Documento sobre os padrões utilizados.

- Adapter

```
 * @author aliso
 */
public interface LoginCon {

    public boolean usuarioExiste(Usuario user);

    public boolean login(Usuario user);

}
```

```
 */
public class AdapterLogin extends LoginConectServer implements LoginCon {

    private Usuario user;

    @Override
    public boolean usuarioExiste(Usuario user) {
        this.user = DocumentController.CadastroDAO.buscar(user, "2017");
        return this.user != null;
    }

    @Override
    public boolean login(Usuario user) {
        this.user = DocumentController.CadastroDAO.buscar(user, "2017");
        return this.user.getLogin().equalsIgnoreCase(user.getLogin()) && this.user.getLogin().equalsIgnoreCase(user.getLogin());
    }

}
```

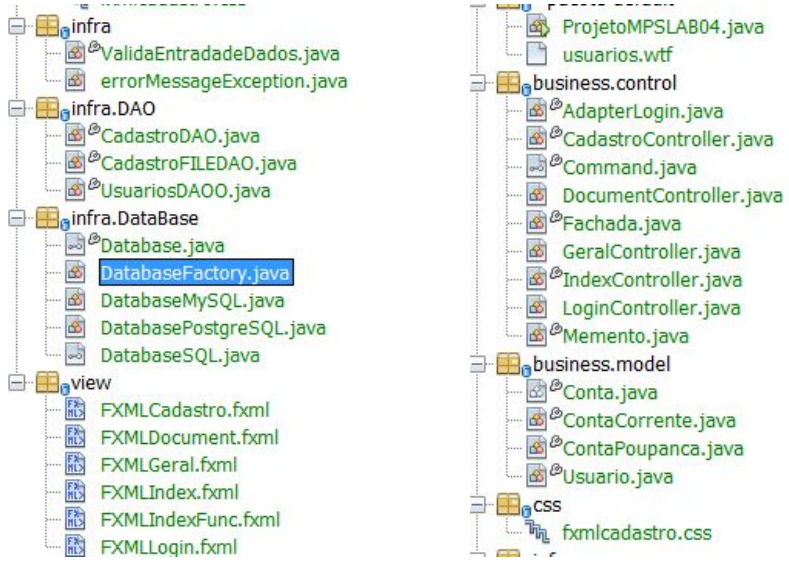
```
    user = new Usuario();
    AdapterLogin api = new AdapterLogin();
    if (!LoginTextField.getText().equalsIgnoreCase("") && !SenhaPasswordField.getText().equalsIgnoreCase("")) {
        user.setLogin(LoginTextField.getText());
        user.setSenha(SenhaPasswordField.getText());

        if (api.usuarioExiste(user)) {
            if (api.login(user)) {
                try {
                    AnchorPane a = (AnchorPane) FXMLLoader.load(getClass().getResource("/view/FXMLIndex.fxml"));
                    AnchorPane.getChildren().setAll(a);

                } catch (IOException ex) {
                    Logger.getLogger(LoginController.class.getName()).log(Level.SEVERE, null, ex);
                }
            } else {
                errorMessageExceptionAmigavel("Campo Login ou senha incorreto");
            }
        } else {
            errorMessageExceptionAmigavel("Usuario não existe");
        }

    } else {
        errorMessageExceptionAmigavel("Campo Login ou Senha vazio");
    }
```

- Camadas view, business e infra



- Factory Method ou Abstract Factory

```
public class DatabaseFactory {
    public static DatabaseSQL getDatabase(String nome){
        switch (nome) {
            case "postgresql":
                return new DatabasePostgreSQL();
            case "mysql":
                return new DatabaseMySQL();
            default:
                break;
        }
        return null;
    }
}
```

- Template Method

```
public class ContaPoupanca extends Conta{

    public ContaPoupanca(double saldo) {
        super(saldo);
    }


    @Override
    public double calculaTaxa() {
        return 0.0;
    }

}
```

```
public class ContaCorrente extends Conta{

    public ContaCorrente(double saldo) {
        super(saldo);
    }
    @Override
    public double calculaTaxa() {
       return 0.2;
    }


}
```

- Facade

```
 * @author aliso
 */
public class Fachada {

    private final UsuariosDAOO usuarios;

    //
    public Fachada(UsuariosDAOO usuarios) throws errorMessageException {
        this.usuarios = usuarios;
    }

    public Object service(Command c) throws Exception {
        Object o = c.execute(usuarios);
        return o;
    }

    public void undo(Memento mem) throws Exception {
        Command c = mem.getEstado();
        if (!(c == null)) {
            c.execute(usuarios);
            mem.setEstado(null);
        }else{
            throw new errorMessageException("Nenhuma operação para desfazer!");
        }

    }

}
```

- Testes Unitários

```
public class ValidaEntradadeDados {

    private String login;
    private String senha;
    private boolean valid;
    private String errorMessageExceptionAmigavel;

    public ValidaEntradadeDados(String login, String senha) {
        this.login = login;
        this.senha = senha;
        this.valid = true;
    }
```

```java
public void valida() {
    if (login.length() > 12) {
        valid = false;
        errorMessageExceptionAmigavel = "Campo login é inválido\n"
                + "Deve conter no máximo 12 letras! por favor, verifique novamente...\n";
        //throw new errorMessageException("Campo login é inválido\n"
        //        + "Deve conter no máximo 12 letras! por favor, verifique novamente...\n");

    } else if (login.equalsIgnoreCase("")) {
        valid = false;
        errorMessageExceptionAmigavel = "Campo login é inválido\n"
                + "Não pode ser vazio! por favor, verifique novamente...\n";

    } else if (login.matches(".*\\d.*")) {
        valid = false;
        errorMessageExceptionAmigavel = "Campo login é inválido\n"
                + "Não Deve conter números! por favor, verifique novamente...\n";

    }

    if (senha.length() > 20) {
        valid = false;
        errorMessageExceptionAmigavel = "Campo senha é inválido\n"
                + "Deve conter no máximo 20 caracteres! por favor, verifique novamente...\n";

    } else if (senha.length() < 8) {
        valid = false;
        errorMessageExceptionAmigavel = "Campo senha é inválido\n"
                + "Deve conter pelo menos 8 caracteres! por favor, verifique novamente...\n";

    } else if ((!senha.matches("[a-zA-Z]*")) && (!senha.matches(".*\\d.*")) || (!senha.matches(".*\\d{2,}.*"))) {
        valid = false;
        errorMessageExceptionAmigavel = "Campo senha é inválido\n"
                + "Deve conter letras e pelo menos 2 números! por favor, verifique novamente...\n";

    }

}
```