# LAB 11 – Construção de uma aplicação de reservas (E-booking)

(programação em HTML, CSS, Bootstrap, Symfony, JavaScript, JQuery, AJAX)

---

O objectivo do trabalho é fazer uma aplicação de reservas online ("E-booking").

Requisitos:

- A aplicação deve ser realizada utilizando o paradigma "**model-view-controller**" --- obrigatóriamente o código (controlador) deverá estar totalmente isolado (num outro ficheiro) do template HTML (vista) da página web.
- Os templates podem ser livremente descarregados da net, mas o aluno **não pode utilizar** os mesmos templates que outro aluno[1]
- **O código "backend" da aplicação tem que ser realizado pelo aluno**.
- **A aplicação tem que ser realizada utilizando o framework Symfony versão 4.4**[2]
- **A aplicação tem que utilizar a *template engine* Symfony Twig**
- No "frontend" da aplicação é permitido livremente o uso de código JavaScript (AJAX, JQuery, VueJS) inclusive de código *open source* descarregado da Internet
- A aplicação tem que disponibilizar graficamente, sob a forma de uma tabela ou outra, informação sobre as reservas já existentes. Esta funcionalidade pode ser realizada por um "plugin" em JavaScript.

- A aplicação deve implementar regras de segurança básicas, por exemplo utilizadores anónimos que tentam aceder a páginas do site de acesso reservado a utilizadores registados são imediatamente re-direccionados para a página de rosto do site.

[1] Caso, por coincidência não intencional, o aluno utilize templates já utilizados por outro aluno, não haverá qualquer penalização

[2] Pode utilizar a versão 5.3 do Symfony ou mais recente, mas assegure-se que essa versão do Symfony corre no servidor de produção!

- A aplicação é suportada por uma base de dados; uma **sugestão** para a estrutura da base de dados é dada em apêndice, mas a base de dados tem que ser adaptada ao tipo de reservas.

- Tem que haver três ou mais recursos passíveis de reserva, e as reservas podem abranger dois ou mais momentos contiguos. Os momentos de reserva podem ser horas ou dias. Nota: NÃO é permitido overbooking!

## EXEMPLOS DE RECURSOS

| Actividade | Recurso |
|---|---|
| Alojamento (hotel, local etc) | quartos |
| Clinicas (médica, dentária, veterinária etc) | médicos |
| Profissões liberais (advogados, contabilistas etc) | Adv., med. etc |
| Transportes (comboio, avião, autocarro) | Lugar |
| Restauração (restaurante...) | mesa |
| Hospital | Sala de operações |

Outros exemplos de actividades são Garagens (recurso: lugares de parqueamento), Agências de emprego, Bancos, Agências Imobiliarias, etc (recurso: entrevistadores).

## SERVIDOR DE PRODUÇÃO

A **aplicação deve correr obrigatóriamente** na área pessoal do aluno no directório "**LAB11**" no servidor web do departamento disponibilizado para o efeito: http://daw.deei.fct.ualg.pt/ (http://10.10.23.184/ na rede UALG).

**O controlador principal tem que estar no ficheiro**

"`EbookingController.php`"

**A classe com as funções de acesso à base de dados tem que estar no ficheiro**

"`Ebooking_modelController.php`"

## NOTA:

- **Opcional!** Caso use o componente **Doctrine ORM** a restrição acima é levantada e pode utilizar um modelo para cada tabela da base de dados. Pode ainda utilizar mais do que um controlador.
- **Opcional!** Pode utilizar **bin/console** para "scaffolding" das funcionalidades de registo e autenticação de utilizadores[3].

- **Opcional!** Pode utilizar **Doctrine SQL Query Builder** para acessar a base de dados.

Se vai utilizar o seu PC ou portátil como sistema de desenvolvimento **deve instalar a versão Symfony 4.4**, caso contrário poderá ser obrigado a modificar o seu código para ele correr no servidor web daw2 (IP 10.10.23.184) do departamento!

**A página de entrada no site tem que ser**

**http://daw.deei.fct.ualg.pt/~a12345/LAB11/public/index.php/ebooking**

**ou caso configure URL rewriting no servidor**

**http://daw.deei.fct.ualg.pt/~a12345/LAB11/ebooking**

[3] Pode livremente adicionar tabelas que os componentes Symfony necessitem (por exemplo a tabela "`reset_password_request`" utilizada pelo componente "Reset Password Bundle"...)

# PRELIMINARES

Faça login no servidor daw2[4] (IP 10.10.23.184)

### A. Download  o framework para dentro da pasta "public_html/LAB11"

```
a12345@daw2:~$ cd public_html

a12345@daw2:~/public_html$
git clone git://github.com/jmatbastos/LAB11.git
```

O comando cria a pasta `LAB11` com a instalação do Symfony 4.4 e todas as livrarias standard.

Note bem: **`LAB11`** SIM, Lab11, lab11 NÃO

### B. Complete a instalação

```
a12345@daw2:~/public_html$ cd LAB11

a12345@daw2:~/public_html/LAB11$ composer install
```

### C. Altere o ficheiro ".env" para utilizar as credenciais da sua base de dados.

Substitua "a12345" pelo seu login e "******" pela password de acesso à sua base de dados[5]

```
a12345@daw2:~/public_html/LAB11$ nano .env

DATABASE_URL=mysql://a12345:******@127.0.0.1:3306/db_a12345?serverVersion=15.1
```

### D. Crie a sua cópia da base de dados

```
a12345@daw2:~/public_html/LAB11$

mysql -u a12345 --password=****** db_a12345 < database.SQL
```

onde "`database.SQL`" é um ficheiro com a estrutura  da sua base de dados

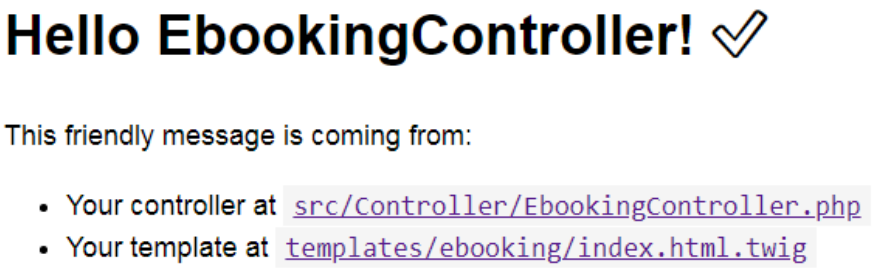
[4] O framework Symfony NÃO funciona no servidor daw (IP 10.10.23.183)
[5] Se não se recorda da password da sua base de dados, recupere-a com o comando
`a12345@daw2:~$ /usr/local/bin/mysql-db`

**E. Teste a instalação do framework.**

A partir do seu browser preferido vá ao seguinte URL

**`http://daw.deei.fct.ualg.pt/~a12345/LAB11/ebooking`**

Deverá receber uma página web de boas-vindas:




---

# ESTRUTURA DO SITE

O site tem as seguintes páginas:

1. "**Home**" é página de entrada no site;
2. "**Recursos**" apresenta os recursos que se podem reservar;
3. "**Details**" apresenta as características de um recurso em particular, e permite a sua reserva a utilizadores registados;
4. "**Register**" para registo de utilizadores;
5. "**Login**" para o login de utilizdores;
6. "**My Bookings**" permite aos utilizadores registados consultar uma listagem das suas reservas

O site pode ter mais (ou menos) páginas, desde que a funcionalidade mínima seja realizada!

Deve ser considerado uma **SUGESTÃO** o seguinte mapeamento entre URLs e controladores:

- `@Route("/ebooking", name="home")`
- `@Route("/ebooking/resources", name="resources")`
- `@Route("/ebooking/resource/details/{id}", name="details")`
- `@Route("/ebooking/register", name="register")`
- `@Route("/ebooking/login", name="login")`
- `@Route("/ebooking/logout", name="logout")`
- `@Route("/ebooking/book/{id}", name="book")`
- `@Route("/ebooking/mybookings", name="mybookings")`
- `@Route("/ebooking/message", name="message")`

**NOTAS:**

- A validação dos dados introduzidos nos formulários deve ser realizada com o serviço **Validator** do Symfony

Apresenta-se no URL

https://daw.deei.fct.ualg.pt/~a999997/ebooking/motel

um site exemplo. O seu site pode ser uma aplicação de reservas muito diferente, seja criativo!

**REFERÊNCIAS:**


- https://www.w3schools.com/bootstrap/default.asp
- https://symfony.com/doc/4.4//index.html
  http://intranet.deei.fct.ualg.pt/DAW/slides/SF_overview.pdf
- http://intranet.deei.fct.ualg.pt/DAW/slides/SF_doctrine.pdf
- http://all.deei.fct.ualg.pt/symfony/
- https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/index.html
- https://www.w3schools.com/js/default.asp
- https://www.w3schools.com/jquery/default.asp

**ANEXO : estrutura da base de dados**

A estrutura da base de dados pode ser criada em

http://daw.deei.fct.ualg.pt/phpMyAdmin

ou com o comando

```
a12345@daw2:~$mysql -u a12345 -p db_a12345 < database.SQL
```

onde `database.SQL` é o ficheiro que contem a estrutura da base de dados:

Mostra-se em seguida a estrutura da base de dados utilizada no site exemplo.
**Nota:** Tem que adaptar a base de dados à sua aplicação!

```
--
-- Table structure for table `users`
--

CREATE TABLE IF NOT EXISTS `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `created_at` datetime NOT NULL,
  `updated_at` datetime NOT NULL,
  `password_digest` varchar(255) DEFAULT NULL,
  `remember_digest` varchar(255) DEFAULT NULL,
  `admin` tinyint(1) DEFAULT NULL,
  `activation_digest` varchar(255) DEFAULT NULL,
  `activated` tinyint(1) DEFAULT NULL,
  `activated_at` datetime DEFAULT NULL,
  `reset_digest` varchar(255) DEFAULT NULL,
  `reset_sent_at` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Table structure for table `room_categories`
--


CREATE TABLE IF NOT EXISTS `room_categories` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `description` varchar(255) DEFAULT NULL,
  `adults` int(2) DEFAULT NULL,
  `bed_type` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `room_categories`
--
```

```
INSERT INTO `room_categories` VALUES (1,'single',1,'single
bed'),(2,'double',2,'double bed'),(3,'suite',4,'water bed');

--
-- Table structure for table `rooms`
--


CREATE TABLE IF NOT EXISTS `rooms` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `price` smallint(6) DEFAULT NULL,
  `category_id` int(11) NOT NULL,
  `facilities` varchar(255) DEFAULT NULL,
  `size` int(11) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `category_id` (`category_id`),
  CONSTRAINT `rooms_ibfk_1` FOREIGN KEY (`category_id`) REFERENCES
`room_categories` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


--
-- Dumping data for table `rooms`
--
INSERT INTO `rooms` VALUES (1,'Bachelor Room',50,1,'Closet with hangers,
HD flat-screen TV, Telephone',10,'1.jpg'),(2,'Family Room',100,2,'Closet
with hangers, HD flat-screen TV, Telephone',20,'4.jpg'),(3,'Presidential
Room',300,3,'Closet with hangers, HD flat-screen TV,
Telephone',40,'2.jpg');

--
-- Table structure for table `bookings`
--


CREATE TABLE IF NOT EXISTS `bookings` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(255) NOT NULL DEFAULT 'RESERVED',
  `created_at` datetime DEFAULT NULL,
  `start` date NOT NULL,
  `end` date NOT NULL,
  `n_nights` int(11) DEFAULT NULL,
  `allDay` int(1) NOT NULL DEFAULT '1',
  `requests` varchar(255) DEFAULT NULL,
  `room_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `room_id` (`room_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `bookings_ibfk_1` FOREIGN KEY (`room_id`) REFERENCES
`rooms` (`id`),
  CONSTRAINT `bookings_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES
`users` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

NOTA:

- Se quiser fazer o site utilizando o componente **Doctrine ORM**, pode gerar uma classe para cada tabela com os comandos

```
php bin/console doctrine:mapping:import "App\Entity" annotation --path=src/Entity
php bin/console make:entity --regenerate App
```

A tabela "users" em Apêndice tem que ser alterada adicionando a coluna "roles"

```
ALTER TABLE users ADD `roles` longtext COMMENT
'(DC2Type:json)';
```

Referência
https://symfony.com/doc/current/doctrine/reverse_engineering.html