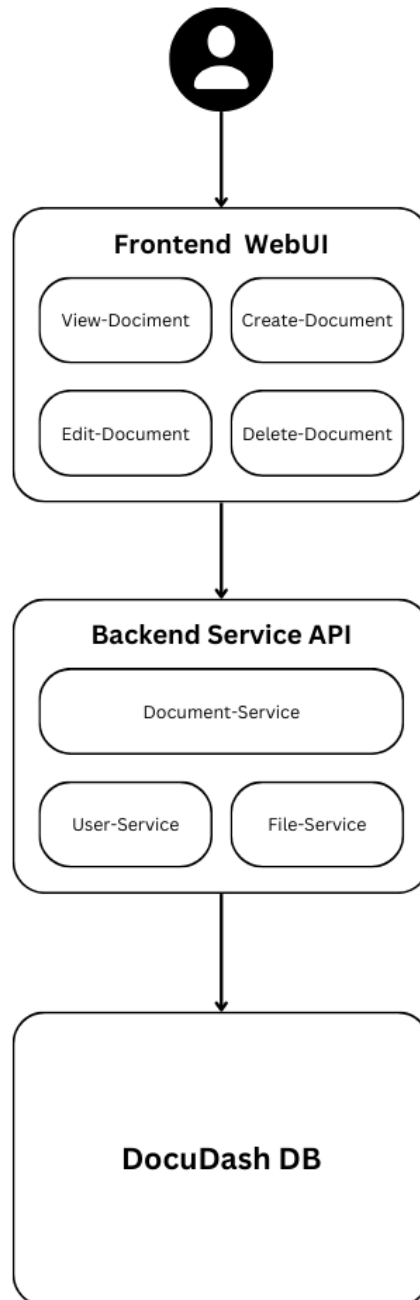


## Technical Report

### 1. Architecture Pattern ใน Project

รูปแบบ : Layered Architecture



**Presentation Layer**

ใช้สำหรับคอยรับและติดต่อกับ Client Request เพื่อส่งต่อไป layer ต่อไปโดยในแต่ละ Controller จะมีหน้าที่ในการแบ่งทางตาม service ที่เราแบ่งไปประมวลผลต่อใน Business Layer และเมื่อได้ผลลัพธ์ก็กลับมาจึงส่งกลับไปหา client เดิม

**Business Layer**

ใช้สำหรับเป็นส่วนของการประมวลผลข้อมูลที่ได้จาก Request ที่ Client ส่งมา โดยสามารถใส่ function ต่างๆ เพื่อทำอะไรบางอย่างกับข้อมูล อย่างเช่น หาจำนวนของข้อมูล, filter ข้อมูล , คำนวณหาค่าเฉพาะของข้อมูลชุดนั้น และคอยเป็นส่วนเชื่อมต่อกับส่วนของ Persistence Layer เพื่อส่งคำสั่งในการเก็บหรือดึงข้อมูลมาใช้งาน

**Persistence Layer**

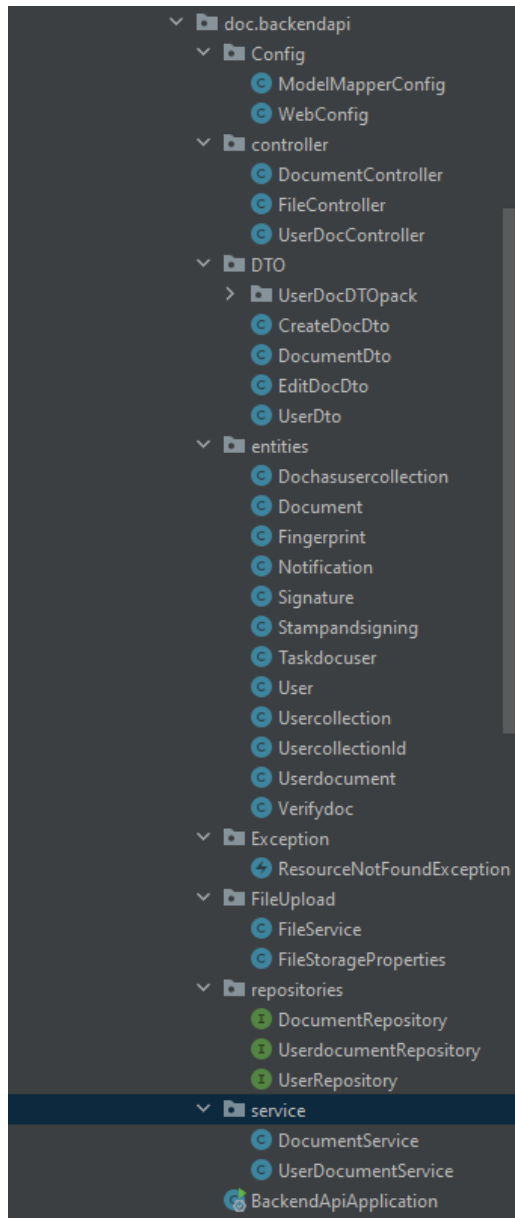
ใช้สำหรับเป็นส่วนที่ติดต่อกับ Database โดยใช้เพื่อดึงข้อมูลหรือบันทึกลงไปโดยคำสั่งที่จะได้รับมาจะได้รับจาก Business Layer ที่จะคอยบอกว่าจะใช้ข้อมูลข้อมูลตัวไหน หรือเก็บข้อมูลตัวนี้ที่ไหน

**Database Layer**

ใช้สำหรับเป็นส่วนที่จะเป็นที่เก็บข้อมูลหลักของ Application ของระบบโดยรับข้อมูลทำหน้าที่เป็นเพียงพื้นที่จัดการกับข้อมูลของระบบ ให้เป็นไปตามรูปแบบของ Database ในแต่ละเจ้าหรือตามการออกแบบของ Application นั้นๆ

## 2. รูปโครงสร้าง และโค้ดส่วน Backend ที่แสดง Implementation ตาม Software Architecture ที่ได้ออกแบบไว้

### Project folder



## DocumentController เป็นส่วน Presentation Layer

```
DocumentController.java
1  package doc.backendapi.controller;
2
3  import ...
4
19
20  @RestController
21  @RequestMapping("/api/doc")
22  public class DocumentController {
23
24      @Autowired
25      private DocumentService documentService;
26      @Autowired
27      private FileService fileService;
28
29      @GetMapping("/")
30      public List<DocumentDto> getDocument() { return documentService.getAllDocument(); }
31
32
33
34      @GetMapping("/newdocid")
35      public int getNewDocId() { return documentService.getNewDocId(); }
36
37
38
39      @GetMapping("/user/{id}")
40      public List<UserdocumentDto> getDocumentByUserId(@PathVariable int id) {
41          return documentService.getDocumentByUserId(id);
42      }
43
44      @GetMapping("/{id}")
45      public DocumentDto getDocumentById(@PathVariable int id) { return documentService.getDocumentById(id); }
46
47
48
49      @PostMapping("/")
50      public CreateDocDto saveDocument(MultipartHttpServletRequest request) throws IOException {
51          MultipartFile file = request.getFile("file");
52          String createDocDtoJson = request.getParameter("data");
53          ObjectMapper objectMapper = new ObjectMapper();
54          CreateDocDto createDocDto = objectMapper.readValue(createDocDtoJson, CreateDocDto.class);
55          fileService.store(file);
56          return documentService.saveDocument(createDocDto, file.getOriginalFilename());
57      }
58
```

## FileController เป็นส่วน Presentation Layer

```
FileController.java
1  package doc.backendapi.controller;
2
3  import ...
16
17  @RestController
18  @RequestMapping("/api/files")
19  public class FileController {
20      3 usages
21      private final FileService fileService;
22
23      @Autowired
24      public FileController(FileService fileService) { this.fileService = fileService; }
25
26
27      @GetMapping("/{filename}")
28      @ResponseBody
29      public ResponseEntity<Resource> serveImage(@PathVariable String filename) {
30          Resource file = fileService.loadFileAsResource(filename);
31          String mimeType;
32          try {
33              mimeType = Files.probeContentType(Paths.get(file.getURI()));
34          } catch (IOException e) {
35              throw new ResponseStatusException(HttpStatus.UNPROCESSABLE_ENTITY, "Error occurred while determining fil
36          }
37          return ResponseEntity.ok().contentType(MediaType.parseMediaType(mimeType)).body(file);
38      }
39
40      @PostMapping("")
41      public String fileUpload(@RequestParam("file") MultipartFile file) {
42          String filePath = fileService.store(file);
43          return "You successfully uploaded " + file.getOriginalFilename() + "! The file path is " + filePath;
44      }
45  }
46
```

## UserDocController เป็นส่วน Presentation Layer

```
UserDocController.java
1  package doc.backendapi.controller;
2
3  import ...
13
14  @RestController
15  @RequestMapping("/api/ud")
16  public class UserDocController {
17
18      @Autowired
19      UserDocumentService userDocumentService;
20
21      @PostMapping("/")
22      public List<CreateUserDocDTO> saveUserDocs(@RequestBody List<CreateUserDocDTO> userDocJsons) {
23          List<CreateUserDocDTO> savedUserDocs = new ArrayList<>();
24
25          for (CreateUserDocDTO userDocJson : userDocJsons) {
26              CreateUserDocDTO savedUserDoc = userDocumentService.save(userDocJson);
27              savedUserDocs.add(savedUserDoc);
28          }
29
30          return savedUserDocs;
31      }
32  }
```

## FileService เป็นส่วน Business Layer

```

1 package doc.backendapi.FileUpload;
2
3 import ...
4
5 6 usages
6 @Service
7 public class FileService {
8     4 usages
9     private final Path fileStorageLocation;
10
11     2 usages
12     private static final Logger logger = LoggerFactory.getLogger(FileService.class);
13
14
15     @Autowired
16     @
17     public FileService(FileStorageProperties fileStorageProperties) {
18         this.fileStorageLocation = Paths.get(fileStorageProperties.getUploadDir()).toAbsolutePath().normalize();
19         try {
20             Files.createDirectories(this.fileStorageLocation);
21         } catch (Exception ex) {
22             throw new RuntimeException("Unable to create the directory for storing uploaded files.", ex);
23         }
24     }
25
26
27     3 usages
28     @
29     public String store(MultipartFile file) {
30         logger.info("Storing file with size: " + file.getSize());
31         // Normalize file name
32         String fileName = StringUtils.cleanPath(Objects.requireNonNull(file.getOriginalFilename()));
33         try {
34             // Check if the file's name contains invalid characters
35             if (fileName.contains("..")) {
36                 throw new RuntimeException("Invalid file name: " + fileName + ". Please ensure the file name does not contain '..'");
37             }
38             // Copy file to the target location (Replacing existing file with the same name)
39             Path targetLocation = this.fileStorageLocation.resolve(fileName);
40             Files.copy(file.getInputStream(), targetLocation, StandardCopyOption.REPLACE_EXISTING);
41             logger.info("Stored file at location: " + targetLocation);
42         }
43     }
44 }

```

## FileStorageProperties เป็นส่วน Business Layer

```

1 package doc.backendapi.FileUpload;
2
3 import ...
4
5 3 usages
6 @ConfigurationProperties(prefix = "file")
7 @Getter
8 @Setter
9 public class FileStorageProperties {
10     private String uploadDir;
11 }
12
13

```

## DocumentService เป็นส่วนของ Business Layer

```
DocumentService.java
1  package doc.backendapi.service;
2
3  import ...
27
28  @Service
29  @RequiredArgsConstructor
30  public class DocumentService {
31
32      @Autowired
33      private final DocumentRepository documentRepository;
34
35      @Autowired
36      private final UserdocumentRepository userdocumentRepository;
37
38      @Autowired
39      private ModelMapper modelMapper;
40
41      1 usage
42      public List<DocumentDto> getAllDocument() {
43          return documentRepository.findAll().stream()
44              .sorted(Comparator.comparing(Document::getDateUpdate).reversed())
45              .map((element) -> modelMapper.map(element, DocumentDto.class))
46              .collect(Collectors.toList());
47      }
48
49      1 usage
50      public List<UserdocumentDto> getDocumentByUserId(int id) {
51          Map<Integer, Userdocument> uniqueDocuments = userdocumentRepository.findByUsers_UserID(id).stream()
52              .collect(Collectors.toMap(
53                  ud -> ud.getDocumentsDocumentid1().getId(), // Key is documentsDocumentid1.id
54                  ud -> ud, // Value is the Userdocument entity
55                  (ud1, ud2) -> ud1 // If there are duplicates, keep the first one
56              ));
57          return new ArrayList<>((uniqueDocuments.values()).stream()
58              .map(ud -> modelMapper.map(ud, UserdocumentDto.class))
59              .collect(Collectors.toList()));
60      }
61  }
```



## UserDocumentService เป็นส่วนของ Business Layer

```

1 package doc.backendapi.service;
2
3 import ...
4
5 2 usages
6
7 @Service
8 @RequiredArgsConstructor
9 public class UserDocumentService {
10
11     @Autowired
12     private UserdocumentRepository userdocumentRepository;
13
14     @Autowired
15     private DocumentRepository documentRepository;
16
17     @Autowired
18     private UserRepository userRepository;
19
20     @Autowired
21     private ModelMapper modelMapper;
22
23     no usages
24     public List<UserdocumentDto> getAllUserDocument() {
25         return userdocumentRepository.findAll().stream().map((element) ->
26             modelMapper.map(element, UserdocumentDto.class)).collect(Collectors.toList());
27     }
28
29     1 usage
30     public CreateUserDocDTO save(CreateUserDocDTO data) {
31         Document document = documentRepository.findById(data.getDocumentIdId())
32             .orElseThrow(() -> new RuntimeException(HttpStatus.NOT_FOUND, "Document not found"));
33
34         User user = userRepository.findById(data.getUserUserIdId())
35             .orElseThrow(() -> new RuntimeException(HttpStatus.NOT_FOUND, "User not found"));
36
37         if (document == null || user == null) {
38             throw new RuntimeException(HttpStatus.NOT_FOUND, "Document or User not found");
39         }
40     }
41
42 }

```

## CreateUserDocDTO เป็นส่วนของ Persistence Layer

```

1 package doc.backendapi.DTO.UserDocDTOPack;
2
3 import ...
4
5 /**
6  * DTO for {@link doc.backendapi.entities.Userdocument}
7  */
8
9 10 usages
10
11 @Data
12 @NoArgsConstructor
13 public class CreateUserDocDTO implements Serializable {
14     @Size(max = 45)
15     String accessLevel;
16     Integer documentsDocumentIdId;
17     Integer usersUserIdId;
18 }
19
20

```

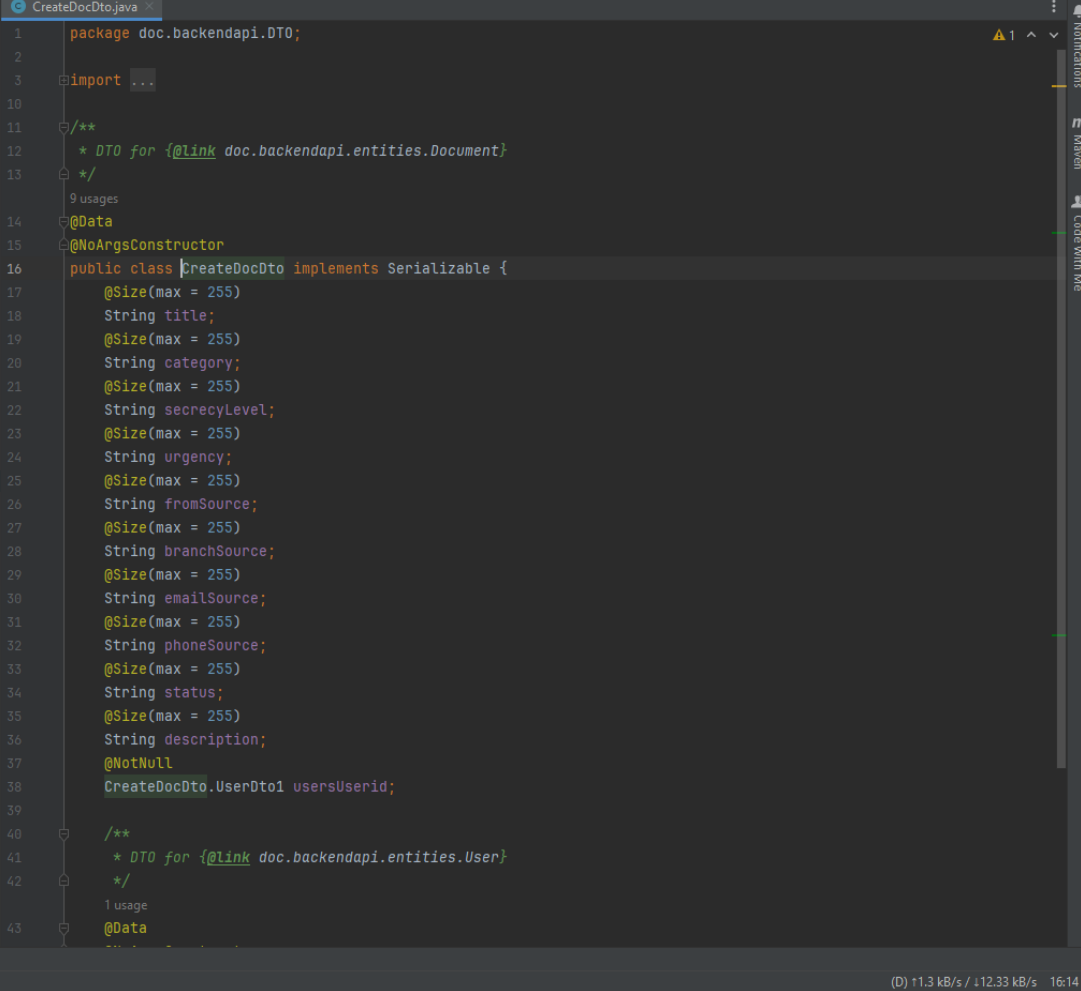
## UserdocumentDto เป็นส่วนของ Persistence Layer

```

1  package doc.backendapi.DTO.UserDocDTOPack;
2
3  import ...
4
10
11  /**
12   * DTO for {@link doc.backendapi.entities.Userdocument}
13   */
14  @Data
15  @NoArgsConstructor
16  public class UserdocumentDto implements Serializable {
17      Integer id;
18      @Size(max = 45)
19      String accessLevel;
20      @NotNull
21      UserdocumentDto.DocumentDto documentsDocumentid1;
22      @NotNull
23      UserdocumentDto.UserDto usersUserId;
24
25      /**
26       * DTO for {@link doc.backendapi.entities.Document}
27       */
28      @Data
29      @NoArgsConstructor
30      public static class DocumentDto implements Serializable {
31          Integer id;
32          @Size(max = 255)
33          String title;
34          @Size(max = 255)
35          String filePath;;
36          Instant dateAdd;
37          Instant dateUpdate;
38          @Size(max = 255)
39          String category;
40          @Size(max = 255)
41          String secrecyLevel;
42          @Size(max = 255)
43          String urgency;
44          ...

```

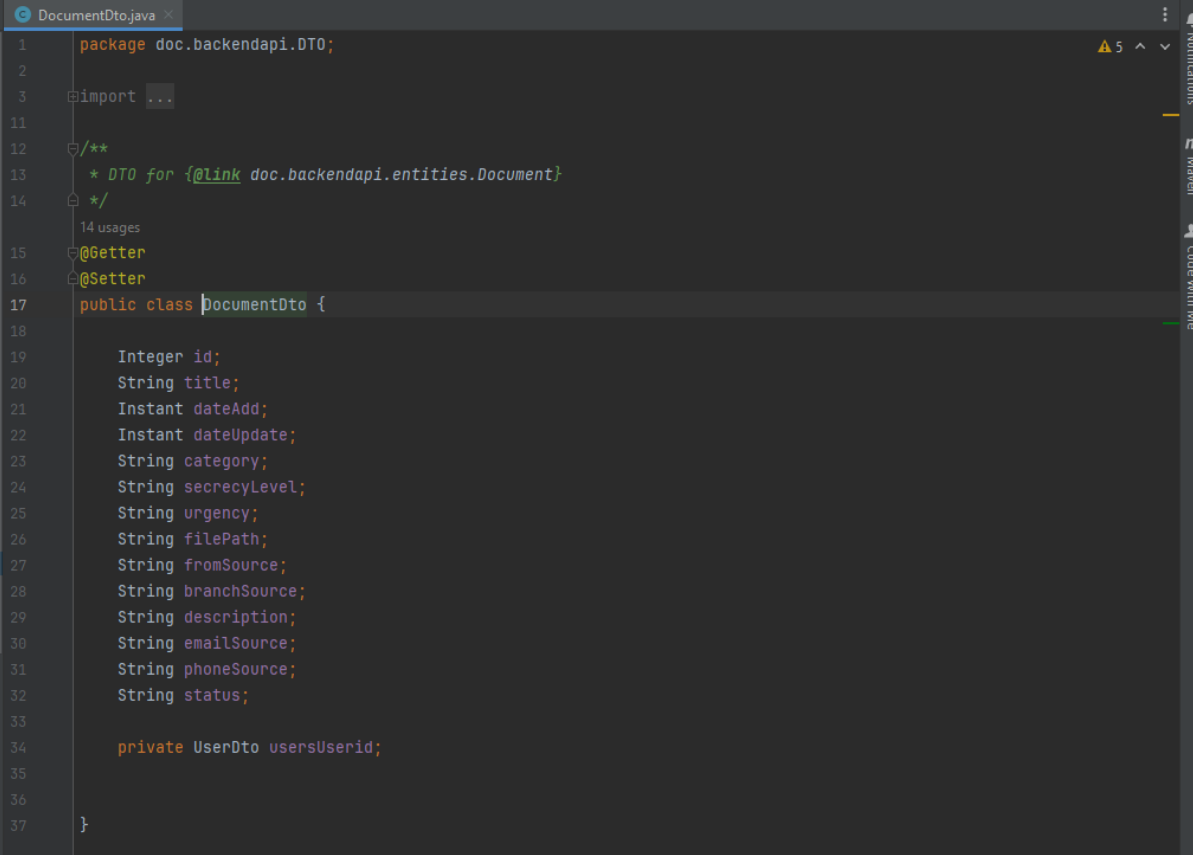
## CreateDocDto เป็นส่วนหนึ่งของ Persistence



```
1 package doc.backendapi.DTO;
2
3 import ...
4
5
6
7
8
9
10
11 /**
12  * DTO for {@link doc.backendapi.entities.Document}
13  */
14 @Data
15 @NoArgsConstructor
16 public class CreateDocDto implements Serializable {
17     @Size(max = 255)
18     String title;
19     @Size(max = 255)
20     String category;
21     @Size(max = 255)
22     String secrecyLevel;
23     @Size(max = 255)
24     String urgency;
25     @Size(max = 255)
26     String fromSource;
27     @Size(max = 255)
28     String branchSource;
29     @Size(max = 255)
30     String emailSource;
31     @Size(max = 255)
32     String phoneSource;
33     @Size(max = 255)
34     String status;
35     @Size(max = 255)
36     String description;
37     @NotNull
38     CreateDocDto.UserDto1 usersUserId;
39
40     /**
41      * DTO for {@link doc.backendapi.entities.User}
42      */
43     @Data
44     ...
45 }
```

(D) 11.3 kB/s / 112.33 kB/s 16:14

## DocumentDto เป็นส่วนหนึ่งของ Persistence



```
1 package doc.backendapi.DTO;
2
3 import ...
4
11
12 /**
13  * DTO for {@link doc.backendapi.entities.Document}
14  */
15
16 14 usages
17 @Getter
18 @Setter
19 public class DocumentDto {
20
21     Integer id;
22     String title;
23     Instant dateAdd;
24     Instant dateUpdate;
25     String category;
26     String secrecyLevel;
27     String urgency;
28     String filePath;
29     String fromSource;
30     String branchSource;
31     String description;
32     String emailSource;
33     String phoneSource;
34     String status;
35
36     private UserDto usersUserId;
37 }
```

## EditDocDto เป็นส่วนหนึ่งของ Persistence

```

1 package doc.backendapi.DTO;
2
3 import ...
4
5 /**
6  * DTO for {@link doc.backendapi.entities.Document}
7  */
8
9 @Data
10 @NoArgsConstructor
11 public class EditDocDto implements Serializable {
12     @Size(max = 255)
13     String title;
14     @Size(max = 255)
15     String filePath;
16     @Size(max = 255)
17     String category;
18     @Size(max = 255)
19     String fromSource;
20     @Size(max = 255)
21     String branchSource;
22     @Size(max = 255)
23     String emailSource;
24     @Size(max = 255)
25     String phoneSource;
26     @Size(max = 255)
27     String description;
28 }

```

## UserDto เป็นส่วนหนึ่งของ Persistence

```

1 package doc.backendapi.DTO;
2
3 import ...
4
5 /**
6  * DTO for {@link User}
7  */
8
9 @Getter
10 @Setter
11 public class UserDto {
12     private Integer id;
13     @Size(max = 255)
14     private String username;
15     @Size(max = 255)
16     private String fullName;
17     @Size(max = 255)
18     private String role;
19     @Size(max = 255)
20     private String email;
21     @Size(max = 255)
22     private String phone;
23     @Size(max = 255)
24     private String branch;
25 }

```

## 3. Repository ของโครงการ

<https://github.com/hunnymc/DocuDash>