Cargos e Salários

Elton F. de Oliveira

Gabriel Simon

Guilherme Ezequiel

Lucas Grijó

Samuel levi

Senior ProWay

Sprint Review

Senior ProWay

O que foi feito

- Implementação DAO
- Interface CRUD com Generics
- Padronização dos métodos
- Testes unitários
- Arquitetura de packages
- Persistência dos dados com ArrayList e CSV
- Atualização do Diagrama de Classes
- Documentação JavaDoc

Senior ProWay

O que retirado ou adicionado

- Arquitetura anterior (service).
- Testes unitários das classes de service (arquitetura anterior).
+ persistência de dados alternativa (CSV)

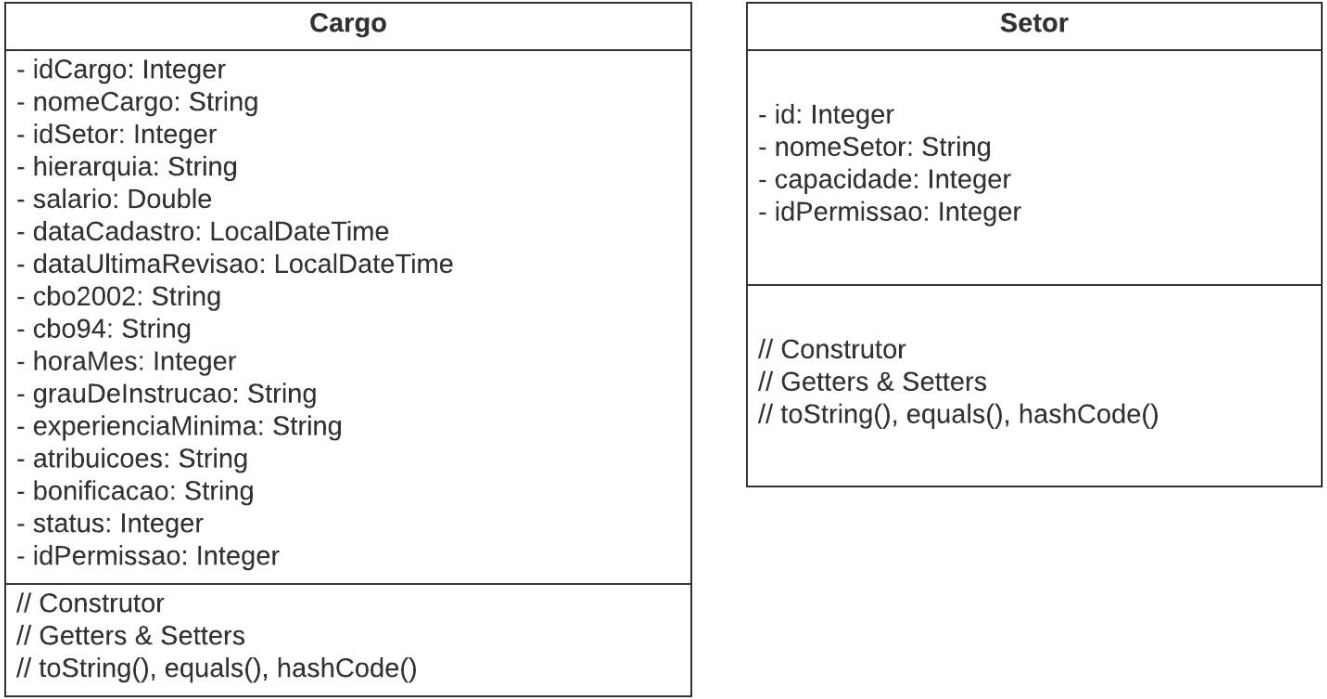
O que não foi feito

- Persistência de dados utilizando CSV para classe Cargo

Senior ProWay

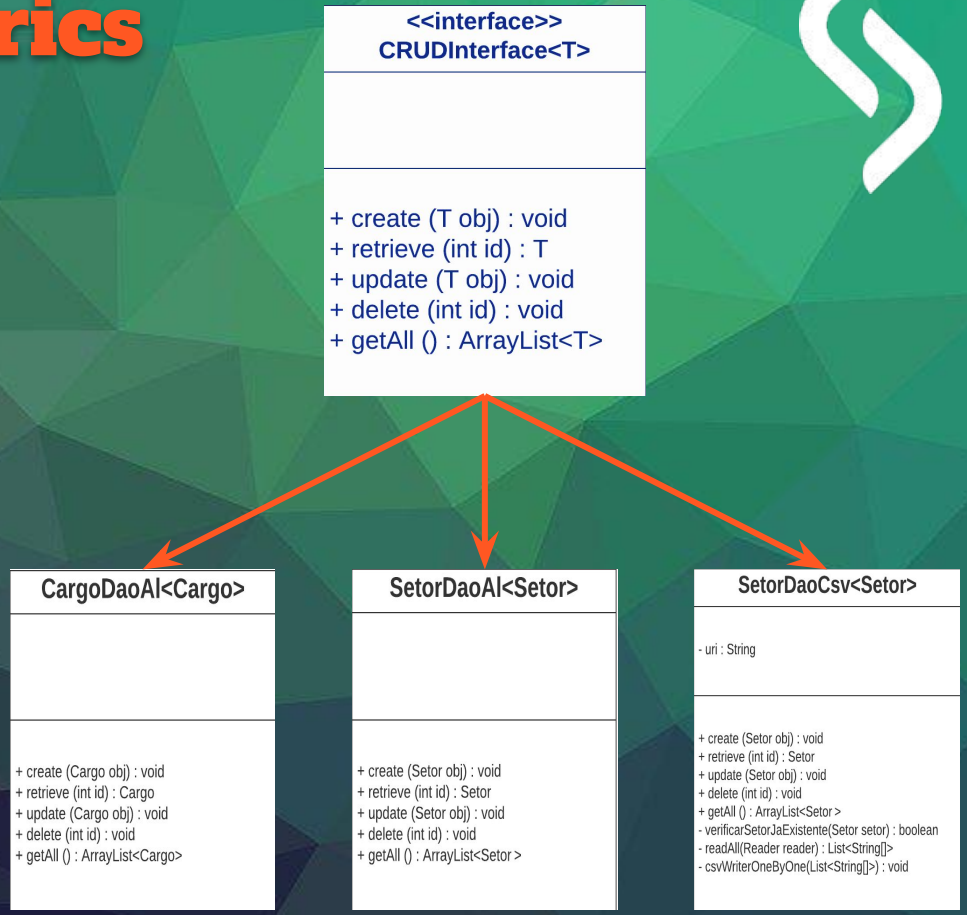
Arquitetura e Código-Fonte

Entidades


Cargo
- idCargo: Integer
- nomeCargo: String
- idSetor: Integer
- hierarquia: String
- salario: Double
- dataCadastro: LocalDateTime
- dataUltimaRevisao: LocalDateTime
- cbo2002: String
- cbo94: String
- horaMes: Integer
- grauDeInstrucao: String
- experienciaMinima: String
- atribuicoes: String
- bonificacao: String
- status: Integer
- idPermissao: Integer
// Construtor
// Getters & Setters
// toString(), equals(), hashCode()
Setor
- id: Integer
- nomeSetor: String
- capacidade: Integer
- idPermissao: Integer
// Construtor
// Getters & Setters
// toString(), equals(), hashCode()

Interface com Generics

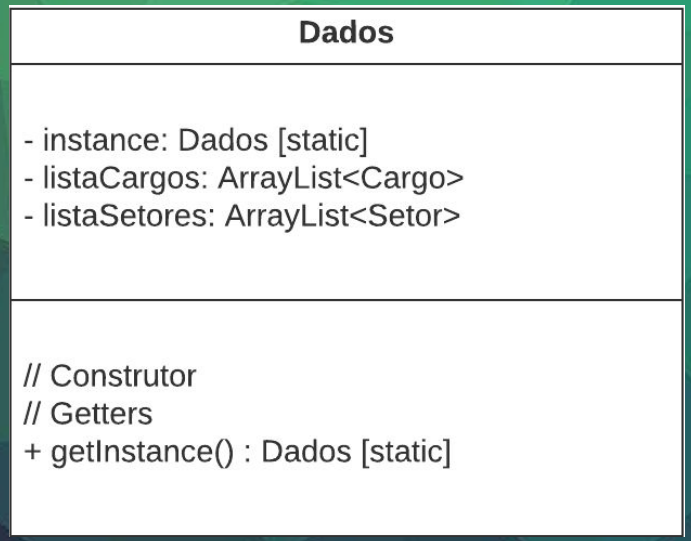
A CRUDInterface<T>, nos permitiu simplificar a criação e manutenção dos métodos de CRUD nas classes de acesso aos dados (DAO), sendo passado apenas o tipo do dado na criação da classe correspondente.



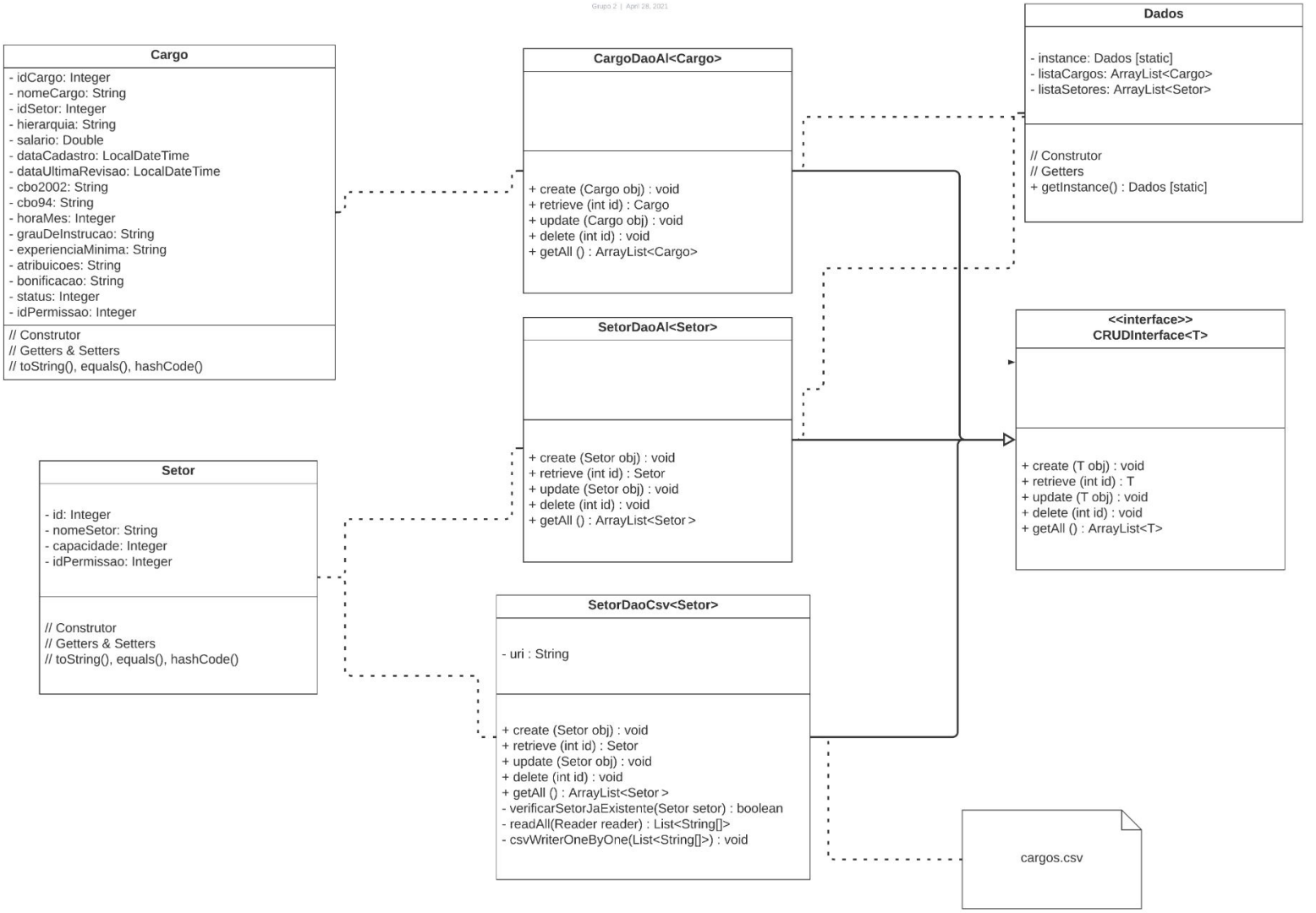
Senior ProWay

Uso do Design Pattern - Singleton

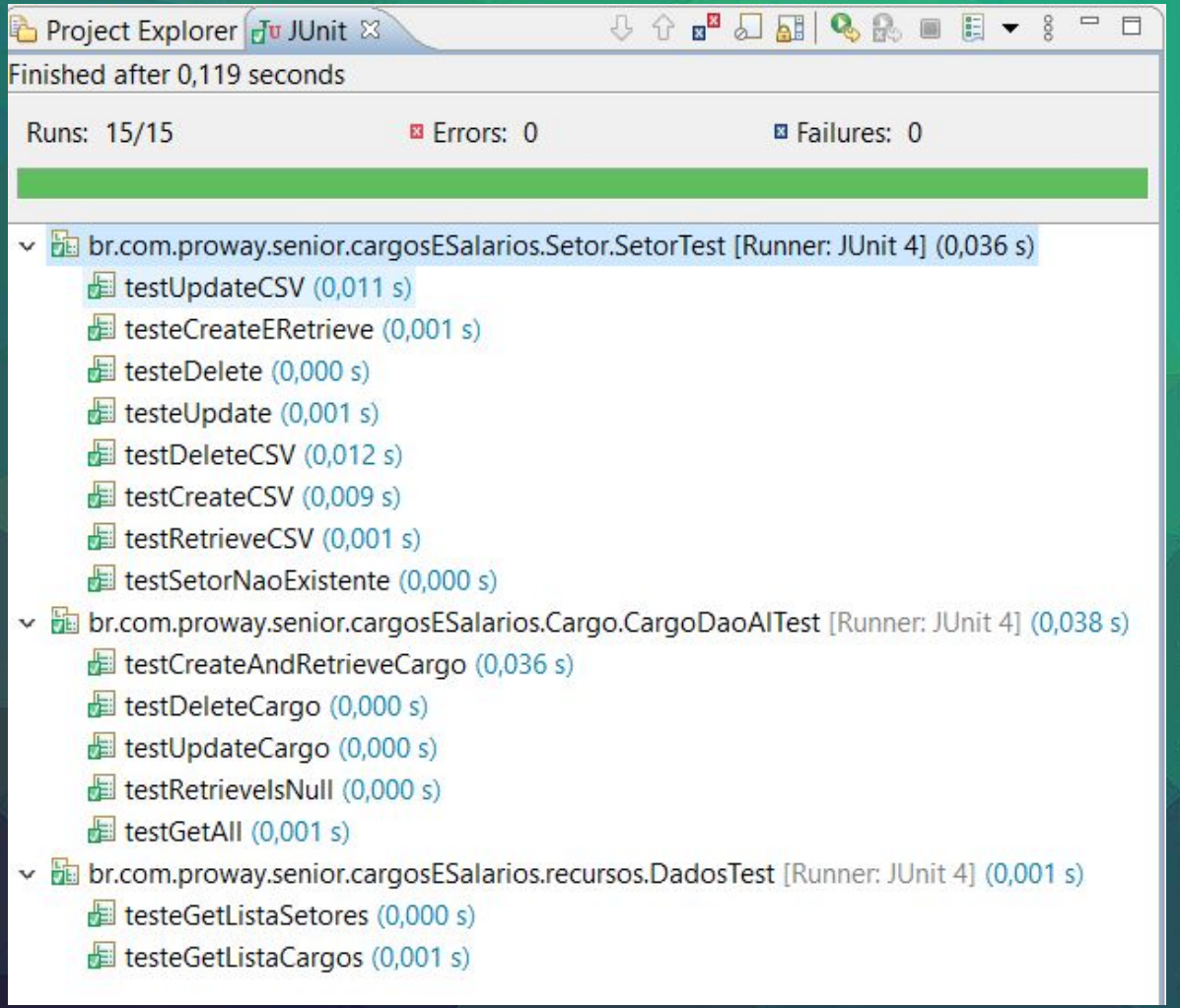
A aplicação do Padrão de Desenho Singleton, permitiu o uso da mesma lista de dados para todo o processo de “persistência” de dados.



Senior ProWay

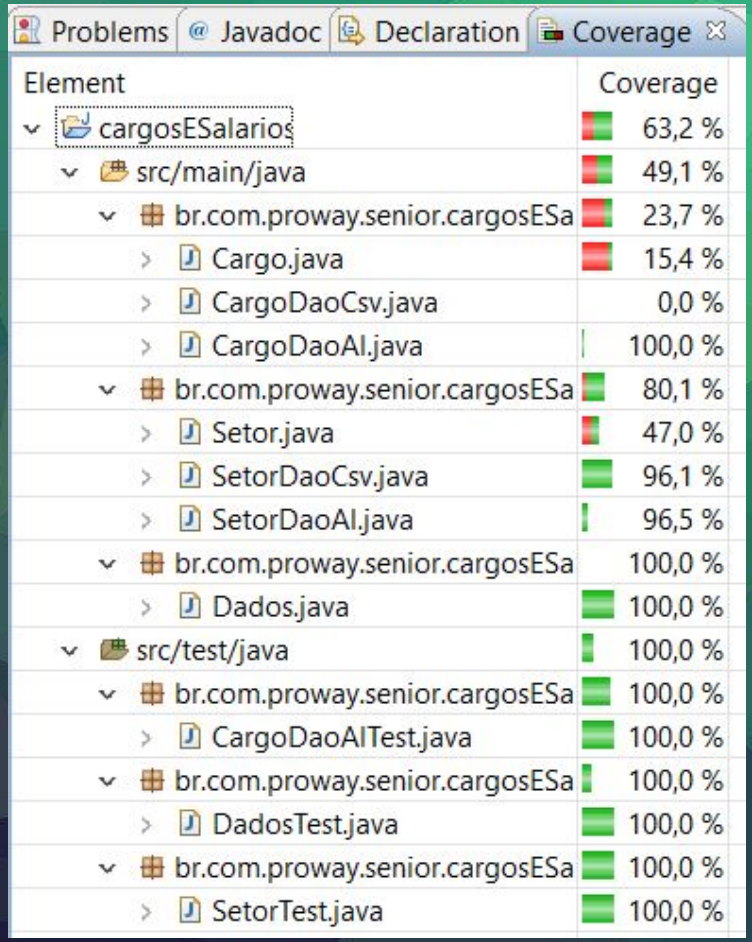

Cargo
- idCargo: Integer
- nomeCargo: String
- idSetor: Integer
- hierarquia: String
- salario: Double
- dataCadastro: LocalDateTime
- dataUltimaRevisao: LocalDateTime
- cbo2002: String
- cbo94: String
- horaMes: Integer
- grauDeInstrucao: String
- experienciaMinima: String
- atribuicoes: String
- bonificacao: String
- status: Integer
- idPermissao: Integer
// Construtor
// Getters & Setters
// toString(), equals(), hashCode()
CargoDaoAl<Cargo>
+ create (Cargo obj) : void
+ retrieve (int id) : Cargo
+ update (Cargo obj) : void
+ delete (int id) : void
+ getAll () : ArrayList<Cargo>
Dados
- instance: Dados [static]
- listaCargos: ArrayList<Cargo>
- listaSetores: ArrayList<Setor>
// Construtor
// Getters
+ getInstance() : Dados [static]
SetorDaoAl<Setor>
+ create (Setor obj) : void
+ retrieve (int id) : Setor
+ update (Setor obj) : void
+ delete (int id) : void
+ getAll () : ArrayList<Setor>
<<interface>>
CRUDInterface<T>
+ create (T obj) : void
+ retrieve (int id) : T
+ update (T obj) : void
+ delete (int id) : void
+ getAll () : ArrayList<T>
Setor
- id: Integer
- nomeSetor: String
- capacidade: Integer
- idPermissao: Integer
// Construtor
// Getters & Setters
// toString(), equals(), hashCode()
SetorDaoCsv<Setor>
- uri : String
+ create (Setor obj) : void
+ retrieve (int id) : Setor
+ update (Setor obj) : void
+ delete (int id) : void
+ getAll () : ArrayList<Setor>
- verificarSetorJaExistente(Setor setor) : boolean
- readAll(Reader reader) : List<String[]>
- csvWriterOneByOne(List<String[]>) : void
cargos.csv

Testes Unitários



Senior ProWay

Cobertura



Senior ProWay

Testes

Setor

```
@Test
public void testeCreateERetrieve() {
    int idSetor = 1;
    SetorDaoAl setorDAO = new SetorDaoAl();
    Setor setor = new Setor(idSetor, "Recursos Humanos", 10, 100);
    Setor setorClone = new Setor(idSetor, "Recursos Humanos", 10, 100);
    setorDAO.create(setor);
    setorDAO.create(setorClone);
    Setor setorRetornado = setorDAO.retrieve(idSetor);
    assertEquals(setorRetornado, setor);
    assertNotSame(setorRetornado, setorClone);
}
```

Testes

Setor

```java
@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Setor other = (Setor) obj;
    if (capacidade != other.capacidade)
        return false;
    if (idPermissao != other.idPermissao)
        return false;
    if (idSetor != other.idSetor)
        return false;
    if (nomeSetor == null) {
        if (other.nomeSetor != null)
            return false;
    } else if (!nomeSetor.equals(other.nomeSetor))
        return false;
    return true;
}
```

Testes

Setor

```java
@Test
public void testSetorNaoExistente() {
    int idSetor = 4;
    SetorDaoAl setorDao = new SetorDaoAl();
    Setor setor = setorDao.retrieve(idSetor);
    assertNull(setor);
}
```

```java
 *
 * Retorna um setor do ArrayList(ListaSetores) pelo id.
 *
 * @param int id
 * @return Retorna o setor procurado ou nulo se não encontrado
 */
public Setor Retrieve(int id) {
    for(Setor setorProcurado : Dados.getInstance().getListaSetores()) {
        if(setorProcurado.getId() == id) {
            return setorProcurado;
        }
    }
    return null;
}
```

Testes

Cargo

```
@Test
public void testDeleteCargo() {
    int idCargo1 = 0;
    Cargo cargo1 = new Cargo(idCargo1, "Gerente", 4, "Supervisor", 500.40, LocalDateTime.now(), LocalDateT
            "5842320-32", "21314", 55, "Superior Completo", "12 meses", "Desenvolvedor", "nenhuma", 1, 1);
    CargoDaoAl cargoDao = new CargoDaoAl();
    int tamanhoInicial = Dados.getInstance().getListaCargos().size();

    cargoDao.create(cargo1);
    assertEquals(tamanhoInicial + 1, Dados.getInstance().getListaCargos().size());
    cargoDao.delete(idCargo1);
    assertEquals(tamanhoInicial, Dados.getInstance().getListaCargos().size());
}
```

Persistência de Dados em CSV

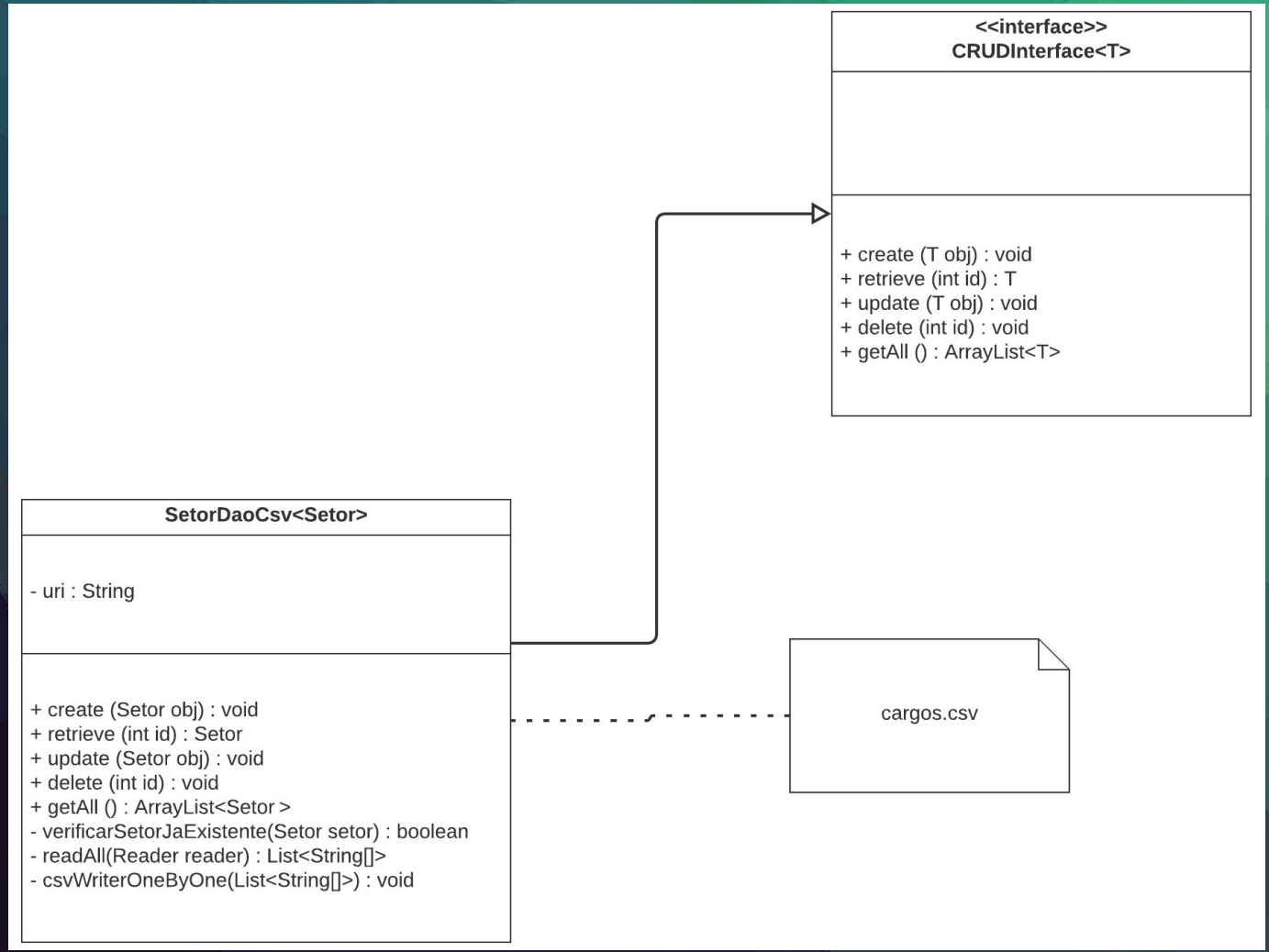
	A	B	C	D
1	idSetor	nomeSetor	capacidade	idPermissao
2	1	Recursos Humanos	30	352
3	2	Comercial	30	657
4	3	Desenvolvimento	45	69542
5				

cargos.csv - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
idSetor;nomeSetor;capacidade;idPermissao
1;Recursos Humanos;30;352
2;Comercial;30;657
3;Desenvolvimento;45;69542
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xml
    <modelVersion>4.0.0</modelVersion>
    <groupId>br.com.proway.senior</groupId>
    <artifactId>br.com.proway.senior.cargosESalarios</
    <version>0.0.1</version>
    <dependencies>
        <dependency>
            <groupId>com.opencsv</groupId>
            <artifactId>opencsv</artifactId>
            <version>4.1</version>
        </dependency>
    </dependencies>
</project>
```


<<interface>>
CRUDInterface<T>
+ create (T obj) : void
+ retrieve (int id) : T
+ update (T obj) : void
+ delete (int id) : void
+ getAll () : ArrayList<T>
SetorDaoCsv<Setor>
- uri : String
+ create (Setor obj) : void
+ retrieve (int id) : Setor
+ update (Setor obj) : void
+ delete (int id) : void
+ getAll () : ArrayList<Setor >
- verificarSetorJaExistente(Setor setor) : boolean
- readAll(Reader reader) : List<String[]>
- csvWriterOneByOne(List<String[]>) : void
cargos.csv

Curiosidade do openCSV

```
163 /**
164  * Método interno para ler todos os dados de um arquivo csv.
165  * @param reader
166  * @return
167  * @throws Exception
168  */
169 private List<String[]> readAll(Reader reader) throws Exception {
170     CSVParser parser = new CSVParserBuilder().withSeparator(';').withIgnoreQuotations(true).build();
171     CSVReader csvReader = new CSVReaderBuilder(reader).withSkipLines(0).withCSVParser(parser).build();
172     List<String[]> list = new ArrayList<String[]>();
173     list = csvReader.readAll();
174     reader.close();
175     csvReader.close();
176     return list;
177 }
```

Sprint retrospective

O que deu certo

- Rodízio de pair programing.
- Interface CRUD generics.
- Entrosamento da equipe para compartilhar ideias.
- Planejamento

O que deu errado

- Testes mais complexos do que o necessário (interface)
- Equals

O que aprendemos

- Implementar DAO com CRUD.
- Não ter medo de mudar o código.
- Estruturar código para facilitar mudanças.
- Design patterns.
- Testes.

Senior ProWay

Sprint retrospective

Fazer mais

- Rodízio pair programing.
- TDD
- Estudar Design patterns

Continuar fazendo

- Estrutura dos packages.
- Planejamento da sprint.
- Dailies organizadas.
- Uso do git.

Começar a fazer

- Aplicar outros Design patterns.

Fazer menos

- Trabalhar nos intervalos e nas aulas.

Parar de fazer

- Commitar na main.

Senior ProWay

Obrigado!

Perguntas?

Senior ProWay