

SALNE

Desarrollo del prototipo de una aplicación
administrativa y de venta en línea para una
librería

Idea y propósito del proyecto original

- Sistema administrativo unificado para los clientes (librerías)
- Base de datos estandarizada y rica en contenido
- Infraestructura como base e interfaces para desarrollos consecuentes

- COLABORACIÓN

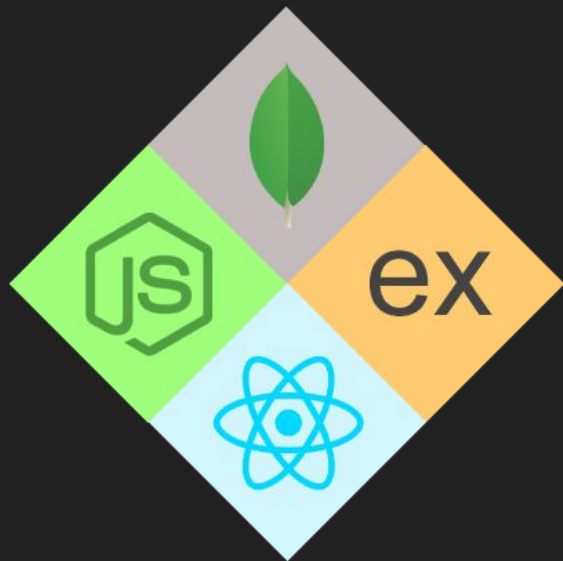
Lo que se va a adaptar para este trabajo

- Sistema administrativo individual (sin soporte para múltiples instituciones)
- Base de datos con modelos e información básica
- Aplicación administrativa y de usuario integradas en el mismo paquete
- El objetivo colaborativo no cambia
- Es necesario comprometer objetivos y funcionalidades para cumplir con las limitaciones la situación

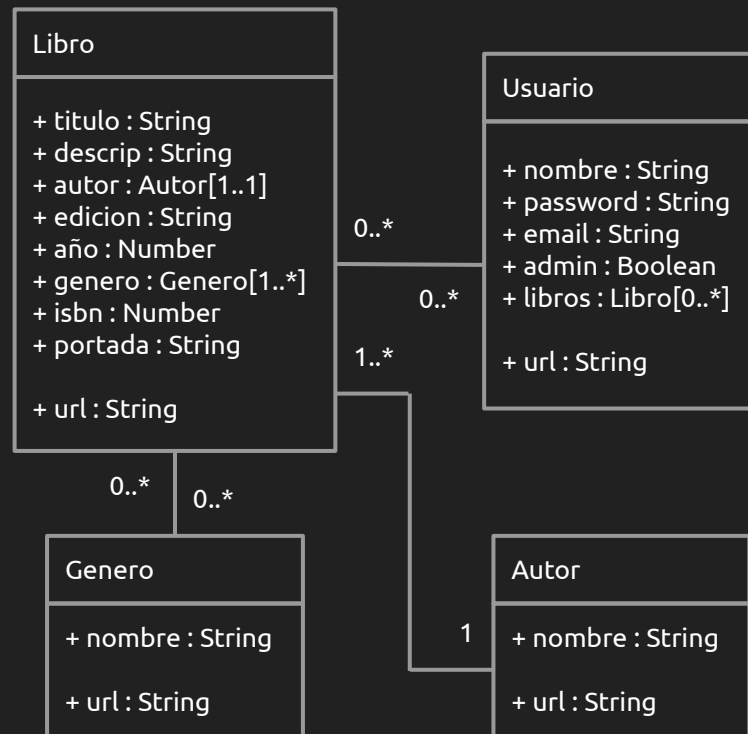
¿Por dónde empezar?

Seleccionar tecnologías

- Plataforma de desarrollo web
- Siguiendo el modelo/stack **MERN/MEAN**



Definir modelos y datos



Estructuración del proyecto

Usando el patrón MVC

- controllers (Controladores - JavaScript)
 - **controlador.js**
 - ...
- models (Modelos - MongoDB/Mongoose)
 - **modelo.js**
 - ...
- node_modules (Módulos/Dependencias - Node.js)
 - ...
- public (Almacenamiento, Estilos, Scripts)
 - stylesheets (CSS)
 - **general.css**
 - ...
 - scripts (JavaScript/jQuery)
 - ...
- routes (Enrutamiento - Express)
 - **ruta.js**
 - ...
- views (Vistas - HTML/Nunjucks)
 - includes (Código Estático)
 - layouts (Plantillas de Página)
 - **vista.html**
 - ...
 - **app.js** (Punto de Inicio - Express)
 - **package.json** (Configuración de Node.js)

Iniciando el desarrollo



express

- Ya que usamos **Express**, podemos generar un proyecto directamente con **Express Generator**:

```
$ npm install express-generator
```

```
$ express salne
```

- Preparamos la aplicación e instalamos las dependencias:

```
$ cd salne
```

```
$ npm install
```

- El servidor local ya se puede ejecutar (y visitar en **localhost:3000**)

- Dentro de **Express** podemos importar módulos de **Node.js** y objetos como enrutadores, controladores, modelos, etc.

```
// Importar módulos de Node.js
var express = require('express');
var session = require('express-session');
var MongoStore = require('connect-mongo');

// Indicar dónde buscar rutas según la url
var IndexRouter = require('./routes/index');
app.use('/', IndexRouter);

// Importar modelo de objeto, estructurado con Mongoose
var UsuarioModel = require('./models/usuario');
```

Implementar la base de datos



mongoDB

mongoose

elegant **mongodb** object modeling for **node.js**

- Para la base de datos elegí MongoDB (NoSQL), implementando el middleware **Mongoose**

```
$ sudo apt install mongodb
```

```
$ npm install mongoose
```

```
// Importar paquete
```

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

```
// Establecer conexión (local o externa a MongoDB Atlas)
```

```
var mongoDB = 'mongodb://127.0.0.1/libreria';
```

```
mongoose.connect(mongoDB, { useNewUrlParser: true, useUnifiedTopology: true });
```

```
var db = mongoose.connection;
```

```
db.on('error', console.error.bind(console, 'MongoDB connection error:'));
```

- **Mongoose** permite modelar objetos (esquemas) con facilidad:

```
const { Schema, model } = require('mongoose');

// Crear modelo en base a esquema (como objetos en JavaScript)
const usuarioSchema = new Schema({
  nombre: { type: String, required: true, maxlength: 50 },
  password: { type: String, required: true, maxlength: 100 },
  email: { type: String, required: true, maxlength: 100 },
  admin: { type: Boolean, required: true },
  libros: [{ type: Schema.Types.ObjectId, ref: 'libro' }]
});

// Exportar modelo
module.exports = model('usuario', usuarioSchema);
```

- Además facilita las operaciones con la base de datos (guardar, actualizar, eliminar)

```
// Instanciar nuevo modelo
```

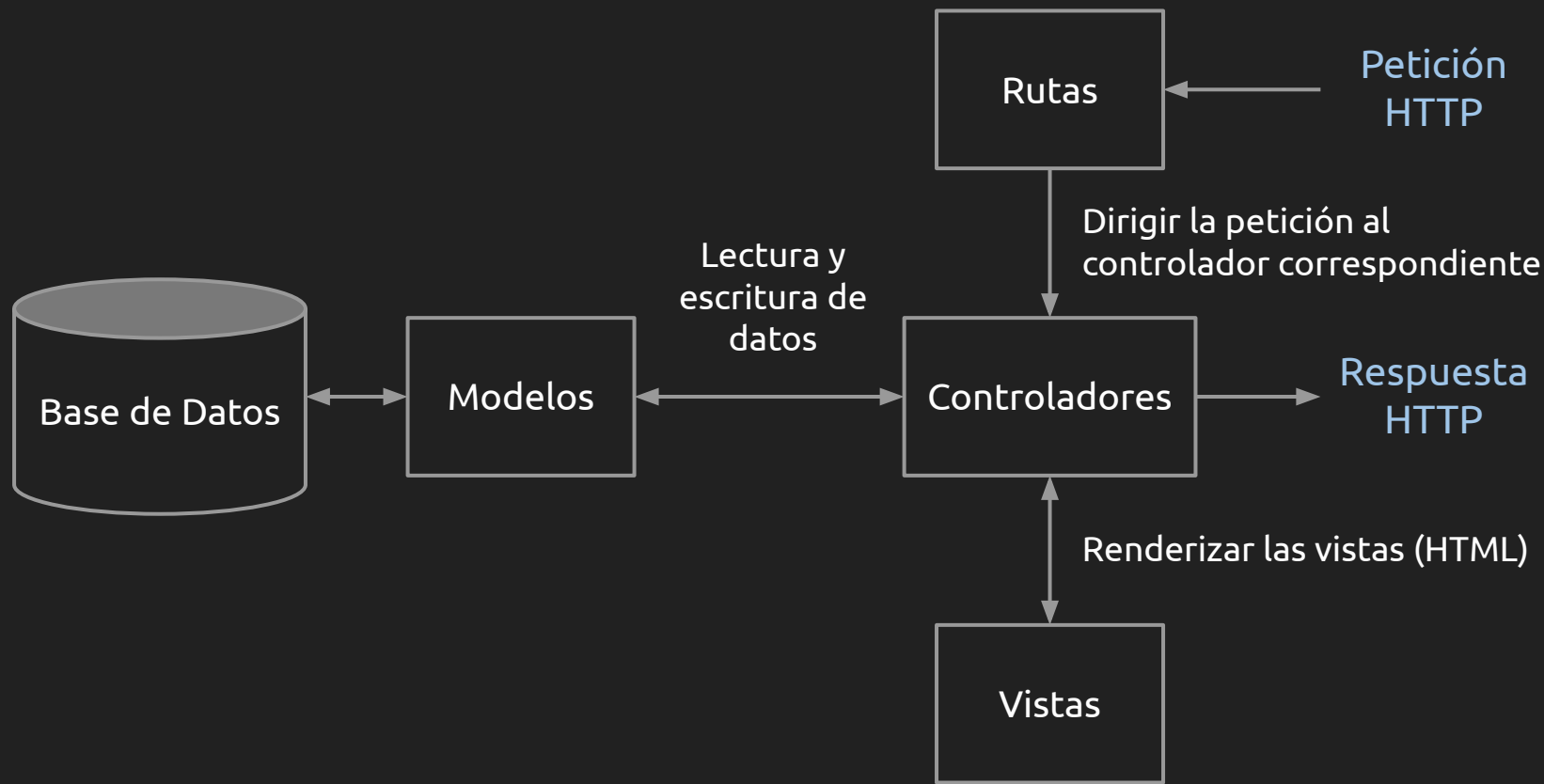
```
var Usuario = new UsuarioModel({  
  nombre: req.body.nombre,  
  password: req.body.password,  
  email: req.body.email,  
  admin: false  
});
```

```
// Buscar modelo en la base de datos
```

```
UsuarioModel.findOne({ 'nombre': req.body.nombre }, function (err, usuario) {  
  // Si se encuentra, guardar la instancia del modelo en la base de datos  
  if (!usuario) Usuario.save();  
});
```

Integrar y adaptar al modelo MVC

Implementando enrutamiento de páginas y
separación de la lógica en base a nuestro modelado



Enrutamiento

- Las funciones de enrutamiento ya vienen incluidas con Express

```
// Indicar dónde buscar rutas según la url  
var IndexRouter = require('./routes/index');  
app.use('/', IndexRouter);
```

- En el archivo, se especifica el tipo de petición, ruta y función a ejecutar

```
// Instanciar el controlador con las funciones necesarias  
var IndexController = require('./controllers/index');  
// Indicar configuración del enrutamiento  
var router = express.Router();  
router.get('/', IndexController.index);  
router.get('/catalogo', IndexController.catalogo);
```


Controladores

- Instruyen el procesamiento lógico de la aplicación

```
module.exports.index = function (req, res, next) {  
  res.render('layout', { content: 'index', title: " });  
}
```

```
var LibroModel = require('../models/libro');  
module.exports.catalogo = function (req, res, next) {  
  LibroModel.find(function (err, librosLista) {  
    // Renderizar la vista indicada (utilizando plantillas)  
    res.render('layout', { content: 'catalogo', title: ' - Catálogo', libros: librosLista });  
  });  
}
```

Vistas

- Usando un **Template Engine (Nunjucks)**, es procesado por el controlador y devuelto como la interfaz gráfica de la aplicación

```
<!DOCTYPE html>
<html lang="en">
{% include "includes/head.html" %}
<body>
  {% include "includes/sidebar.html" %}
  <div class="contents">
    {% include content + ".html" %}
  </div>
</body>
</html>
```

content



```
<div class="index">
  <div class="header">
    <p class="titulo">SALNE</p>
    <p class="sub-titulo">Sistema de Administración
    para Librerías Nacionales Estandarizado</p>
  </div>
  <a href="catalogo">
    <button class="btn-index">Ingresar</button>
  </a>
</div>
```

Desarrollo de la aplicación

Implementando un ABM básico para después poder expandir en funcionalidades adicionales

SALNE

Inicio

Autores

Géneros

Libros

Bienvenido al Sistema de Administración para Librerías Nacionales Estandarizado

Para comenzar seleccione una de las categorías en el panel lateral izquierdo.

Ver elementos y realizar funciones de ABM

SALNE

Inicio

Autores

Géneros

Libros

Gestión de libros

Registrar nuevo libro

Operación de Alta

Título



Bringing back to modern life – Tonic Trouble



Renzo Pigliacampo - 1ra Edición - 2020 - Software

An overlook on JVS I-O emulation and implementation



Renzo Pigliacampo - 2da Edición - 2020 - Software

Portada

Autor

Edición

Año

Género(s)

SALNE

Inicio

Autores

Géneros

Libros

Detalles del libro

Título: Bringing back to modern life – Tonic Trouble

Descripción: Develop a solution to give new life to an old title in modern times.

Autor: [Renzo Pigliacampo](#)

Edición: 1ra Edición

Año: 2020

Género: [Software](#)

ISBN: 1234567890

Metadatos



Volver

Editar información

Eliminar libro

Modificar

Eliminar

Registrar nuevo libro

Título:

Descripción:

Autor:

Renzo Pigliacampo



Edición:

Año:

Género:

Software



ISBN:

Imagen:

Seleccionar archivo

No se eligió archivo

Todos los campos son validados tanto en el lado del cliente (HTML) como en el servidor (JS + Express-Validator)

Cancelar

Añadir

¿Y ahora qué?

- Añadir sistema de usuarios (con distintos grados/roles/**permisos**)
- Incluyendo gestión de usuarios administradores y datos personales
- Implementar **sesiones** de usuario
- Implementar sistema de compras y biblioteca de usuario

Permisos

- A través de un atributo en el modelo del Usuario

- Delimita funcionalidades

Administrador

- Gestiona categorías y elementos

Cliente

- Puede comprar contenido
- Maneja una biblioteca personal con sus productos adquiridos



Sesiones

- Nos referimos a la capacidad de registrarse e ingresar al sistema con una identidad única, que nos permite almacenar información individual
- Además de implementar un sistema de sesiones persistente para el cliente, a través del uso de cookies
 - Para ello, en Express usamos el paquete `Express-Session`

```
// Instanciar el paquete Express-Session  
var session = require('express-session');
```

```
// Configurar los parámetros de sesión  
app.use(session({  
  secret: '55aMKLU$%kZ3iDa$YSV4Hk4XN!7wik',  
  resave: false,  
  saveUninitialized: false,  
  store: MongoStore.create({ mongoUrl: mongoDB })  
}));
```

```
// Habilitar el acceso de la sesión para toda la aplicación  
app.use(function (req, res, next) {  
  res.locals.SesionActual = req.session;  
  next();  
});
```

SALNE

Inicio

Iniciar Sesión

Registrarse


Iniciar sesión

Usuario:

Contraseña:

CancelarIngresar

```
module.exports.iniciarSesionPost = function (req, res, next) {  
  UsuarioModel.findOne({ 'nombre': req.body.nombre, 'password': req.body.password }, function (err, usuario) {  
    if (usuario) {  
      req.session.id_usuario = usuario._id;  
      res.redirect('/catalogo');  
      return;  
    } else {  
      res.render('layout', { content: 'iniciarSesion', title: ' - Iniciar Sesión', invalid: 1 });  
      return;  
    }  
  });  
};
```



Compras

- Realizar un sistema de compras, donde el usuario pueda añadir artículos a un “carrito”, y luego realizar la compra.

SALNE


Inicio

Cuenta

Carrito

Cerrar Sesión

Carrito



An overlook on JVS I-O emulation and implementation

Renzo Pigliacampo - 2da Edición - 2020 - Software

Realizar Compra

- Para ello volvemos a hacer uso de las cookies, haciendo el carro de compras persistente para la sesión.

```
module.exports.anadirLibro = function (req, res, next) {  
  var carritoUsuario = [];  
  if (req.cookies.carrito)  
    carritoUsuario = req.cookies.carrito;  
  if (!carritoUsuario.includes(req.params.id)) {  
    carritoUsuario.push(req.params.id);  
    res.cookie('carrito', carritoUsuario).send();  
  }  
  res.redirect('/usuarios/' +  
    res.locals.UsuarioActual._id + '/carrito');  
}
```

```
module.exports.realizarCompra = function (req, res, next) {  
  var Usuario = res.locals.UsuarioActual;  
  for (libroCarrito of req.cookies.carrito) {  
    Usuario.libros.push(libroCarrito);  
  }  
  UsuarioModel.findByIdAndUpdate(res.locals.UsuarioActual._id,  
    Usuario, {}, function (err) {  
    if (err) return handleError(err);  
    res.clearCookie('carrito');  
    res.redirect('/catalogo');  
  });  
}
```

Biblioteca

- Un lugar donde se puedan ver los artículos adquiridos

SALNE


[Inicio](#)

[Cuenta](#)

[Carrito](#)

[Cerrar Sesión](#)

Biblioteca



Bringing back to modern life – Tonic Trouble

Renzo Pigliacampo - 1ra Edición - 2020 - [Software](#)

Volver

Publicar

- Para distribuir una aplicación es necesario moverse del ambiente de ejecución local y trasladar el proyecto a un servidor web online (en este caso, [Heroku](#))
- Este servidor debe ser compatible con las tecnologías que estamos utilizando (en este caso, [Node.js](#))
- Esto incluye sólo la parte de hosting (procesamiento y almacenamiento estático), la base de datos se ubica en un servidor dedicado (en este caso, [MongoDB Atlas](#))

Proyecto Finalizado

Inicio

Autores

Géneros

Libros

Iniciar Sesión

Registrarse

Bienvenido al Sistema de Administración para Librerías Nacionales Estandarizado

Para comenzar seleccione una de las categorías en el panel lateral izquierdo.

Últimos libros añadidos al catálogo:



Bringing back to modern life – Tonic Trouble

Renzo Pigliacampo



An overlook on JVS I-O emulation and implementation

Renzo Pigliacampo

Imagen no disponible

Prueba

Renzo Pigliacampo

¿En que se puede mejorar?

(Y desarrollar otra respuesta que no sea “en todos los aspectos”)

- Principalmente, añadir opciones de búsqueda y orden en todos los lugares que sea posible y necesario
 - Una interfaz más completa e interactiva (asumiendo una escala mayor en el contenido y funcionalidades de la aplicación)
-
- Separar la parte administrativa de la experiencia del usuario
 - Ampliar la parte administrativa, principalmente a través de la clasificación por organización, considerando sus atributos únicos
 - En base al punto anterior, ofrecer ventas de artículos físicos, manejando un sistema de stock institucional de forma individual

¿De qué otros servicios se puede estudiar y aprender?



The MIT Press



Enseñanzas y conocimientos que dejó este proyecto

- Desarrollo completo de una aplicación web Full Stack, utilizando metodologías y tecnologías modernas
- Implementación de un proyecto de software a un producto real
- Escala de tal aplicación, y la necesidad de un grupo de trabajo para cumplir con los requerimientos y tiempos establecidos

Fin

Renzo Pigliacampo - 2021