Trabalho Final
Engenharia de Software - N

Eduardo Renani, Victor Almeida, Marcellus Farias e Rodolfo Helfenstein

1. Arquitetura

Como escrito pelo grupo analista, o usuário da aplicação teria acesso a lista de DVD`s disponíveis para troca tanto do seu computador pessoal em casa, quanto de algum computador/totem disposto dentro do estabelecimento da empresa OldButGold. Logo, uma aplicação distribuída seria a solução mais rápida e eficaz para o problema, visto que temos uma imensa gama de ferramentas e linguagens para Web e qualquer device conectado a internet poderia acessar a base de DVD`s da empresa.

Afim de realizarmos um desenvolvimento e deploy rápido, aliado ao fato de nossas classes serem de poucos relacionamentos, escolhemos pela *stack* NodeJS (backend), ReactJS (frontend) e MongoDB (banco de dados NoSQL), seguindo o padrão de arquitetura MVC. Como tanto o backend quanto frontend são desenvolvidos em Javascript e o banco de dados possuí uma sintaxe de consulta altamente abstraída, o reuso de conhecimento no desenvolvimento das partes do código foi altíssimo, sendo tudo escrito na mesma linguagens e sem necessidade de converter/desembrulhar estruturas de dados.

2. Casos de Uso e Simplificaçōes

Após checarmos toda a lista de casos de uso, escolhemos, dentre os Essenciais, os dois casos de uso que agregariam o maior valor ao usuário, permitindo um prova de conceito no momento da demonstração da aplicação. Foram estes:

a. Registro de Troca (Funcionário)
b. Consulta a Catálogo (Usuário)

Para o caso de uso (a), foram simplificadas questões referentes ao log da troca do usuário e o tipo de mídia, visto que o *core* o caso de uso era o registro coerente no banco de dados de que determinado DVD foi retirado e outro foi posto no lugar. Assim, durante uma primeira iteração com o cliente já seria possível enxergar, em tempo real, os DVD`s sumindo e aparecendo da base de dados.

Para o caso de uso (b), foram omitidas as informações sobre o tipo de mídia; mostrando apenas o título do filme, o ano de lançamento e o número de cópias.

Afim de tonar a demonstração mais visual e sem comprometer as boas práticas de desenvolvimento modular, fizemos um mockup de login que permite, a partir da checagem do nome do usuário, redirecionar para a página de visualização de acervo disponível (*Home* do usuário) ou para página de gerenciamento de acervo (*Home* do funcionário). Para isso, arbitramos que o login “Admin” seria o login de funcionário e qualquer outro *input* seria um login de usuário.

3. Diagrama de Classes UML

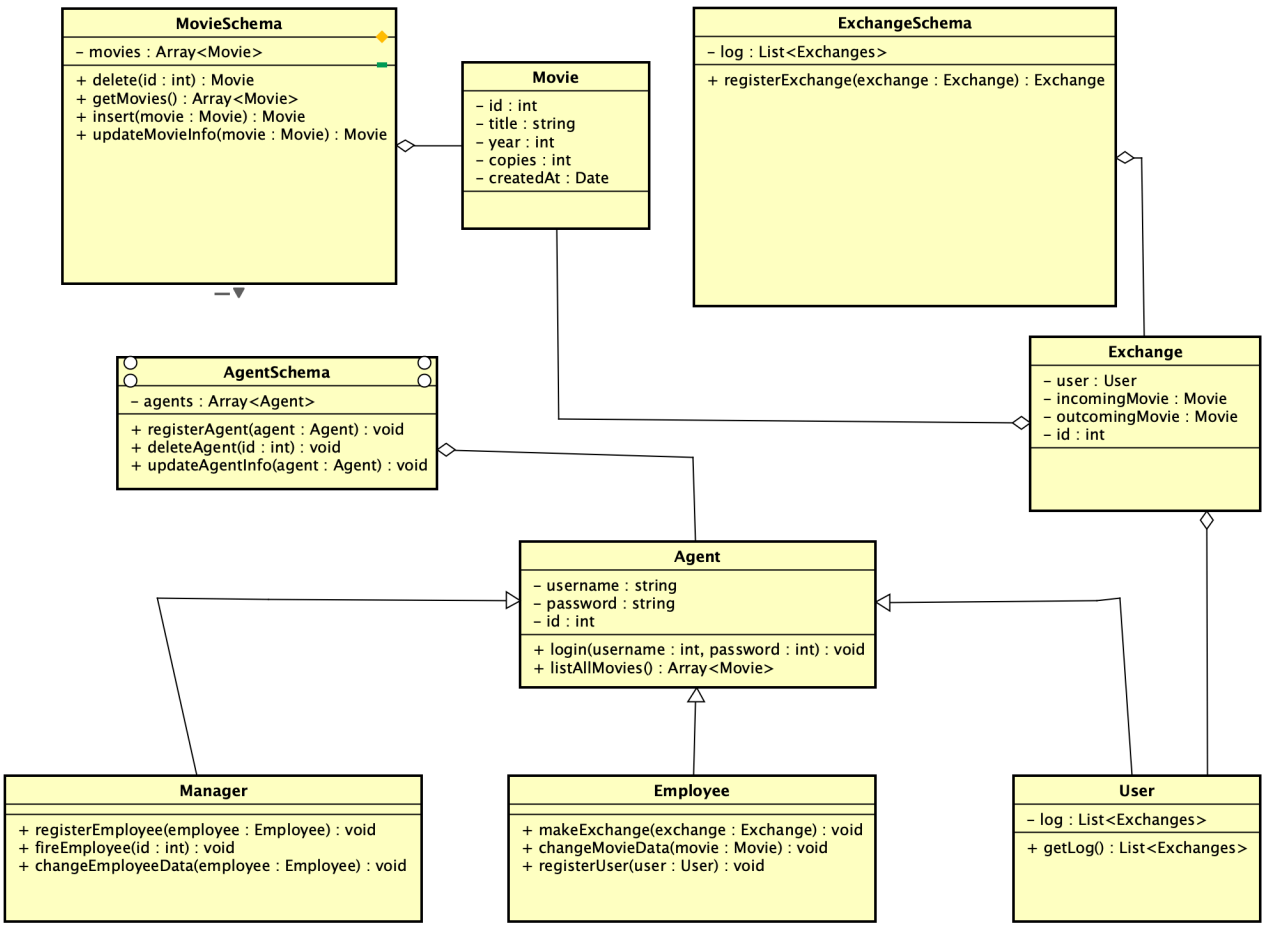


Diagrama de classes backend

4. Implementação dos casos de uso (CRUD e Schema)

```
async store(req, res) {
    var movie;
    try {
        movie = await Movie.findOneAndUpdate(
            { title: req.body.title },
            {
                $inc:
                    {
                        copies: 1
                    },
            },
        );
        if( movie === null) throw "null"

        req.io.emit("newMovieAddCopy", movie);
    }
    catch(err) {

        movie = await Movie.create(req.body);
        req.io.emit("newMovie", movie);

    }

    return res.json(movie);

},
```

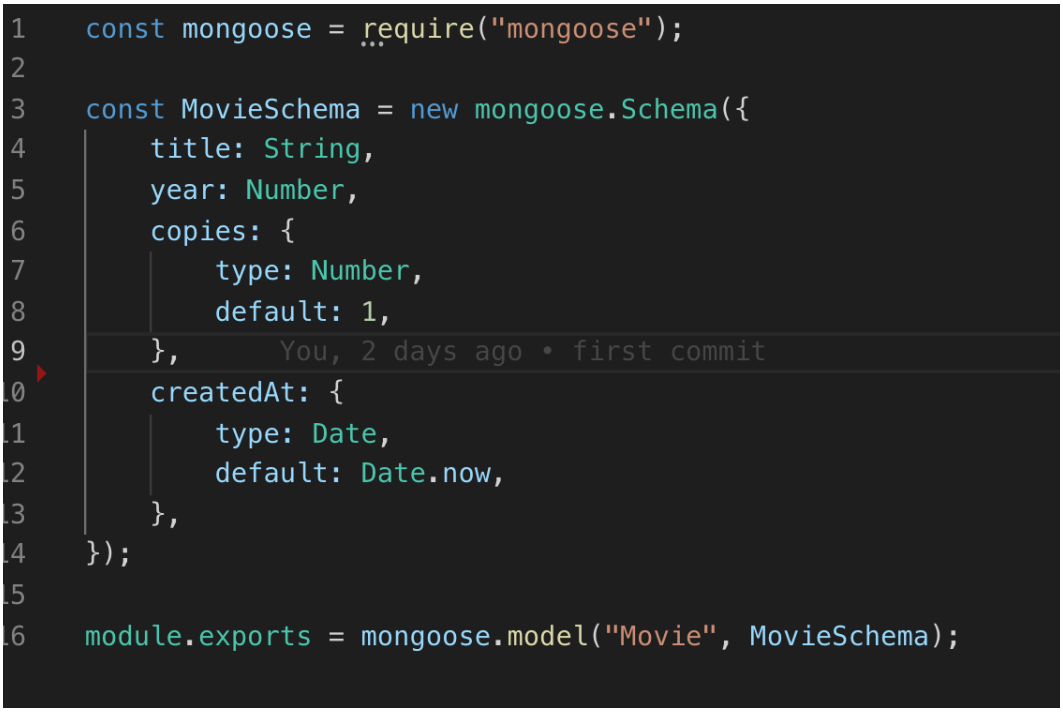
Criação de filme no bando de dados

```
async index(req, res){
    const movies = await Movie.find({}).sort("-year");

    return res.json(movies);
},
```

Leitura de todos os filmes da base e retorno da lista para o frontend

```
async delete(req, res) {

    const movie = await Movie.findById(req.params.id);

    if( movie.copies > 1 ) {

        movie.set({ copies: (movie.copies - 1) });

        req.io.emit("removeMovieCopy", movie);

    }
    else {

        movie.delete();
        req.io.emit("removeMovie", movie);

    }

    await movie.save();

    return res.json(movie);

}
```

Deleção um filme da base de dados



```
const mongoose = require("mongoose");

const MovieSchema = new mongoose.Schema({
    title: String,
    year: Number,
    copies: {
        type: Number,
        default: 1,
    },      You, 2 days ago • first commit
    createdAt: {
        type: Date,
        default: Date.now,
    },
});

module.exports = mongoose.model("Movie", MovieSchema);
```

Classe Movie do banco de dados