Matriz de Insumos

Esquema do diretório principal

- 🗀 **matriz-insumos-v1**
  - 🗀 **exception**
    contém as exceções compartilhadas
  - 🗀 **handle**
    contém os handles do sistema, com interceptadores globais
  - 🗀 **util**
    contém classes específicas para o projeto , conforme especificação da Klabin
  - 🗀 **lib**
    contém as classes bases/lib do sistema
  - 🗀 **crud**
    - 🗀 **lookup**
      tabelas auxiliares/lookup com os campos Id, Nome onde o Id é do tipo auto-increment
    - 🗀 **internal**
      tabelas de controle interno do sistema id fixo, são gerenciandas pelo DBA/Admin do sistema
    - 🗀 **produto**
      tabela de produto
    - 🗀 **matriz**
      tabela de matriz
    - 🗀 **matriz-documento**
      tabela de relacionamento da tabela matriz e documento
    - 🗀 **informacao-adicional**
      tabela da informação adicionai
    - 🗀 **informacao-adicional-matriz**
      tabela de informação adicional relacionada com a matriz
    - 🗀 **especie**
      tabela de espécia

- 🗀 **tipo-risco-potencial**
  tabela de risco potencial
- 🗀 **planta-daninha**
  tabela de planta daninha

Estrutura das pastas dentro de cada 🗀 */crud/<artefato>/*

- design-pattern
  - 🗀 **entrypoint**
    deverá conter os pontos de entrada para api
    - 🗀 **config**
      contém as classes de configuração para o entrypoint, como por exemplo o ControllerAdvisor, que captura as exceptions

```
public class ControllerAdvisor extends ResponseEntityExceptionHandler  {

    @ExceptionHandler(BusinessException.class)
    protected ResponseEntity<Object> handleBusinessException(BusinessException ex, WebR
        return handle(HttpStatus.UNPROCESSABLE_ENTITY, request, Arrays.asList(ex.getMes
    }

    private ResponseEntity<Object> handle(HttpStatus status, WebRequest request, List<S
        var body = new LinkedHashMap<String, Object>();
        body.put("timestamp", LocalDate.now());
        body.put("status", status.value());
        body.put("errors", errors);

        if (request != null) {
            body.put("path", ((ServletWebRequest) request).getRequest().getRequestURI()
        }

        return new ResponseEntity<>(body, status);
    }
}
```

    - 🗀 **rest**
      controller de entrada da aplicação, devem ser configurado o swagger da aplicação

```
@RestController
@RequestMapping("/classe")
public class ClasseRestController {

    private final ClasseUseCase classeUseCase;

    public ClasseRestController(ClasseUseCase classeUseCase) {
        this.classeUseCase = classeUseCase;
    }

    @Operation(summary = "Cadastrar classes para <nome do projeto>")
    @PostMapping(produces = MediaType.APPLICATION_JSON_VALUE)
    @ApiResponses(value =
    {
        @ApiResponse(responseCode = "201",
                     description = "Classe cadastrada com sucesso",
                     content = {@Content(
                                    mediaType = MediaType.APPLICATION_JSON_VALUE,
                                    schema = @Schema(implementation = ClasseResponse.cl
                                ) }),
            @ApiResponse(responseCode = "422", description = "Classe não cadastrada", c
    })
    public ResponseEntity<Object> insert(@RequestBody AtuacaoDTO atuacao) {

        return ResponseEntity.status(HttpStatus.CREATED).body(atuacaoDTO);
    }

}
```

- 📂 **domain**
  contém as regras de negócio da API, é uma camada isolada que não conhece os pontos de entrada, nem a origem dos dados
  - 📂 **cross**
    classe de formatação/uteis, como por exemplo classe de formatação de datas
  - 📂 **dto**
    classe de transporte para comunicação entre EntryPoint e UseCase

```
public class ClasseDTO {

    private Long id;

    private String nome;
}
```

- 🗀 **entity**
  classe de transporte para comunicação entre UseCase e DataProvider

```
public class ClasseEntity {

    private Long id;

    private String nome;
}
```

- 🗀 **exception**
  exceções específicas das regras de negócio

```
public class BusinessException extends RuntimeExceptiony {

    private static final long serialVersionUID = 6993878570229158267L;

    public BusinessException() {
    }

    public BusinessException(String message) {
        super(message);
    }
}
```

- 🗀 **gateway**
  interface para que na infra seja implementada pela classe <artefato>DataProvider e faça a busca de dados, a entrada são do tipo **dto** e retorno como **entity**

```
public interface ClasseGateway {
   ClasseEntity salvar(ClasseDTO dto);
}
```

- 🗀 **mapper**
  Classe de conversão de **request** para **dto**

```
import org.mapstruct.Mapper;

@Mapper(componentModel = "spring")
public interface ClasseDTOMapper{

    ClasseDTOMapper INSTANCE = Mappers.getMapper(ClasseDTOMapper.class);

    ClasseDTO toDTO(ClasseRequest request);
}
```

- 🗀 **usecase**
  contém as interfaces para as implementações das regras de negócio, tem como entrada a classe **request** e de retorno a classe **response**

```
public interface ClasseUseCase{
    ClasseResponse salvar(ClasseRequest request);
}
```

  - 🗀 **request**
    classe de entrada de dados para o usecase

```
public class ClasseRequest {

    private Long id;

    private String nome;
}
```

  - 🗀 **response**
    classe de saída de dados para o usecase

```
public class ClasseResponse {

    private Long id;

    private String nome;
}
```

  - /🗀 **response**
    - /🗀 **mapper**
      Classe de conversão de **entity** para **response**

```
import org.mapstruct.Mapper;

@Mapper(componentModel = "spring")
public interface ClasseResponseMapper{

    ClasseResponseMapper INSTANCE = Mappers.getMapper(ClasseResponseMapper.class);

    ClasseResponse toDTO(ClasseEntity entity);
}
```

- 🗀 **impl**

  Implementação da interface do useCase

```
public interface ClasseUseCaseImpl implements ClasseUseCase {

    private final ClasseGateway classeGateway;

    public ClasseUseCaseImpl(ClasseGateway classeGateway) {
        this.classeGateway = classeGateway;
    }

    public ClasseResponse salvar(ClasseRequest request){
        validar(request);
        var dto = ClasseDTOMapper.INSTANCE.toDTO(request);
        var entity = classeGateway.salvar(dto);
        return ClasseResponseMapper.INSTANCE.toResponse(entity);
    }

    public validar(ClasseRequest request){
        throw new BusinessException("Mensagem de erro caso haja algo de errado");
    }

}
```

- 🗀 **infra**

  local de storage dos dados, neste projeto a base de dados em MySql

  - 🗀 **dataprovider**

    contém as classes de implementação das interface de /gateway

```
public class ClasseDataProvider implements ClasseGateway {

    private final ClasseRepository classeRepository;

    public ClasseDataProvider(ClasseRepository classeRepository){
        this.classeRepository = classeRepository;
    }

@Override
public ClasseEntity salvar(ClasseEntity entity) {
    var jpa = ClasseMapper.INSTANCE.toJpa(entity);
    var responseJPA = repository.save(jpa);
    return ClasseMapper.INSTANCE.toEntity(responseJPA);
}




}
```

- 🗀 **jpa**
  classe de mapeamento para dos dados em jpa
  - 🗀 **model (@entity)**
    classe de mapeamento das tabelas de dados, as classe devem terminar com o sufixo Jpa.java

```
@Entity
@Table(name = "Classe")
public class ClasseJpa implements Serializable {

    private static final long serialVersionUID = -8742454888558739122L;

    @Id
    @Column(name = "Id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "Nome")
    private String nome;

}
```

- 🗁 **repository**
  interface de mapeamento do JPARepository

```
public class ClasseRepository extends JpaRepository<ClasseJpa, Long> {

    boolean existsByNomeIgnoreCase(String nome);

    boolean existsById(Long id);

    Optional<ClasseJpa> findByNomeIgnoreCase(String nome);

    Long findIdByNomeIgnoreCaseAndIdNot(String nome, Long id);

}
```

- 🗁 **mapper**
  classe de mapeamento dos entity e model(jpa)

```
@Mapper(componentModel = "spring")
public interface ClasseMapper{

    ClasseMapper INSTANCE = Mappers.getMapper(ClasseMapper.class);

    ClasseEntity toEntity(ClasseJpa model);
    ClasseJpa toJpa(ClasseEntity entity);
}
```

Atualizações

Item	Data	Autor - Empresa	Descrição
1	28/12/2023	Dannyrooh - Regazzo	Criação do documento com as definições da estrutura de desenvolvimento para o modelo de design pattern