

- *Package* com.polstatstis.labkom.auth

  Package ini mengatur proses autentikasi pada program yang juga mencakup register dan proses konfigurasi untuk keamanan program.

  - AuthentificationController.java

```
package com.polstatstis.labkom.auth;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.dto.UserDto;
import com.polstatstis.labkom.service.UserService;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.authentication.AuthenticationMana
ger;
import
org.springframework.security.authentication.BadCredentialsExce
ption;
import
org.springframework.security.authentication.UsernamePasswordAu
thenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class AuthentificationController {
    @Autowired
    AuthenticationManager authManager;

    @Autowired
    JwtUtil jwtUtil;

    @Autowired
    UserService userService;

    @PostMapping("/changePassword")
    public ResponseEntity<?> changePassword(@RequestBody
PasswordRequest request) {
```

```
        UserDto user =
userService.changePassword(request.getNim(),request.getPasswor
d());
        return ResponseEntity.ok().body(user);
    }

    @PostMapping("/register")
    public ResponseEntity<?> register(@RequestBody UserDto
request) {
        if (!request.getRole().equals("admin") &&
!request.getRole().equals("mahasiswa")) {
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST)
                                    .body("Role harus admin atau
mahasiswa.");
        }
        UserDto user = userService.createUser(request);
        return ResponseEntity.ok().body(user);
    }

    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody @Valid
AuthentificationRequest request) {
        try {
            Authentication authentication =
authManager.authenticate(
                new
UsernamePasswordAuthenticationToken(request.getNim(),
request.getPassword()));
            String accessToken =
jwtUtil.generateAccessToken(authentication);
            AuthentificationResponse response = new
AuthentificationResponse(request.getNim(), accessToken);
            return ResponseEntity.ok().body(response);
        } catch (BadCredentialsException ex) {
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("NIM atau
password salah.");
        }
    }
}
```

- AuthentificationRequest.java

```
package com.polstatstis.labkom.auth;

/**
 * @author Agape Bagus Rega Anggara (222212455)
```

```
 */

import jakarta.validation.constraints.Max;
import jakarta.validation.constraints.NotNull;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class AuthentificationRequest {
    @NotNull @Max(20)
    private String nim;
    @NotNull @Max(16)
    private String password;
}
```

- AuthentificationResponse.java

```
package com.polstatstis.labkom.auth;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class AuthentificationResponse {
    private String nim;
    private String accessToken;
}
```

- JwtFilter.java

```
package com.polstatstis.labkom.auth;

/**
 * @author Agape Bagus Rega Anggara (222212455)
```

```
 */

import com.polstatstis.labkom.dto.UserDto;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.authentication.UsernamePasswordAu
thenticationToken;
import
org.springframework.security.core.context.SecurityContextHolde
r;
import
org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.web.authentication.WebAuthenticat
ionDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.util.ObjectUtils;
import org.springframework.web.filter.OncePerRequestFilter;

@Component
public class JwtFilter extends OncePerRequestFilter{
    @Autowired
    private JwtUtil jwtUtil;

    private boolean hasAuthorizationBearer(HttpServletRequest
request) {
        String header = request.getHeader("Authorization");
        if (ObjectUtils.isEmpty(header) ||
!header.startsWith("Bearer")) {
            return false;
        }
        return true;
    }

    private String getAccessToken(HttpServletRequest request)
{
        String header = request.getHeader("Authorization");
        String token = header.split(" ")[1].trim();
        return token;
    }

    private UserDetails getUserDetails(String token) {
        String subject = jwtUtil.getSubject(token);
```

```
        return UserDto.builder().email(subject).build();
    }

    private void setAuthenticationContext(String token,
HttpServletRequest request) {
        UserDetails userDetails = getUserDetails(token);
        UsernamePasswordAuthenticationToken authentication =
new
        UsernamePasswordAuthenticationToken(userDetails, null,
null);
        authentication.setDetails(
        new
WebAuthenticationDetailsSource().buildDetails(request));
        SecurityContextHolder.getContext().setAuthentication(a
uthentication);
    }

    @Override
    protected void doFilterInternal(HttpServletRequest
request, HttpServletResponse response,
    FilterChain filterChain)throws ServletException,
IOException {
        if (!hasAuthorizationBearer(request)) {
            filterChain.doFilter(request, response);
            return;
        }
        String token = getAccessToken(request);
        if (!jwtUtil.validateAccessToken(token)) {
            filterChain.doFilter(request, response);
            return;
        }
        setAuthenticationContext(token, request);
        filterChain.doFilter(request, response);
    }
}
```

- JwtUtil.java

```
package com.polstatstis.labkom.auth;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.ExpiredJwtException;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.MalformedJwtException;
```

```java
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.SignatureException;
import io.jsonwebtoken.UnsupportedJwtException;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import java.util.Date;
import org.springframework.security.core.Authentication;
import
org.springframework.security.core.userdetails.UserDetails;

@Component
public class JwtUtil {
    @Value("${jwt.secret}")
    private String SECRET_KEY;

    @Value("${jwt.expiration}")
    private long EXPIRE_DURATION;

    public String generateAccessToken(Authentication
authentication){
        UserDetails userDetails = (UserDetails)
authentication.getPrincipal();
        return Jwts.builder()
        .setSubject(userDetails.getUsername())
        .setIssuer("Polstat")
        .setIssuedAt(new Date())
        .setExpiration(new
Date(System.currentTimeMillis()+EXPIRE_DURATION))
        .signWith(SignatureAlgorithm.HS512, SECRET_KEY)
        .compact();
    }

    public String getSubject(String token) {
        return parseClaims(token).getSubject();
    }

    private Claims parseClaims(String token) {
        return Jwts.parser()
        .setSigningKey(SECRET_KEY)
        .parseClaimsJws(token)
        .getBody();
    }

    public boolean validateAccessToken(String token) {
        try {
            Jwts.parser().setSigningKey(SECRET_KEY).parseClaim
sJws(token);
            return true;
```

```
        } catch (ExpiredJwtException ex) {
            System.out.println("JWT expired" +
ex.getMessage());
        } catch (IllegalArgumentException ex) {
            System.out.println("Token is null, empty or only
whitespace" +
            ex.getMessage());
        } catch (MalformedJwtException ex) {
            System.out.println("JWT is invalid" + ex);
        } catch (UnsupportedJwtException ex) {
            System.out.println("JWT is not supported" + ex);
        } catch (SignatureException ex) {
            System.out.println("Signature validation failed");
        }
        return false;
    }
}
```

- PasswordRequest.java

```
package com.polstatstis.labkom.auth;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import jakarta.validation.constraints.Max;
import jakarta.validation.constraints.NotNull;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class PasswordRequest {
    @NotNull @Max(20)
    private String nim;
    @NotNull @Max(16)
    private String password;
}
```

- SecurityConfig.java

```
package com.polstatstis.labkom.auth;
```

```java
/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import
com.polstatstis.labkom.service.CustomUserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.AuthenticationMana
ger;
import
org.springframework.security.config.annotation.authentication.
builders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.Ht
tpSecurity;
import
org.springframework.security.config.annotation.web.configurati
on.EnableWebSecurity;
import
org.springframework.security.config.http.SessionCreationPolicy
;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncod
er;
import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswo
rdAuthenticationFilter;
import static
org.springframework.security.web.util.matcher.AntPathRequestMa
tcher.antMatcher;

@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Autowired
    private JwtFilter jwtTokenFilter;
    private final CustomUserDetailsService userDetailsService;

    public SecurityConfig(CustomUserDetailsService
userDetailsService) {
        this.userDetailsService = userDetailsService;
    }
```

```
    @Bean
    public AuthenticationManager
authenticationManager(HttpSecurity http, PasswordEncoder
passwordEncoder)
    throws Exception {
        AuthenticationManagerBuilder
authenticationManagerBuilder = http.
                getSharedObject(AuthenticationManagerBuilder.c
lass);
        authenticationManagerBuilder.userDetailsService(userDe
tailsService).passwordEncoder(passwordEncoder);
        return authenticationManagerBuilder.build();
    }

    @Bean
    public SecurityFilterChain
securityFilterChain(HttpSecurity http) throws Exception{
        http.csrf().disable();
        http.sessionManagement().sessionCreationPolicy(Session
CreationPolicy.STATELESS);
        http.authorizeRequests()
            .requestMatchers(antMatcher("/login"),
antMatcher("/register"), antMatcher("/changePassword"))
                .permitAll() // Endpoint ini diizinkan untuk
diakses semua pengguna
            .anyRequest()
                .authenticated(); // Semua endpoint lainnya
memerlukan autentikasi
        http.addFilterBefore(jwtTokenFilter,
UsernamePasswordAuthenticationFilter.class);
        return http.build();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

- *Package* com.polstatstis.labkom.dto

  Package ini mengatur setiap pembentukan objek di dalam aplikasi. Objek ini merupakan hasil pembentukan dari suatu baris table di dalam basis data.

  - KomputerDto.java

```
package com.polstatstis.labkom.dto;
```

```java
/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.entity.Laboratorium;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class KomputerDto {
    private int id_Komputer;

    private Laboratorium laboratorium;

    private String status;
}
```

- LaboratoriumDto.java

```java
package com.polstatstis.labkom.dto;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class LaboratoriumDto {
    private int id_Laboratorium;

    private String nama_Laboratorium;

    private String lokasi;
}
```

- PeminjamanDto.java

```java
package com.polstatstis.labkom.dto;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.entity.Komputer;
import com.polstatstis.labkom.entity.User;
import java.util.Date;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.format.annotation.DateTimeFormat;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class PeminjamanDto {
    private int id_Peminjaman;

    private User user;

    private Komputer komputer;

    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
    private Date waktu_Peminjaman;

    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
    private Date waktu_Pengembalian;
}
```

- UserDto.java

```java
package com.polstatstis.labkom.dto;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.security.core.GrantedAuthority;
import
org.springframework.security.core.userdetails.UserDetails;
```

```java
import java.util.Collection;
import java.util.List;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class UserDto implements UserDetails{
    private String nim;
    private String role;
    private String nama;
    private String email;
    private String password;
    private String kelas;

    @Override
    public Collection<? extends GrantedAuthority>
getAuthorities() {
        return List.of((GrantedAuthority) () -> "ROLE_" +
this.role);
    }

    @Override
        public String getUsername() {
//        if (this.role=="ADMIN")
//            return this.role;
        return this.nim;
    }

    @Override
        public boolean isAccountNonLocked() {
        return true;
    }

    @Override
        public boolean isAccountNonExpired() {
        return true;
    }

    @Override
        public boolean isEnabled() {
        return true;
    }

    @Override
        public boolean isCredentialsNonExpired() {
        return true;
    }
```

```
}
```

- *Package* com.polstatstis.labkom.entity

  Package ini mengatur setiap pembentukan table dan entitas di dalam basis data.

  - Komputer.java

```java
package com.polstatstis.labkom.entity;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.hibernate.annotations.OnDelete;
import org.hibernate.annotations.OnDeleteAction;

@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "komputer")
public class Komputer {
    @Id
    private int id_Komputer;

    @ManyToOne
    @JoinColumn(name = "lab_id")
    @OnDelete(action = OnDeleteAction.CASCADE)
    @JsonIgnore
    private Laboratorium laboratorium;
```

```
    @Column(nullable = false)
    private String status;

    @OneToMany(mappedBy = "komputer")
    private List<Peminjaman> peminjaman;
}
```

- Laboratorium.java

```
package com.polstatstis.labkom.entity;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "laboratorium")
public class Laboratorium {
    @Id
    private int id_Laboratorium;

    @Column(nullable = false)
    private String nama_Laboratorium;

    @Column(nullable = false)
    private String lokasi;

    @OneToMany(mappedBy = "laboratorium")
    private List<Komputer> komputer;
}
```

- Peminjaman.java

```java
package com.polstatstis.labkom.entity;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Table;
import java.util.Date;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.hibernate.annotations.OnDelete;
import org.hibernate.annotations.OnDeleteAction;
import org.springframework.format.annotation.DateTimeFormat;


@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "peminjaman")
public class Peminjaman {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id_Peminjaman;

    @ManyToOne
    @JoinColumn(name = "user_id")
    @OnDelete(action = OnDeleteAction.CASCADE)
    @JsonIgnore
    private User user;

    @ManyToOne
    @JoinColumn(name = "komputer_id")
    @OnDelete(action = OnDeleteAction.CASCADE)
```

```
    @JsonIgnore
    private Komputer komputer;

    @Column(nullable = true)
    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
    private Date waktu_Peminjaman;

    @Column(nullable = true)
    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
    private Date waktu_Pengembalian;
}
```

- User.java

```
package com.polstatstis.labkom.entity;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;


@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "user")
public class User {
    @Id
    private String nim;

    @Column(nullable = false)
    private String nama;

    @Column(nullable = true)
```

```
    private String kelas;

    @Column(nullable = true)
    private String email;

    @Column(nullable = false)
    private String password;

    @Column(nullable = false)
    private String role;

    @OneToMany(mappedBy = "user")
    private List<Peminjaman> peminjaman;
}
```

- *Package* com.polstatstis.labkom.mapper

  Package ini mengatur seluruh pemetaan entitas di suatu basis data yang akan dikonversi ke dalam bentuk objek di dalam program.

  - KomputerMapper.java

```
package com.polstatstis.labkom.mapper;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.dto.KomputerDto;
import com.polstatstis.labkom.entity.Komputer;

public class KomputerMapper {
    static Komputer mapToKomputer(KomputerDto komputerDto){
        return Komputer.builder()
            .id_Komputer(komputerDto.getId_Komputer())
            .laboratorium(komputerDto.getLaboratorium())
            .status(komputerDto.getStatus())
            .build();
    }
    public static KomputerDto mapToKomputerDto(Komputer
komputer){
        return KomputerDto.builder()
            .id_Komputer(komputer.getId_Komputer())
            .laboratorium(komputer.getLaboratorium())
            .status(komputer.getStatus())
            .build();
    }
}
```

- LaboratoriumMapper.java

```
package com.polstatstis.labkom.mapper;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.dto.LaboratoriumDto;
import com.polstatstis.labkom.entity.Laboratorium;

public class LaboratoriumMapper {
    static Laboratorium mapToLab(LaboratoriumDto labDto){
        return Laboratorium.builder()
            .id_Laboratorium(labDto.getId_Laboratorium())
            .nama_Laboratorium(labDto.getNama_Laboratorium())
            .lokasi(labDto.getLokasi())
            .build();
    }
    public static LaboratoriumDto mapToLabDto(Laboratorium
lab){
        return LaboratoriumDto.builder()
            .id_Laboratorium(lab.getId_Laboratorium())
            .nama_Laboratorium(lab.getNama_Laboratorium())
            .lokasi(lab.getLokasi())
            .build();
    }
}
```

- PeminjamanMapper.java

```
package com.polstatstis.labkom.mapper;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.dto.PeminjamanDto;
import com.polstatstis.labkom.entity.Peminjaman;

public class PeminjamanMapper {
    static Peminjaman mapToPeminjaman(PeminjamanDto
peminjamanDto){
        return Peminjaman.builder()
            .id_Peminjaman(peminjamanDto.getId_Peminjaman())
            .user(peminjamanDto.getUser())
            .komputer(peminjamanDto.getKomputer())
```

```
            .waktu_Peminjaman(peminjamanDto.getWaktu_Peminjama
n())
            .waktu_Pengembalian(peminjamanDto.getWaktu_Pengemb
alian())
            .build();
    }
    public static PeminjamanDto mapToPeminjamanDto(Peminjaman
peminjaman){
        return PeminjamanDto.builder()
            .id_Peminjaman(peminjaman.getId_Peminjaman())
            .user(peminjaman.getUser())
            .komputer(peminjaman.getKomputer())
            .waktu_Peminjaman(peminjaman.getWaktu_Peminjaman()
)
            .waktu_Pengembalian(peminjaman.getWaktu_Pengembali
an())
            .build();
    }
}
```

- UserMapper.java

```
package com.polstatstis.labkom.mapper;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.dto.UserDto;
import com.polstatstis.labkom.entity.User;

public class UserMapper {
   public static User mapToUser(UserDto userDto){
        return User.builder()
            .nim(userDto.getNim())
            .nama(userDto.getNama())
            .kelas(userDto.getKelas())
            .email(userDto.getEmail())
            .password(userDto.getPassword())
            .role(userDto.getRole())
            .build();
    }
    public static UserDto mapToUserDto(User user){
        return UserDto.builder()
            .nim(user.getNim())
            .nama(user.getNama())
            .kelas(user.getKelas())
            .email(user.getEmail())
```

```
            .password(user.getPassword())
            .role(user.getRole())
            .build();
    }
}
```

- *Package* com.polstatstis.labkom.repository

  Package ini mengatur seluruh logika program yang berhubungan dengan CRUD.

  - KomputerRepository.java

```
package com.polstatstis.labkom.repository;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.entity.Komputer;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository
;
import
org.springframework.data.rest.core.annotation.RepositoryRestRe
source;

@RepositoryRestResource(collectionResourceRel = "komputer",
path = "komputer")
public interface KomputerRepository extends
PagingAndSortingRepository<Komputer,Long>,
CrudRepository<Komputer,Long>, JpaRepository<Komputer, Long>{

}
```

  - LaboratoriumRepository.java

```
package com.polstatstis.labkom.repository;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.entity.Laboratorium;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
```

```
import
org.springframework.data.repository.PagingAndSortingRepository
;
import
org.springframework.data.rest.core.annotation.RepositoryRestRe
source;

@RepositoryRestResource(collectionResourceRel =
"laboratorium", path = "laboratorium")
public interface LaboratoriumRepository extends
PagingAndSortingRepository<Laboratorium, Long>,
CrudRepository<Laboratorium, Long>,
JpaRepository<Laboratorium, Long>{

}
```

- PeminjamanRepository.java

```
package com.polstatstis.labkom.repository;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.entity.Peminjaman;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository
;
import
org.springframework.data.rest.core.annotation.RepositoryRestRe
source;

@RepositoryRestResource(collectionResourceRel = "peminjaman",
path = "peminjaman")
public interface PeminjamanRepository extends
PagingAndSortingRepository<Peminjaman,Long>,
CrudRepository<Peminjaman,Long>, JpaRepository<Peminjaman,
Long>{

}
```

- UserRepository.java

```
package com.polstatstis.labkom.repository;

/**
```

```
 * @author Agape Bagus Rega Anggara (222212455)
 */
import com.polstatstis.labkom.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository
;
import
org.springframework.data.rest.core.annotation.RepositoryRestRe
source;

@RepositoryRestResource(collectionResourceRel = "user", path =
"user")
public interface UserRepository extends
JpaRepository<User,String>,PagingAndSortingRepository<User,Str
ing>,
CrudRepository<User,String> {
    public User findByNim(String nim);
}
```

- *Package* com.polstatstis.labkom.service

  Package ini mengatur seluruh logika program yang berhubungan dengan *package* repository agar dapat digunakan di dalam controller.

  - CustomUserDetailsService.java

```
package com.polstatstis.labkom.service;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.repository.UserRepository;
import com.polstatstis.labkom.entity.User;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsServi
ce;
import
org.springframework.security.core.userdetails.UsernameNotFound
Exception;
import org.springframework.stereotype.Service;

@Service
```

```
public class CustomUserDetailsService implements
UserDetailsService{
    @Autowired
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String nim) throws
    UsernameNotFoundException {
        User user = userRepository.findByNim(nim);
        if (user == null) {
            throw new UsernameNotFoundException("User not
found with username: " +
        nim);
        }
        UserDetails userDetails =
org.springframework.security.core.userdetails.User.builder()
            .username(user.getEmail())
            .password(user.getPassword())
            .build();
        return userDetails;
    }
}
```

- UserService.java

```
package com.polstatstis.labkom.service;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */

import com.polstatstis.labkom.dto.UserDto;

public interface UserService {
    public UserDto createUser(UserDto user);
    public UserDto getUserByNim(String nim);
    public UserDto changePassword(String nim, String
password);
}
```

- UserServiceImpl.java

```
package com.polstatstis.labkom.service;

/**
 * @author Agape Bagus Rega Anggara (222212455)
 */
```

```java
import com.polstatstis.labkom.dto.UserDto;
import com.polstatstis.labkom.entity.User;
import com.polstatstis.labkom.mapper.UserMapper;
import com.polstatstis.labkom.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

@Service
public class UserServiceImpl implements UserService{
    @Autowired
    private UserRepository userRepository;

    @Autowired
    private PasswordEncoder passwordEncoder;


    @Override
    public UserDto createUser(UserDto userDto) {
        // Validasi role yang diperbolehkan
        if (!userDto.getRole().equalsIgnoreCase("admin") &&
!userDto.getRole().equalsIgnoreCase("mahasiswa")) {
            throw new IllegalArgumentException("Role harus
berupa 'admin' atau 'mahasiswa'.");
        }

        // Set password terenkripsi
        userDto.setPassword(passwordEncoder.encode(userDto.get
Password()));

        // Simpan user
        User user =
userRepository.save(UserMapper.mapToUser(userDto));
        return UserMapper.mapToUserDto(user);
    }

    @Override
    public UserDto changePassword(String nim, String password)
{
        User user = userRepository.findByNim(nim);
        if (user == null) {
            throw new RuntimeException("User not found with
NIM: " + nim);
        }
        user.setPassword(passwordEncoder.encode(password));
        user = userRepository.save(user);
        return UserMapper.mapToUserDto(user);
```

```
    }


    @Override
    public UserDto getUserByNim(String nim) {
        User user = userRepository.findByNim(nim);
        return UserMapper.mapToUserDto(user);
    }
}
```

- Other Sources

  Dibagian ini biasanya berisi source lain yang kita gunakan dalam program, tetapi kali ini saya hanya memerlukan application.properties yang berisi informasi database, server port, dan penghubungannya.

  - application.properties

```
#
#@author Agape Bagus Rega Anggara (222212455)
#

#dbms
spring.datasource.url=jdbc:mysql://localhost:3306/laboratorium
_komputer_stis
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-
platform=org.hibernate.dialect.MySQLDialect

#server
server.port = 8088
server.error.whitelabel.enabled=false

#jwt
jwt.secret=mysecretkey
jwt.expiration=86400000
```

- Poject Files

  Disini biasanya hanya terdapat file pom.xml yang berisi *dependencies* yang akan kita instal atau pakai.

  - pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.1.4</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.polstatstis</groupId>
    <artifactId>labkom</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>laboratorium_komputer_stis</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-
jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web-
services</artifactId>
        </dependency>

        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
```

```
            <scope>test</scope>
        </dependency>
                <dependency>
                        <groupId>jakarta.validation</groupId>
                        <artifactId>jakarta.validation-
api</artifactId>
                        <version>3.0.2</version>
                        <type>jar</type>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</gro
upId>
                        <artifactId>spring-boot-starter-data-
rest</artifactId>
                </dependency>
                <dependency>
                    <groupId>org.springframework.boot</groupId
>
                    <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
                <dependency>
                    <groupId>io.jsonwebtoken</groupId>
                    <artifactId>jjwt</artifactId>
                    <version>0.9.1</version>
                </dependency>
                <dependency>
                    <groupId>javax.xml.bind</groupId>
                    <artifactId>jaxb-api</artifactId>
                    <version>2.3.1</version>
                </dependency>
                <dependency>
                    <groupId>org.glassfish.jaxb</groupId>
                    <artifactId>jaxb-runtime</artifactId>
                </dependency>
                <dependency>
                    <groupId>org.springdoc</groupId>
                    <artifactId>springdoc-openapi-starter-
webmvc-ui</artifactId>
                    <version>2.1.0</version>
                </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
```

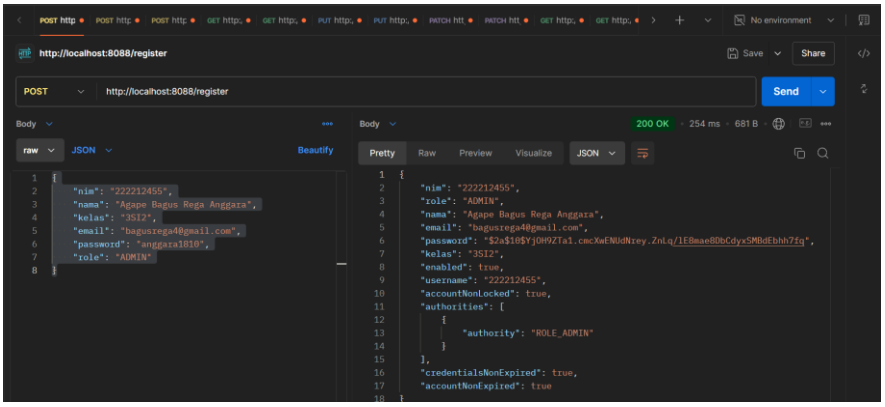
```
                <artifactId>spring-boot-maven-
plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupI
d>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

D. DOKUMENTASI END POINT

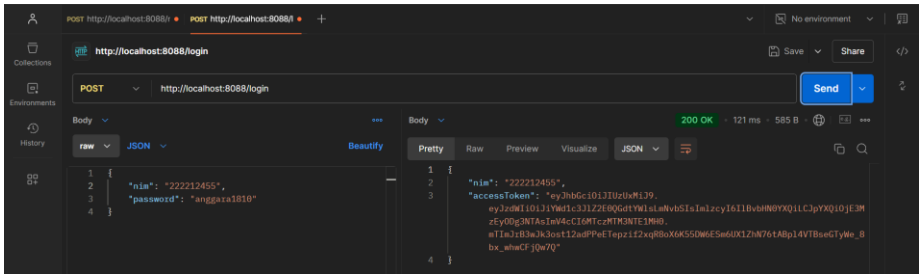
Berikut adalah beberapa dokumentasi yang saya lampirkan.

- Registrasi Akun



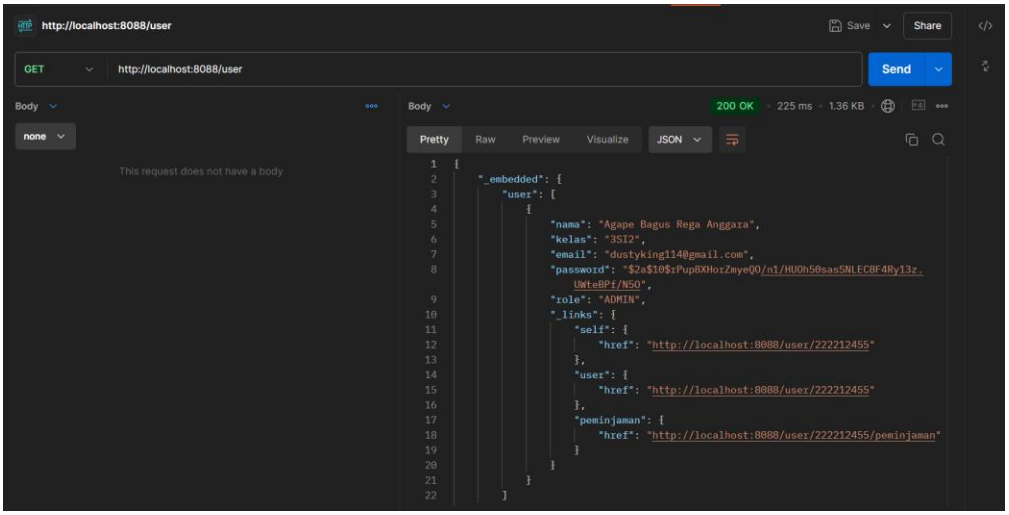
Pertama, kita lakukan registrasi akun terlebih dahulu dengan mengisikan NIM, Nama, Kelas, Email, Password, dan Role.

- Login



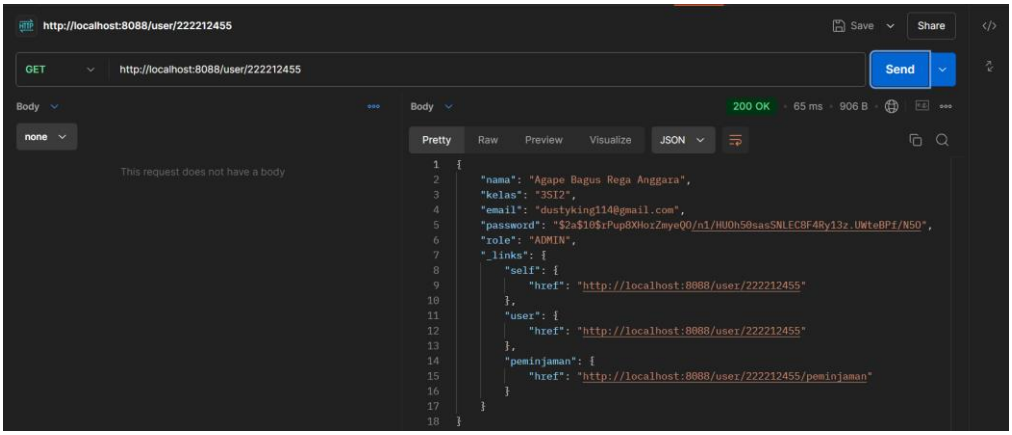
Pertama, kita melakukan login dengan nim dan password.

- Akses Data User



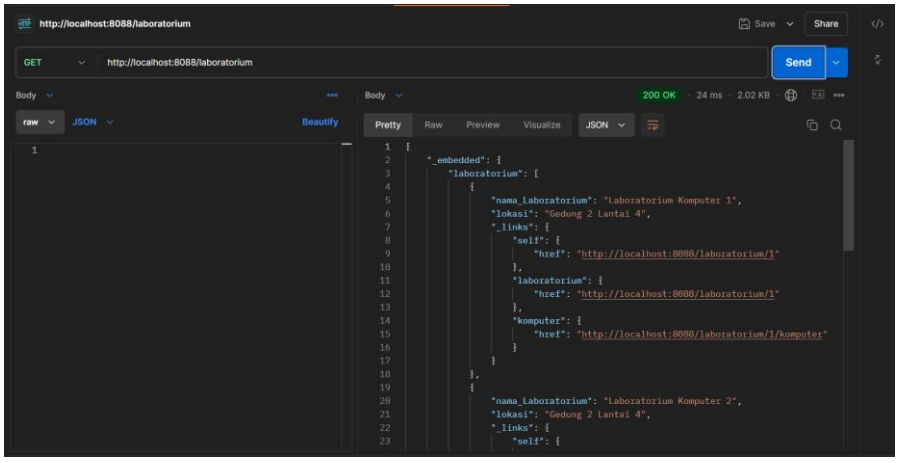
Dengan cara diatas, kita dapat melihat detail lengkap user. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

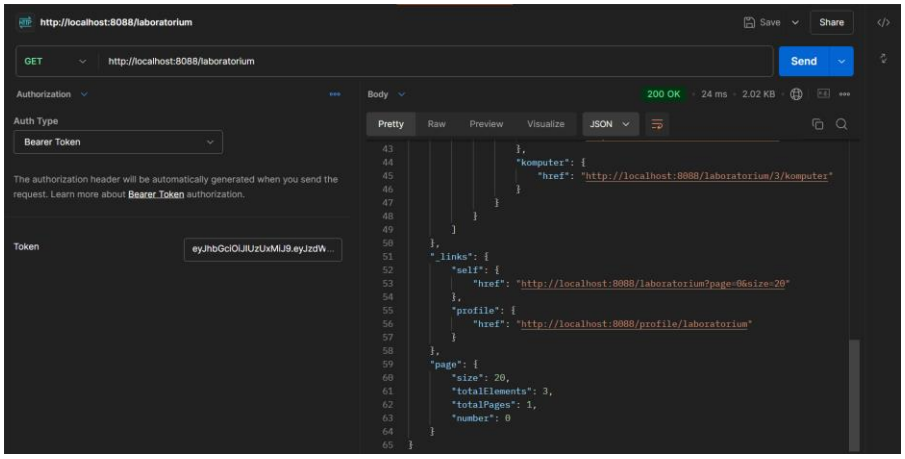
- Mencari User dengan NIM



Dengan cara diatas, kita bisa mencari user berdasarkan NIM-nya. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

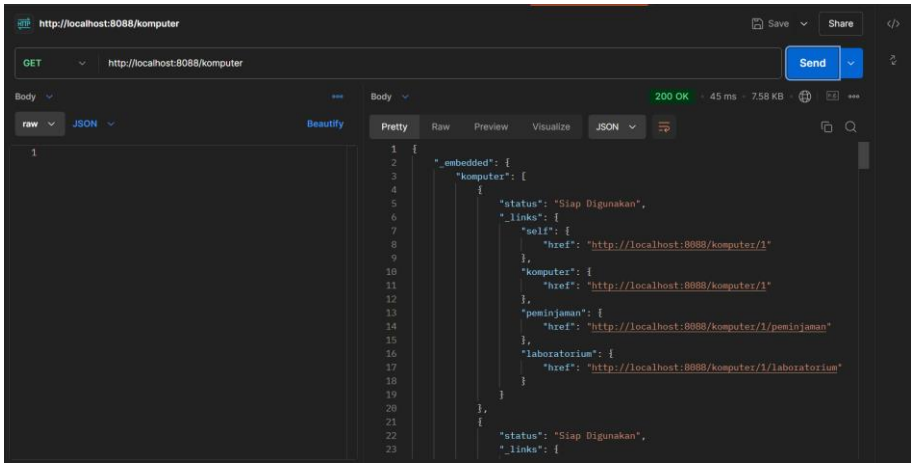
- Akses Data Laboratorium

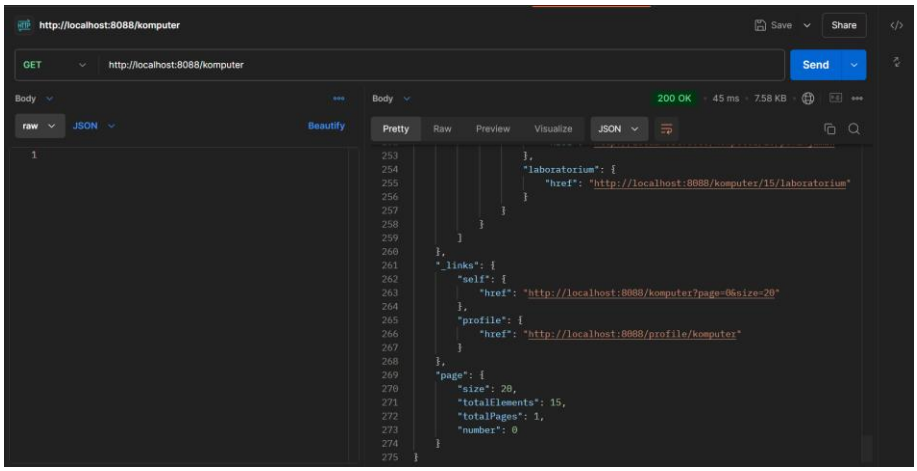




Terlihat bahwa terdapat 3 laboratorium yang terletak di gedung 2 lantai 4 Polstat STIS. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

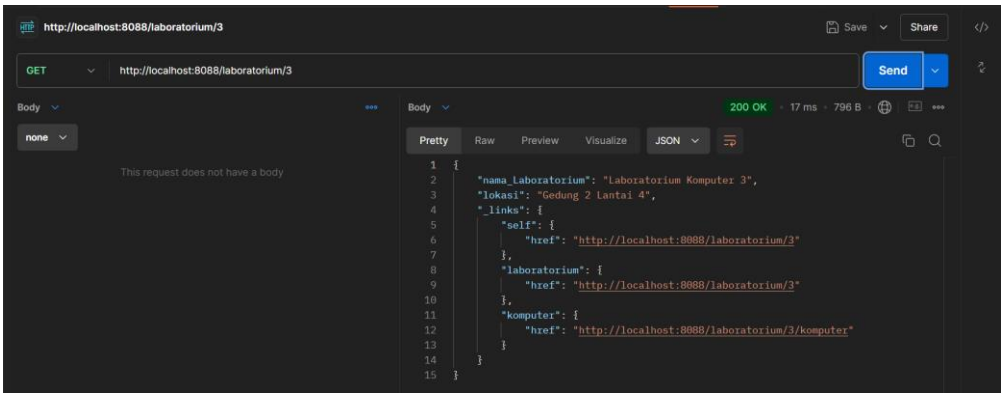
- Akses Data Komputer





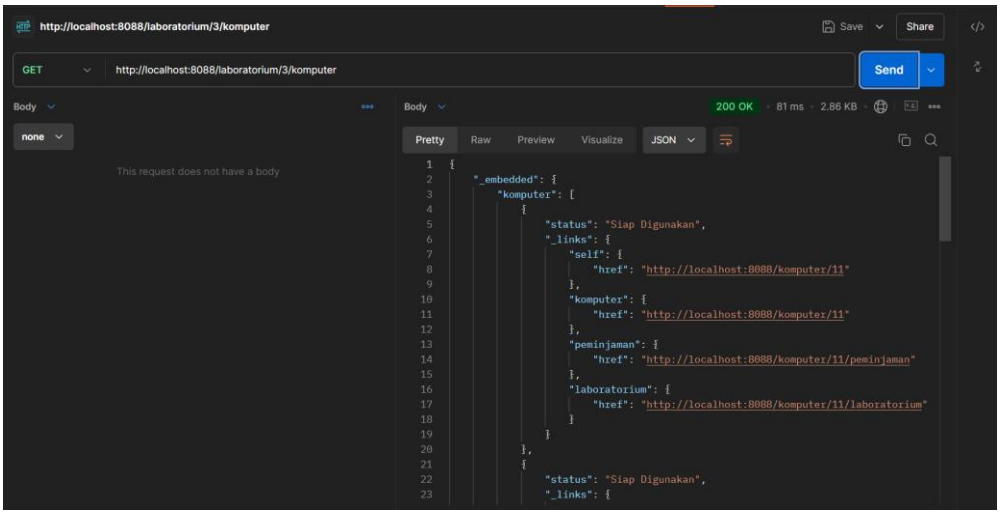
Terlihat bahwa ada 15 komputer tersedia yang terbagi dalam 3 laboratorium. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

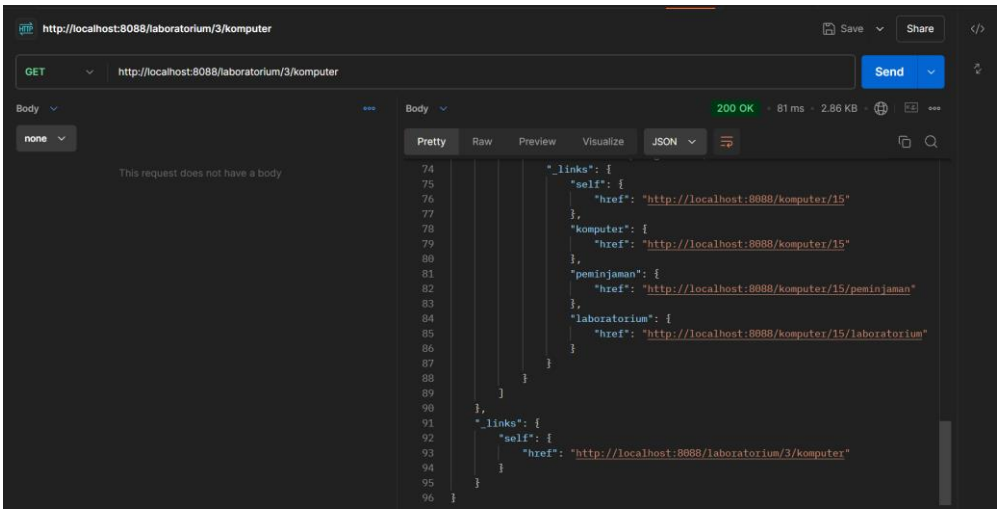
- Akses Data Laboratorium dengan ID



Dengan cara diatas, kita dapat mencari data laboratorium dengan ID-nya. Pada gambar diatas kita mencari detail Laboratorium dengan ID 3. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

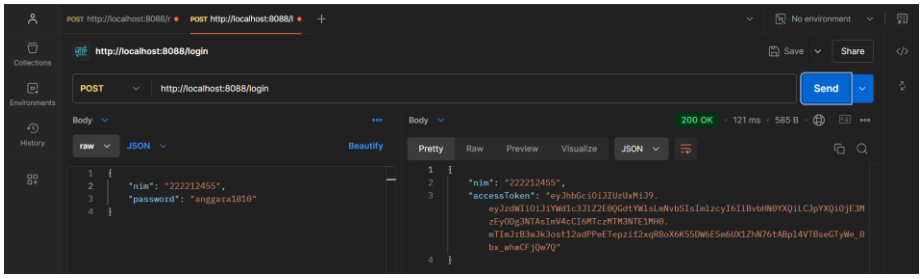
- Akses Daftar Komputer di suatu Laboratorium





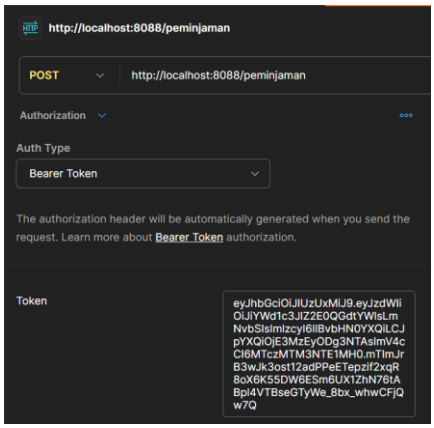
Dengan cara diatas kita dapat mengakses data komputer apa saja yang ada pada laboratorium tertentu, pada gambar diatas saya pakai laboratorium 3. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

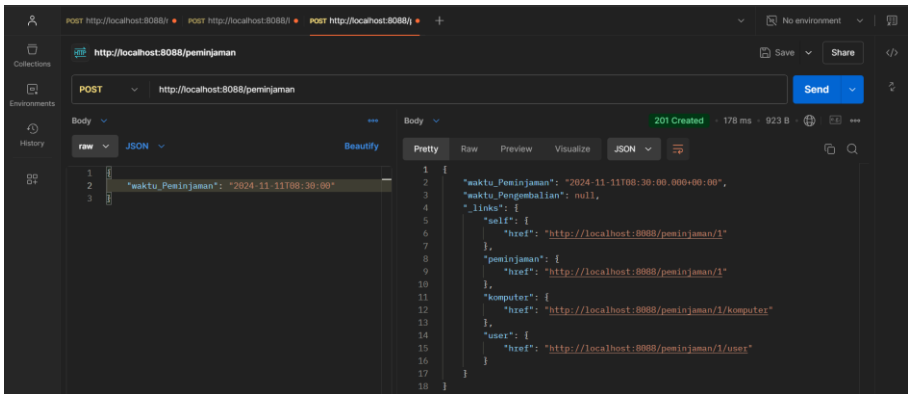
- Prosedur Peminjaman Komputer di Laboratorium Komputer
  - Login



Pertama, kita melakukan login dengan nim dan password.

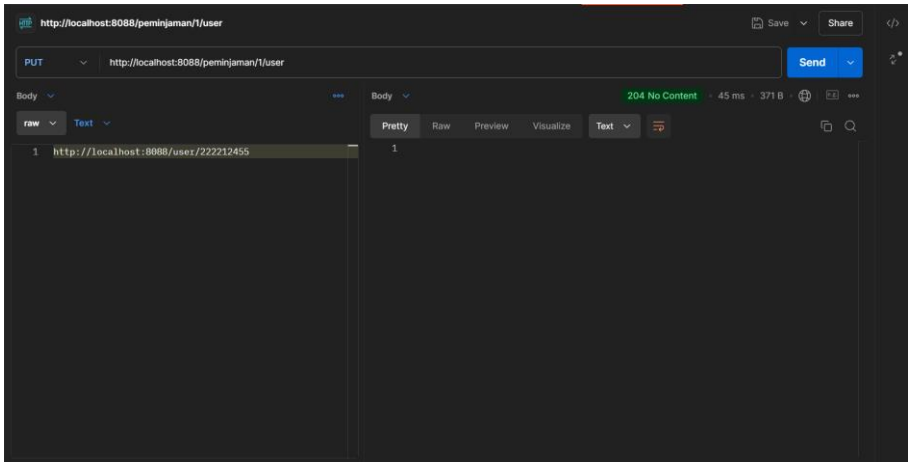
  - Membuat Pinjaman



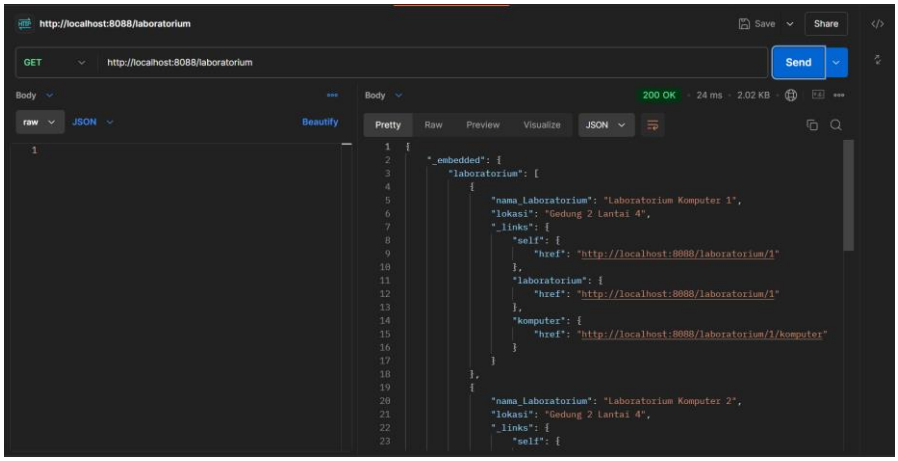


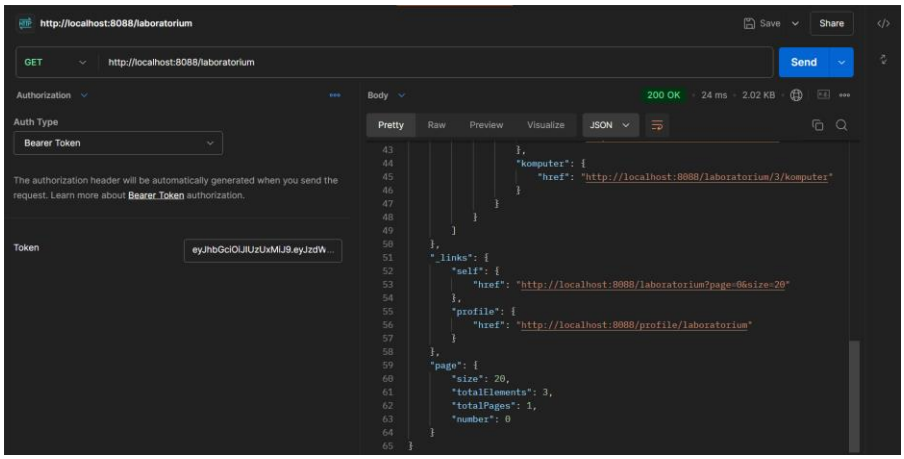
Kita membuat pinjaman dengan url diatas dan raw json yang mengisi waktu peminjaman komputer. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

  - Mengisi Data Pengguna Peminjam



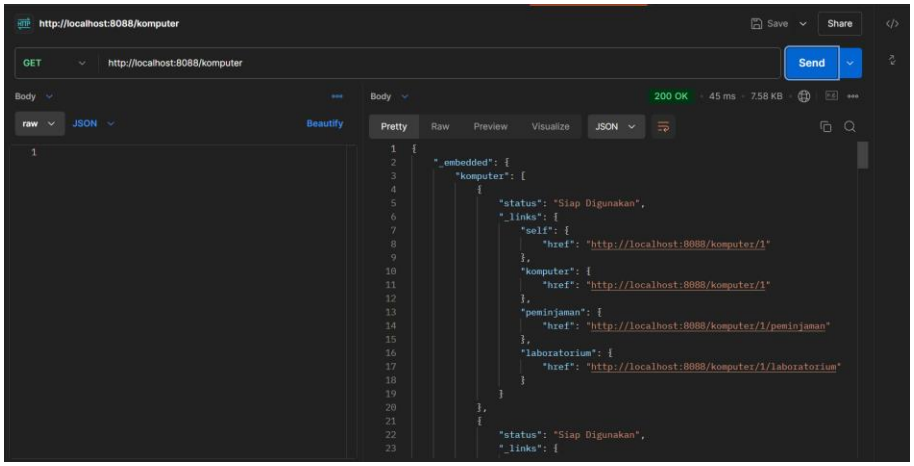
- Melihat Daftar Laboratorium

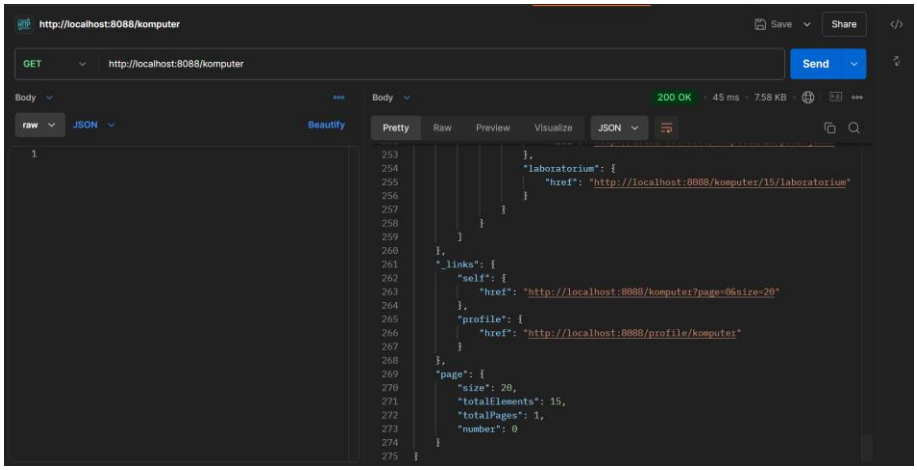




Terlihat bahwa terdapat 3 laboratorium yang terletak di gedung 2 lantai 4 Polstat STIS. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

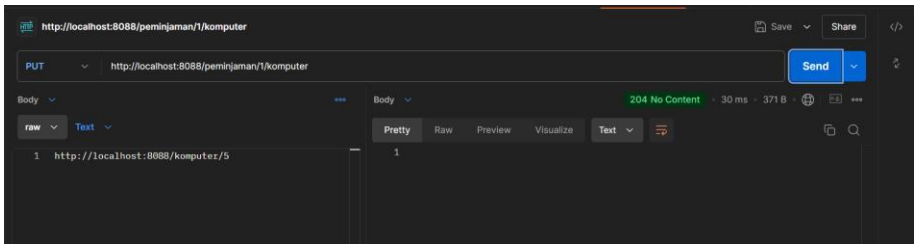
- Melihat Daftar Komputer





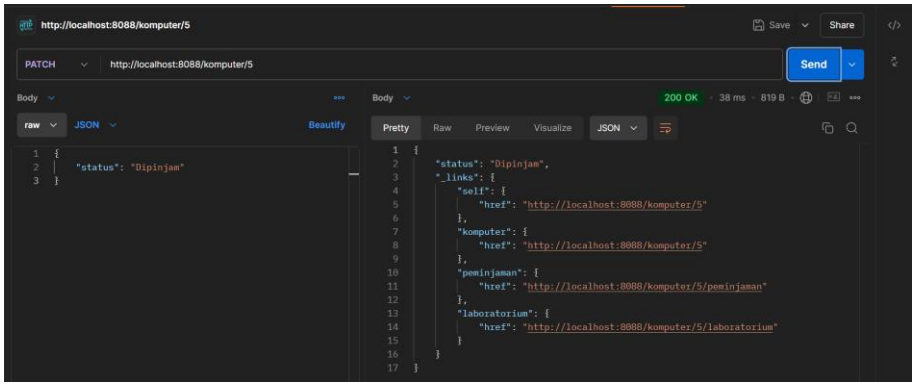
Terlihat bahwa ada 15 komputer tersedia yang terbagi dalam 3 laboratorium. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

- Memilih Komputer yang Ingin Dipinjam

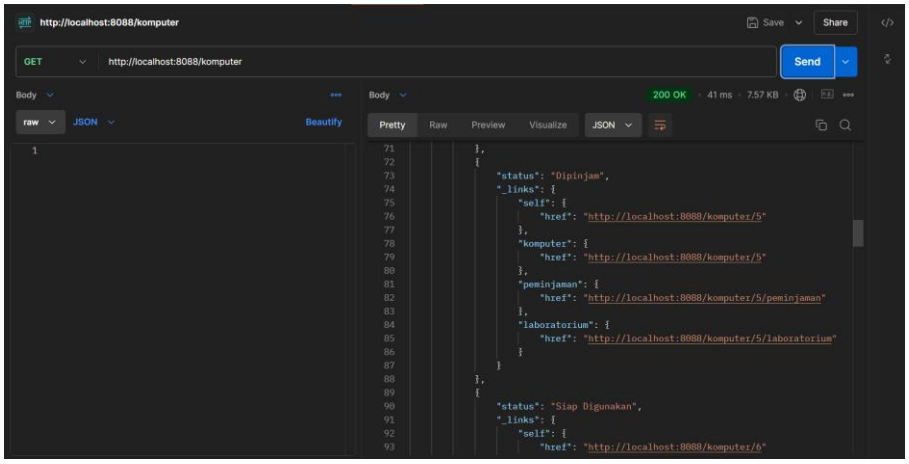


Disini saya akan meminjam komputer yang ber-id 5 yang bertempat di laboratorium 1 gedung 2 lantai 4 Polstst STIS. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

- Perubahan Status Komputer yang Dipinjam

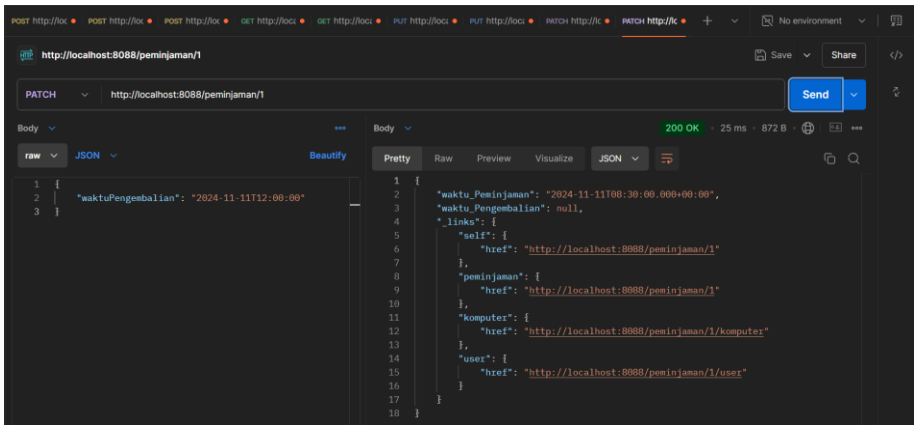


Jadi disini admin bisa mengubah status komputer yang telah diminta oleh pengguna menjadi “Dipinjam”. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.



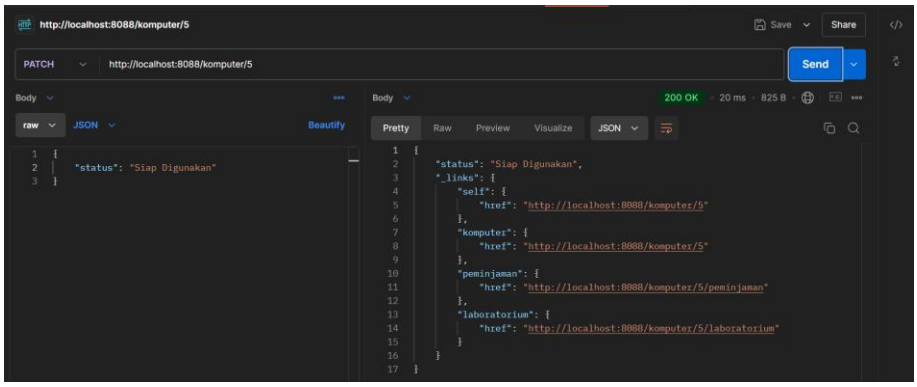
Sehingga ketika kita melihat keseluruhan status komputer, maka kita dapat melihat bahwa komputer dengan id 5 sedang dipinjam. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

- Pengembalian Komputer yang Telah Dipinjam

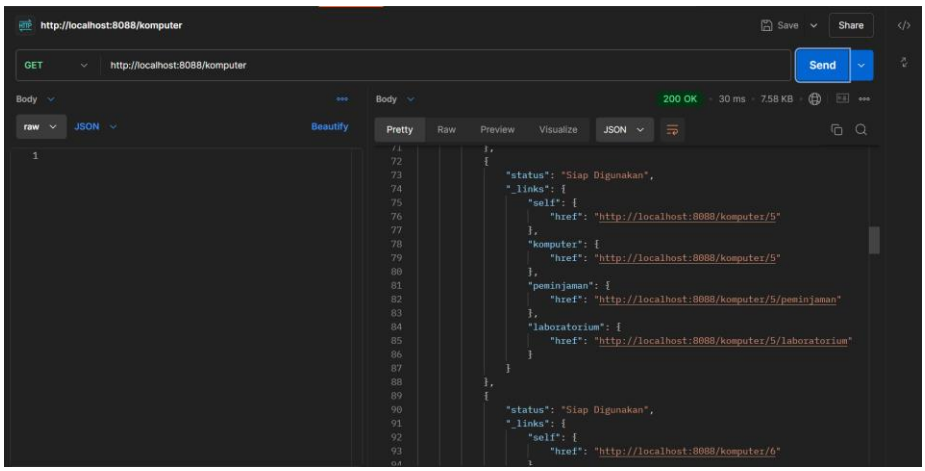


Kita membuat pinjaman dengan url diatas dan raw json yang mengisi waktu pengembalian komputer. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

➢ Perubahan Status Komputer yang Telah Dikembalikan

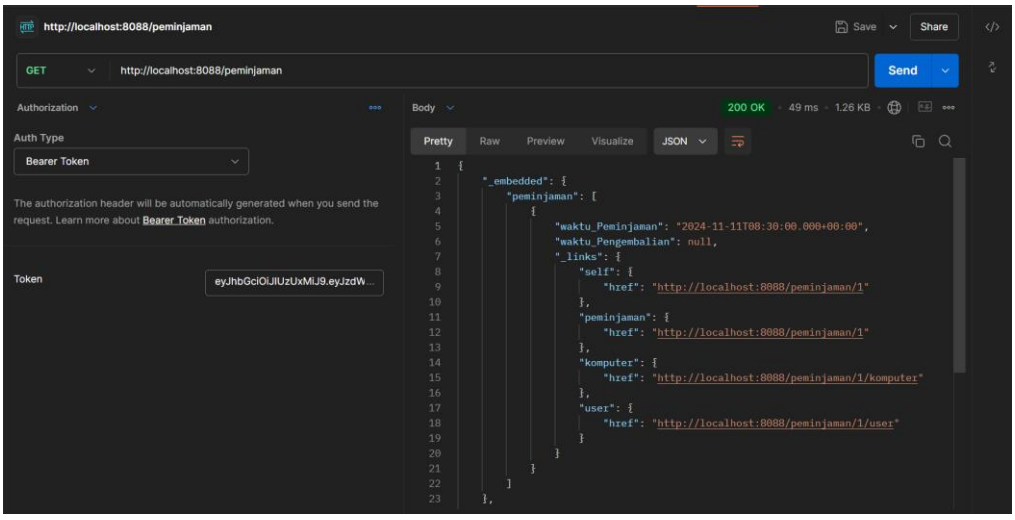


Jadi disini admin bisa mengubah status komputer yang telah diminta oleh pengguna menjadi “Siap Digunakan”. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.



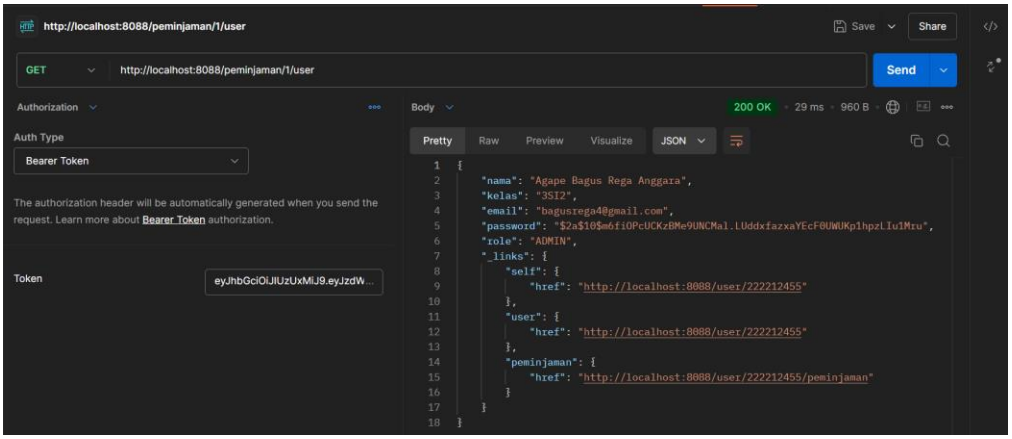
Sehingga ketika kita melihat keseluruhan status komputer, maka kita dapat melihat bahwa komputer dengan id 5 siap digunakan. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

- Akses Daftar Peminjaman



Disini kita dapat melihat semua peminjaman yang telah dilakukan. Berdasarkan gambar, terlihat bahwa peminjaman yang telah dilakukan adalah salah satunya pada tanggal 11-11-2024 pada pukul 08.30.00. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

- Detail User pada Peminjaman Id 1

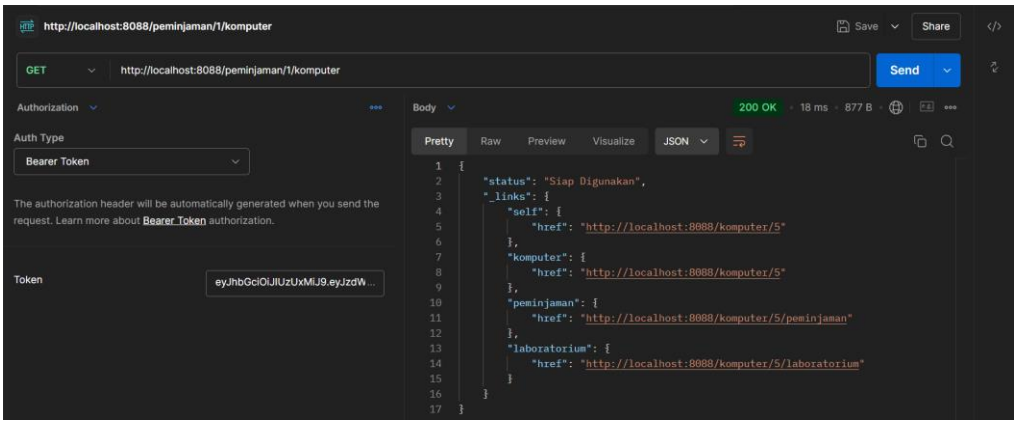


Disini kita bisa melihat detail user yang melalukan peminjaman pada Peminjaman ID 1. Pada gambar terlihat bahwa detailnya adalah sebagai berikut:

- ✓ Nama: Agape Bagus Rega Anggara
- ✓ Kelas: 3SI2
- ✓ Email: bagusrega4@gmail.com
- ✓ Password: *terenkripsi*
- ✓ Role: ADMIN

Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

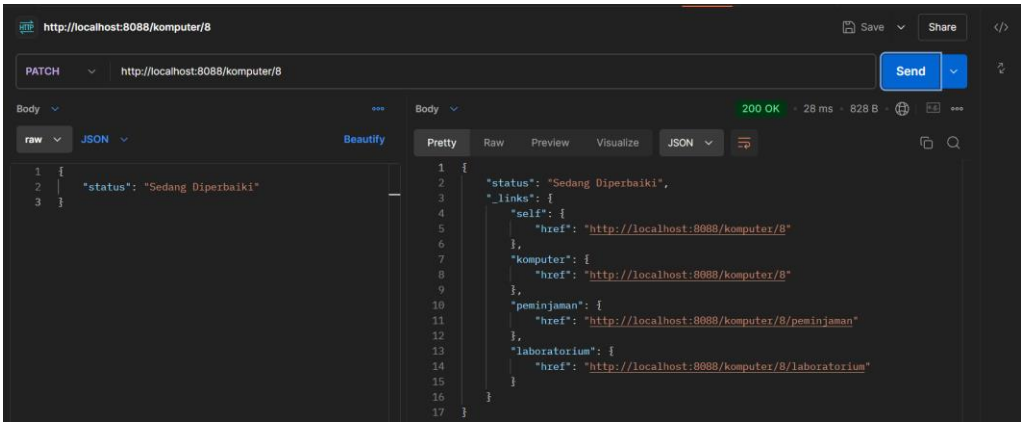
- Detail Komputer pada Peminjaman Id 1



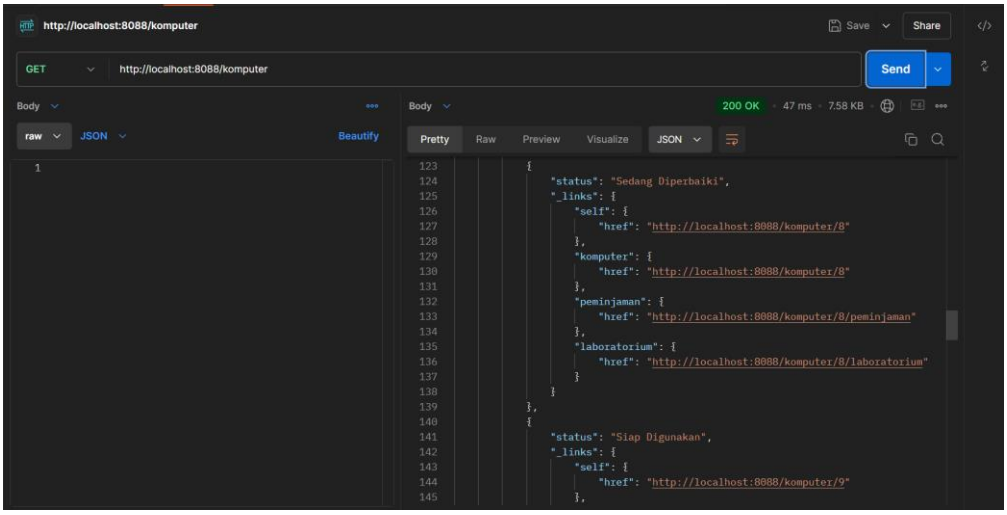
Disini kita bisa melihat detail komputer yang dipinjam pada Peminjaman ID 1. Pada gambar terlihat bahwa komputer yang dipinjam adalah komputer dengan

ID 5, dan sudah dikembalikan (karena statusnya “Siap Digunakan”). Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

- Pengubahan Status

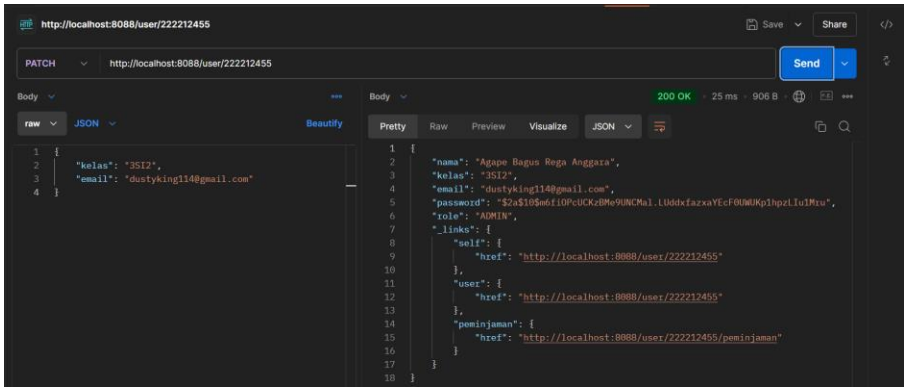


Disini kita sebagai admin dapat mengganti status dari komputer yang sedang bermasalah atau sedang diperbaiki. Semisal kita akan mengubah status komputer yang memiliki ID 8 menjadi “Sedang Diperbaiki”. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.



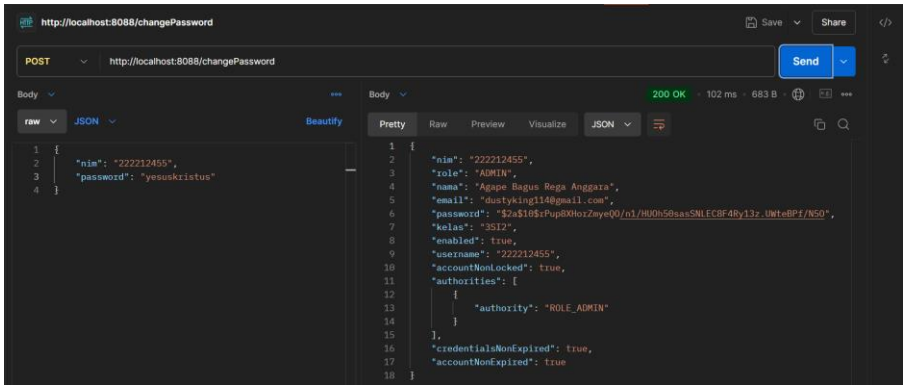
Maka ketika kita melihat semua komputer yang ada, maka komputer dengan ID 8 akan memiliki status “Sedang Diperbaiki”. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

- Penggantian Profil & Password
  - ➢ Mengedit Profil

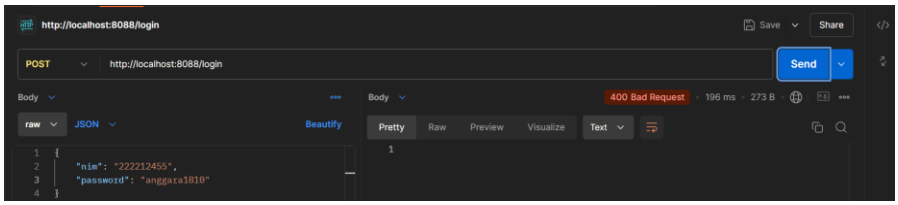


Dengan cara diatas, kita dapat mengedit profil user kita sendiri. Perlu diingat, bahwa kita memerlukan token untuk mengesekusinya.

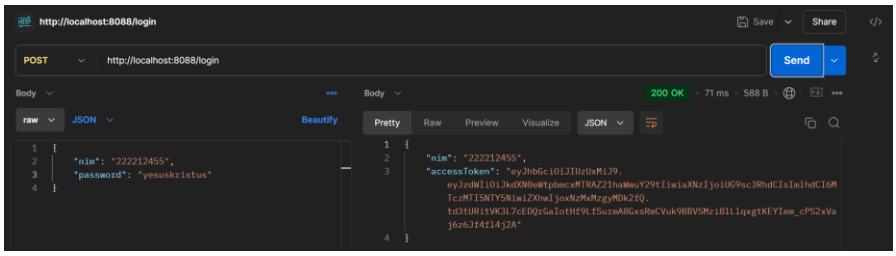
- Mengganti Password



Diatas, kita telah mengganti password akun kita, sebagai bukti adalah gambar dibawah ini:

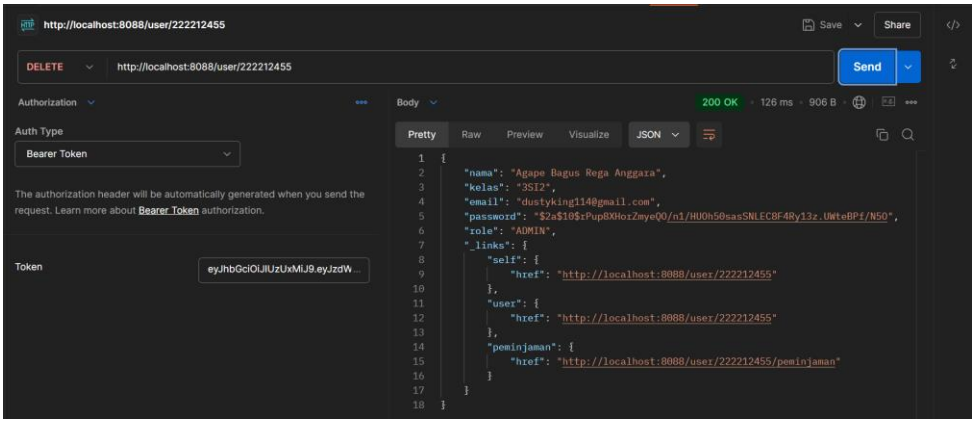


Gagal login menggunakan password lama



Berhasil login dengan password baru. Perlu diingat, bahwa kita memerlukan token untuk mengesekusi semua program diatas.

- Penghapusan Akun



Dapat dilihat bahwa kita telah berhasil menghapus akun milik kita. Perlu diingat, bahwa kita memerlukan token untuk mengesekusi semua program diatas.

E. PENUTUP

Sebagai kesimpulan, pengembangan layanan web servis peminjaman laboratorium komputer di Polstat STIS merupakan langkah strategis dalam meningkatkan efisiensi pengelolaan fasilitas kampus, khususnya untuk mendukung aktivitas akademik yang memerlukan akses laboratorium komputer. Dengan adanya sistem ini, proses reservasi menjadi lebih terstruktur, penggunaan laboratorium dapat dioptimalkan, dan data peminjaman tersimpan secara rapi serta mudah diakses. Diharapkan, layanan web servis ini dapat terus disempurnakan seiring dengan kebutuhan kampus dan perkembangan teknologi.