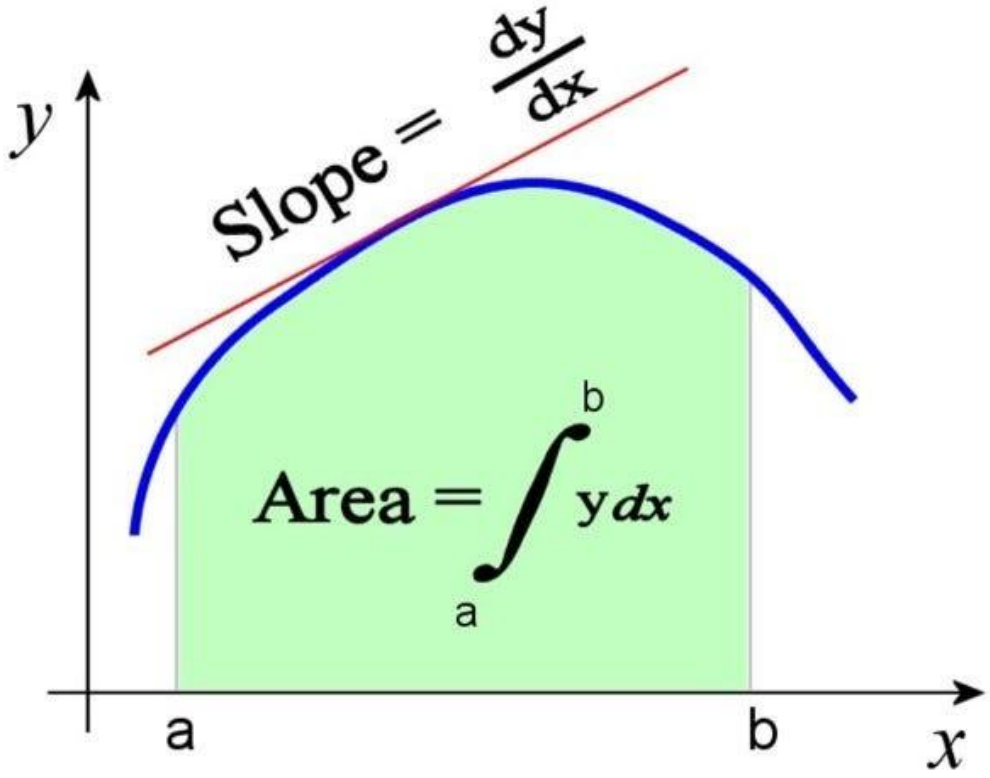
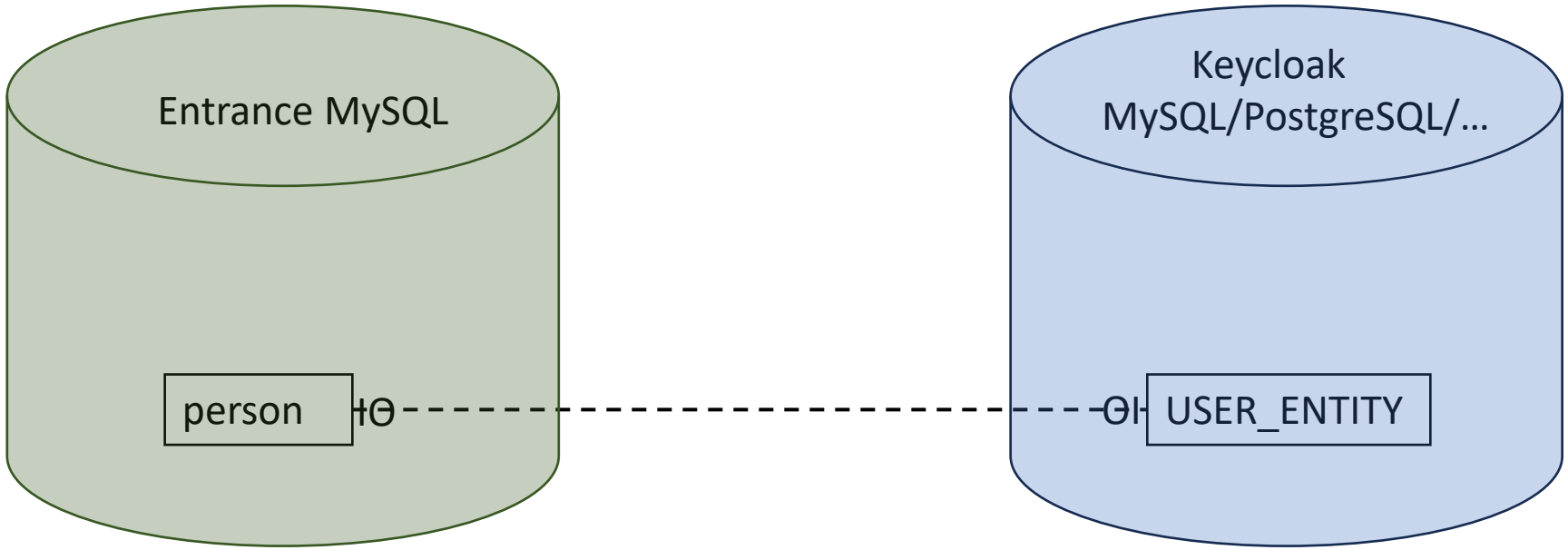
Integrácia užívateľov

Projekt 1



Manažment užívateľov

- Keycloak slúži ako centrálna databáza užívateľov (user)
  - ... pre aplikácie (klientov) v organizácii (ríši)
- Aplikácie majú svoje vlastné databázy s tabuľkou používateľov (napr. tabuľka person)



Manažment užívateľov

- Potrebujeme párovanie medzi KC **user**-mi a Entrance **person**-ami
- Teraz admin môže vytvárať person v Entrance a následne ručne v vytvoriť user-a v KC s tým istým emailom, s rolou normal_user a nejakým heslom.
- Potrebujeme vhodný „cudzí kľúč“ medzi entitami person a user
  - ID nemôžeme použiť, lebo Entrance aj Keycloak si ID generujú samostatne
  - Použime atribút **email**
    - V Entrance je email unikátny pre person-u
    - V KC máme email unikátny pre user-a **(POZOR**, musí to byť zapnuté na úrovni ríše)

Integrujme vytváranie užívateľov v React-e

- Pomocná funkcia na zistenie, či prihlásený používateľ je admin

```
export function userIsAdmin(user: User | null | undefined): boolean {
    if (!user) {
        return false
    }

    const access_token = jwtDecode<any>(user.access_token)
    const realmAccess = access_token["realm_access"]
    if (!realmAccess) {
        return false
    }
    const roles = realmAccess.roles as string[]
    if (!roles) {
        return false
    }

    return roles.includes("admin")
}
```

Vytvárať používateľov môže iba admin

typescript/src/persons/personList/PersonList.tsx

```
...
function PersonList() {
   const auth = useAuth();
   const isAdmin = userIsAdmin(auth.user)
    ...
   return (
      <>
         <TableContainer component={Paper}>
         ...
         </TableContainer>
         {isAdmin ? <IconButton color='primary' size="large"
                        onClick={() => navigate('/add-person')}><AddIcon/></IconButton> : null}
      </>
   );
   ...
}
```

Vytváranie používateľa v React-e

- Heslo nového používateľa musíme zadať už v našej appke

typescript/src/persons/PersonAdd.tsx

```
export function PersonAdd() {
  ...
  return (
    <Stack spacing={2}>
      ...
      <TextField
        fullWidth
        margin="normal"
        label="Password"
        name="password"
        value={person.password}
        type="password"
        onChange={(e) => {
          setPerson((currPerson) => ({...currPerson, password: e.target.value} as Person));
        }}
      />
      ...
    </Stack>
  );
}
```

typescript/src/persons/person.tsx

```
export type Person = {
  id?: number;
  name: string;
  email: string;
  active: boolean;
  gender?: Gender;
  cardReaders: CardReader[];

  password?: string;
}
```

Integrujme vytváranie užívateľov v Spring Boot-e

- React pri vytváraní entity person posiela do Javy aj atribút password
- Ale v Spring Boot-e trieda Person nemá inšt. premennú password
  - Žiaden problém, veď ju pridajme. Či?
  - Veľký problém, lebo password patrí do KC a nie do našej DB (trieda Person = MySQL tabuľka person)

.../controllers/PersonController.java

```
@PostMapping("/persons")
@ResponseStatus(HttpStatus.CREATED)
public Person create(..., @RequestBody Person person) {
  ...
}
```

.../entities/Person.java

```
@Data
@Entity
@Table(name = "person")
public class Person {
    ....
    private String password;
    ....
}
```


DFREIGHT
WEB GUI
REST API
Data Transfer Object

Data Transfer Object

- Entita person z perzistentnej vrstvy (Person.java + MySQL) vyzerá ináč ako potrebujeme na strane prezentačnej vrstvy (PersonController.java + React)
- Navyše potrebujeme entitu Person aj s heslom nejako dostať do Keycloaku (t.j. entita USER_ENTITY v úplne inej DB)
- Uvítajme vzor DTO
- PersonDTO bude obsahovať atribúty z entity Person + heslo

Data Transfer Object

```
@Data
public class PersonDto {
   ...
   private String password;

   public PersonDto() {}

   public PersonDto(Person person) {   ...   }

   public Person toEntity() {  ...   }

   public enum GenderDto {  ...  }
}
```

Skopírujeme všetky inštančné premenné z Person.java

Pridáme potrebné nové inšt. premenné

Prázdny konštruktor pre JSON mapovanie

Konvertovanie z entity->dto a naopak

Pre „úplnosť“ by sme mali aj urobiť aj GenderDto a spraviť aj ChipReaderDto

Integrujme vytváranie užívateľov v Spring Boot-e (pokus 2)

.../controllers/PersonController.java

```
@PostMapping("/persons")
@ResponseStatus(HttpStatus.CREATED)
public PersonDto create(EntranceUser entranceUser, @RequestBody PersonDto personDto) {
    if (!entranceUser.isAdmin()) {
        throw new ResponseStatusException(HttpStatus.FORBIDDEN, "Only admins can create persons");
    }

    if (personDto.getPassword() == null || personDto.getPassword().isBlank()) {
        return new PersonDto(personRepository.save(personDto.toEntity()));
    }

    ...
}
```

Iba admin môže vytvárať nových používateľov

Ak nenastavujeme heslo, tak iba uložíme do DB. T.j. nový užívateľ sa nebude môcť automaticky prihlásiť. (súčasný stav)

V opačnom prípade vytvoríme užívateľa aj v KC s heslom. T.j. nový užívateľ sa bude vedieť prihlásiť ako normal_user.

Keycloak Admin Client

- KC má vlastné administračné REST API pre programatické manažovanie užívateľov
- Zabalené v knižniciach pre Javu a JavaScript (+ neoficiálne pre Python)
  - V iných jazykoch používať surový HTTP klient

pom.xml

```
<dependency>
   <groupId>org.keycloak</groupId>
   <artifactId>keycloak-admin-client</artifactId>
   <version>24.0.1</version>
</dependency>
```

POZOR: Verzia musí byť identická s KC serverom (viď Docker obraz pre KC)

Nakonfigurujme KC klienta

- Potrebujeme v ríši zaregistrovať nového klienta pre našu Spring Boot appku

init_keycloak.json

```
{
  "realm": "entrance-be",
  ...
  "clients": [
    {
      "clientId": "entrance-be",
      "secret": "CHANGE_ME",
      "enabled": true,
      "directAccessGrantsEnabled": true
    }
  ]
}
```

Nakonfigurujme KC klienta

- A taktiež aj nového používateľa so správnou rolou

init_keycloak.json

```
...
{
  "username": "entrance-realm-admin",
  "email": "entrance-realm-admin@entrance.com",
  "firstName": "entrance-realm-admin",
  "lastName": "entrance-realm-admin",
  "enabled": true,
  "emailVerified": true,
  "credentials": [
    {
      "type": "password",
      "value": "CHANGE_ME",
      "temporary": false
    }
  ],
  "clientRoles": {
    "realm-management": ["realm-admin"]
  }
}
...
```

Nakonfigurujme KC clienta

- Potrebujeme nakonfigurovať parametre pre KC admin knižnicu

src/main/resources/application.properties

```
...
app.keycloak.admin.url=${KEYCLOAK_URL:http://localhost:9080}
app.keycloak.admin.client.id=${KEYCLOAK_ADMIN_CLIENT_ID:entrance-be}
app.keycloak.admin.client.secret=${KEYCLOAK_ADMIN_CLIENT_SECRET:CHANGE_ME}
app.keycloak.admin.username=${KEYCLOAK_ADMIN_USERNAME:entrance-realm-admin}
app.keycloak.admin.password=${KEYCLOAK_ADMIN_PASSWORD:CHANGE_ME}
```

Vytvorme KC clienta

- Továreň pre KC klienta

src/main/java/.../configs/Config.java

```
@Bean
public Keycloak keycloak(@Value("${app.keycloak.admin.url}") String authServerUrl,
                         @Value("${app.keycloak.realm}") String realm,
                         @Value("${app.keycloak.admin.client.id}") String clientId,
                         @Value("${app.keycloak.admin.client.secret}") String clientSecret,
                         @Value("${app.keycloak.admin.username}") String username,
                         @Value("${app.keycloak.admin.password}") String password)  {
   return KeycloakBuilder
          .builder()
          .serverUrl(authServerUrl)
          .realm(realm)
          .clientId(clientId)
          .clientSecret(Objects.equals(clientSecret, "") ? null : clientSecret)
          .username(username)
          .password(password)
          .grantType(OAuth2Constants.PASSWORD)
          .build();
}
```

Musíme do DTO dodať aj mapovanie pre KC user-a

.../PersonDto.java

```
public UserRepresentation toUserRepresentation() {
    UserRepresentation userRepresentation = new UserRepresentation();
    userRepresentation.setUsername(email);
    userRepresentation.setEmail(email);

    var splitName = name.split(" ");
    if (splitName.length == 0) {
        userRepresentation.setFirstName("");
        userRepresentation.setLastName("");
    } else {
        userRepresentation.setFirstName(splitName[0]);
        userRepresentation.setLastName(splitName[splitName.length - 1]);
    }

    userRepresentation.setEnabled(active);
    // TODO: remove once Keycloak email integration is implemented
    userRepresentation.setEmailVerified(true);

    return userRepresentation;
}
```

Musíme do DTO dodať aj mapovanie pre user-password

.../PersonDto.java

```
public CredentialRepresentation toCredentialRepresentation(boolean tempPass) {
    var credentials = new CredentialRepresentation();
    credentials.setType(CredentialRepresentation.PASSWORD);
    credentials.setValue(password);
    credentials.setTemporary(tempPass);
    return credentials;
}
```

Heslá užívateľov sa v KC musia ukladať samostatne v objekte typu **CredentialRepresentation**.
Aj keď UserRepresentation z predchádzajúceho slajdu má setter pre atribút typu CredentialRepresentation, tak to nebude fungovať a heslo musíme uložiť v samostatnom API volaní. Prečo? Preto.

Vytvorenie user-a v KC zo Spring Boot-u

.../controllers/PersonController.java, metóda **create**

```
...
String keycloakId;
try (var response = keycloak.realm("entrance").users().create(personDto.toUserRepresentation())) {
    if (response.getStatus() != 201) {
        throw new ResponseStatusException(HttpStatus.INTERNAL_SERVER_ERROR, "Failed to create user");
    }
    keycloakId = CreatedResponseUtil.getCreatedId(response);
}
...
```

Vytvorenie hesla v KC zo Spring Boot-u

.../controllers/PersonController.java, metóda **create**

Podľa interného KC ID user-a si ho vytiahneme

```
var keycloakUser = keycloak.realm("entrance").users().get(keycloakId);
```

Nastavíme mu heslo z personDto

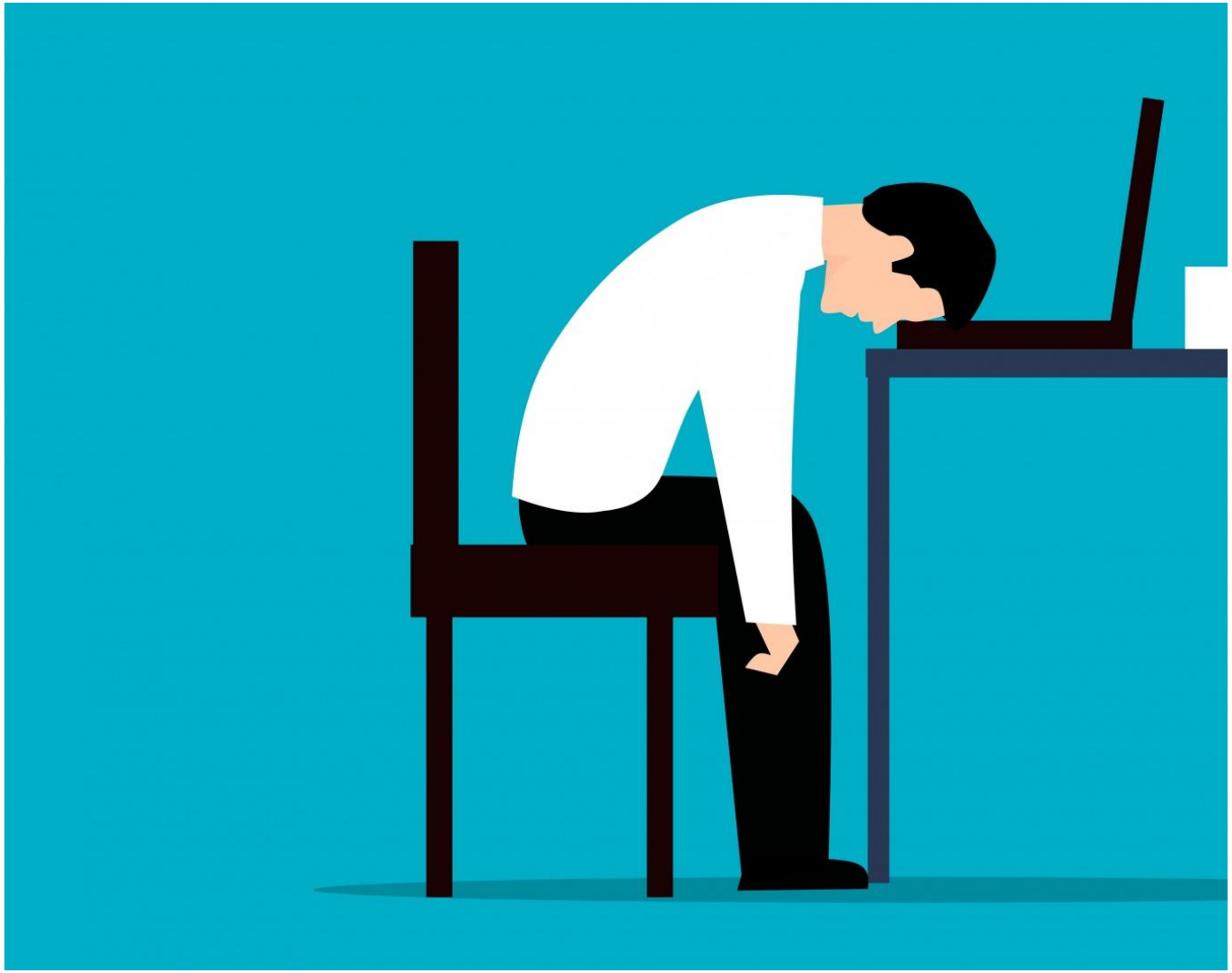
```
keycloakUser.resetPassword(personDto.toCredentialRepresentation(true));
```

Nastavíme mu rolu

```
var role = keycloak.realm("entrance").roles().get("normal_user").toRepresentation();
keycloakUser.roles().realmLevel().add(List.of(role));


return new PersonDto(personRepository.save(personDto.toEntity()));
```

Treba ešte poriešiť update a delete



Ďakujem za pozornosť

Táto prednáška by zabila aj Chucka Norrisa