Node:

Seguimos o modelo MVC:

- ➔ Os Models são a ponte entre as camadas View (que é realizada em angular) e (Controller), que consiste na parte lógica da aplicação, que monitoriza o comportamento dos dados através de regras de negócios, lógica e funções. Fica apenas à espera da chamada por parte das funções, que permite o acesso para os dados serem recolhidos, gravados e, exibidos. e estão divididos em Order e User sendo que os
- ➔ Os Controllers são responsáveis pela mediação da entrada e saída, comandando a visão e o modelo para serem alterados de forma apropriada conforme o utilizador solicitou através do pedido. O foco do Controller é a ação do utilizador, onde são manipulados os dados que o utilizador insere ou atualiza, chamando em seguida o Service.

Para a criação do Model Order, é usado a const orderSchema que contem os seguintes atributos:

- `_id`
- `User`
- `Product` (que utiliza um objeto com dados retornados do MDP)
- `Quantity`
- `Status`
- `Changestatusorder` (data atualizada quando ocorre mudança de status)
- `Date` (data do pedido)
- `Estimateddeliverydate` (data de entrega estimada)

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

var ProductObject =  {
    productId: Number,
    name: String,
    price: Number,
    duration: Number,
    fabricPlanId: Number
};

const orderSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required:
true },
    product: ProductObject,
    quantity: { type: Number, required: true },
    status: {type: String, default: 'Created'},
    changestatusorder: {type: Date, default: Date.now},
    date: {type: Date, default: Date.now},
    estimateddeliverydate: {type: Date}
```

```
});

module.exports = mongoose.model('Order', orderSchema);
```

Para a criação do Model User, é usado a const orderSchema que contem os seguintes atributos:

- _id
- name
- email
- password
- address
- role (que utiliza um objeto com a definição do papel do utilizador)
- actions (que utiliza um array de objetos com perfis de edição)

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

const Rules = {
    admin: 0,
    client: 1
}

const ActionsDefault = {
    NewOrder: 1,
    ViewOrder: 1,
    EditOrder: 0,
    EditClientAdmin: 0,
    EditClient: 1
}

const userSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    name : { type: String, required: true },
    email: {
        type: String,
        required: true,
        useCreateIndex: true,
        match: /[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-
9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-
9](?:[a-z0-9-]*[a-z0-9])?/
    },
    password: { type: String, required: true },
    address: { type: String, required: true },
    role: { type: Number, required: true, default: Rules.client },
    actions: { type: Array, required: true, default: ActionsDefault }
});
```

```
module.exports = mongoose.model('User', userSchema);
```

Interação dos componentes

Além de dividir a aplicação em três tipos de componentes, o desenho MVC define as interações entre eles.

O Controlador (Controller) envia comandos para o modelo para atualizar o seu estado (por exemplo, editando um documento). O controlador também pode enviar comandos para a visão associada para alterar a apresentação da visão do modelo (por exemplo, percorrendo um documento).

Um modelo (Model) armazena dados e notifica os controladores associados quando há uma mudança no seu estado. Estas notificações permitem que as visões produzam saídas atualizadas e que os controladores alterem o conjunto de comandos disponíveis.

A visão (view) gera uma representação (Visão) dos dados presentes no modelo solicitado, fazendo a exibição dos dados, sendo ela por meio de um HTML.

Angular:

- ➔ Visão pode ser qualquer saída de representação dos dados, como uma tabela. É onde os dados do Modelo (*Model*) são exibidos. A Visão também provoca interações com o utilizador, que interage com o *Controller*.
- ➔ Existe um layout definido como template para cada componente e existe um componente específico para cada tipo de ação.
- ➔ Foram definidas permissões específicas para serem mostradas a utilizadores com Roles definidos à priori.