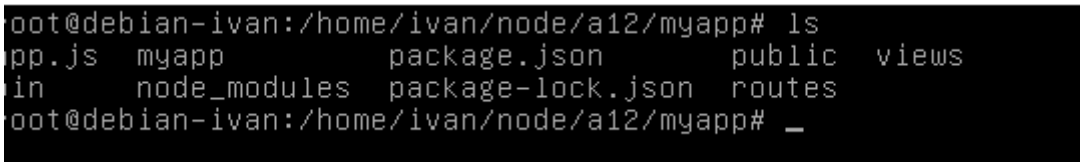
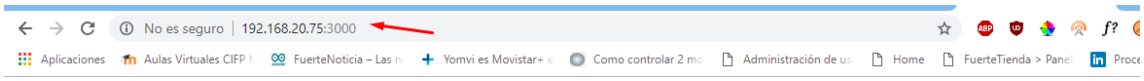
1. Genera el esqueleto de la aplicación. Sólo tienes que invocar el generador en la línea de comandos con un nombre para el nuevo proyecto, especificando también el motor de plantillas y el generador de CSS a utilizar.

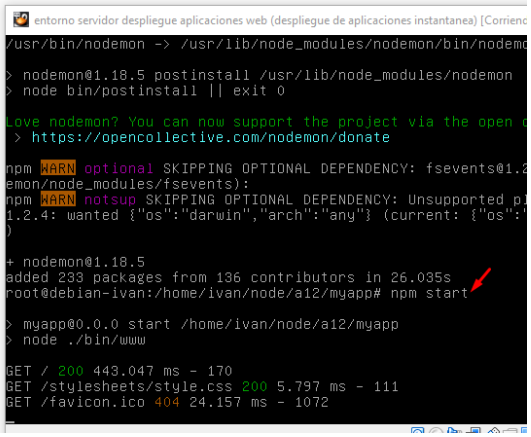
Creamos nuestro Express application generator



Especificamos el motor de plantilla

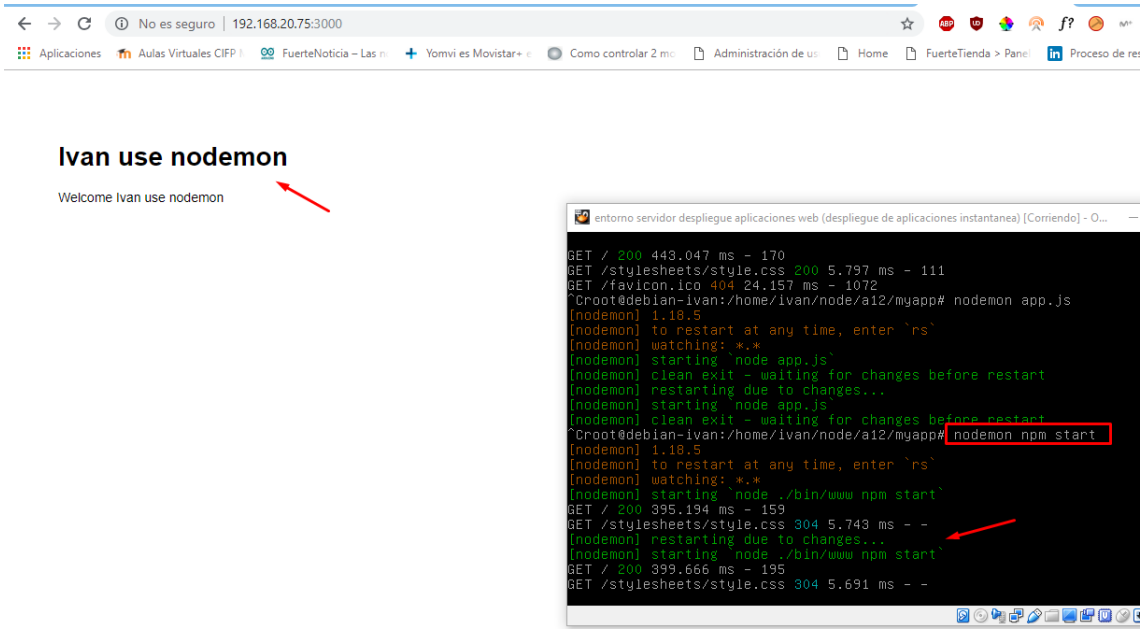
express --view=pug myapp





3En principio, cualquier cambio que hagas en la app no será visible hasta que reinicies el servidor, lo cual se vuelve tedioso. El módulo nodemon permite automatizar el reinicio del servidor cuando sea necesario. Busca cómo utilizarlo e intégralo en el proyecto.

Probamos el módulo nodemon (nodemon npm start)

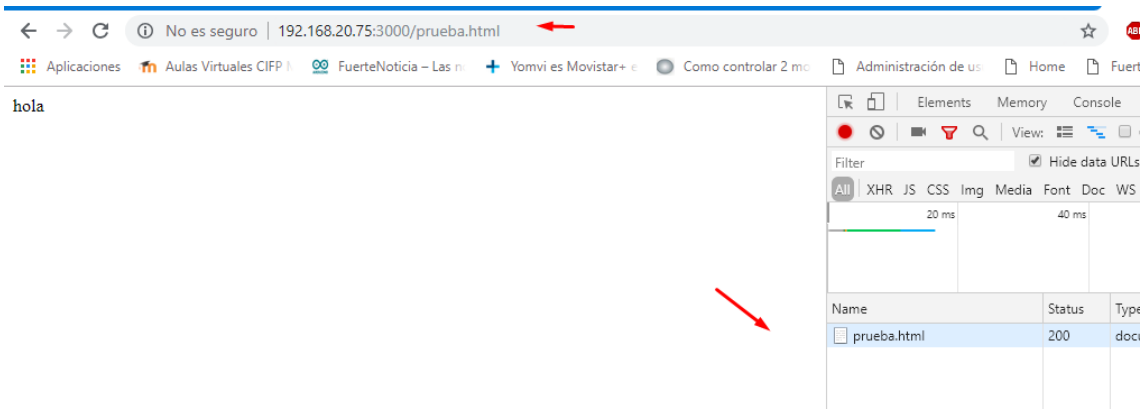


4. El archivo package.json define las dependencias de la aplicación. ¿Qué módulos instala por defecto el Express Generator y para qué sirve cada uno?

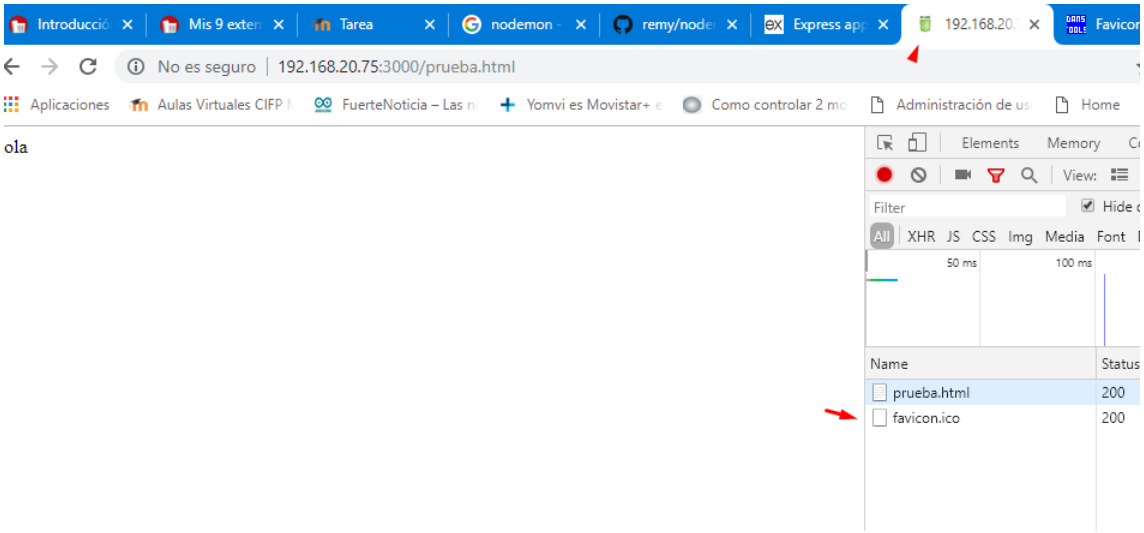
"dependencies": {
  "cookie-parser": "^1.4.3",
  "debug": "^2.6.9",
  "express": "^4.16.4",
  "http-errors": "^1.6.3",
  "morgan": "^1.9.1",
  "pug": "^2.0.0-beta11"
 }

Cookie-parser es un analizador de cookies, donde analiza las cookies adjuntas al objeto de solicitud del cliente.
Debug: es un depurador de código js
Express: Es un framework muy usado en NodeJs para crear aplicaciones web y Apis Rest de forma más rápida.
http-errors: Módulo que sirve para generar errores http.
Morgan: Genera un log de solicitudes de un cliente.
Pug: framework que permite generar código HTML de manera simple

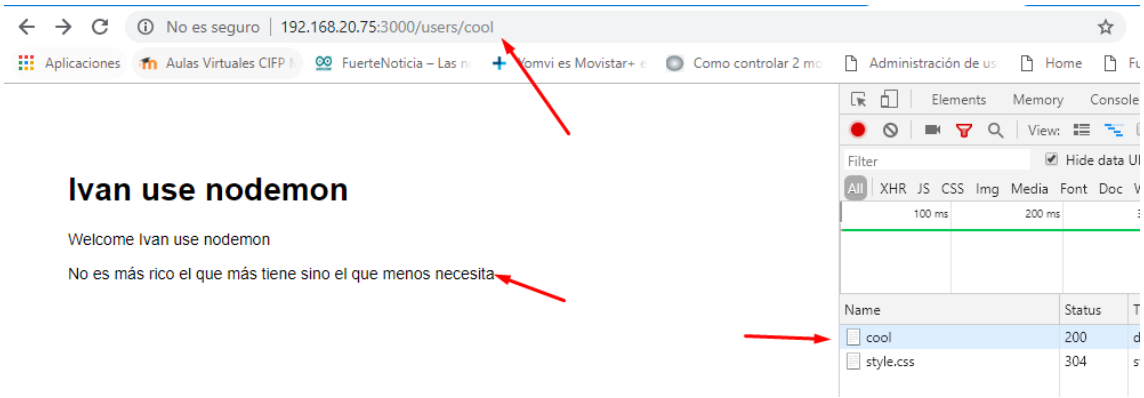
5. Crea un fichero estático prueba.html en el directorio public y comprueba que puedes acceder a él con el navegador.



6. Crea un favicon para tu aplicación y comprueba que funciona.



7. Crea una nueva ruta en /routes/users.js que muestre el texto "No es más rico el que más tiene sino el que menos necesita"" en la URL /users/cool/. Pruébala.



Creamos la ruta /cool puesto que ya se ha creado en app.js la ruta “users”

App.js (var usersRouter = require('./routes/users')

```
var express = require('express');
var router = express.Router();

/* GET users listing. */
router.get('/', function(req, res, next) {
  res.send('respond with a resource');
});

router.get('/cool/', function(req, res, next) {
  res.render('index',{ title: 'Ivan use nodemon' ,message:'No es más rico el que más tiene sino el que menos necesita'});
});


module.exports = router;
```

8. El fichero app.js es el script principal de la aplicación. La secuencia típica para crear una aplicación en Node se muestra a continuación. Localiza cada uno de estos pasos en el fichero app.js y añade un comentario al principio de cada sección:

Sección “Imports”

```
//IMPORTS
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
```

Sección “Routes” e “Instantations”

```
//ROUTES
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');

//INSTANTATIONS
var app = express();
```

Sección “Configurations”

```
// configurations
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
```

Sección “Middleware”

```
//Middleware
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
```

Sección “Routes”

```
//ROUTES
app.use('/', indexRouter);
app.use('/users', usersRouter);
```

Sección “error handlers”

```
// ERROR HANDLERS
app.use(function(req, res, next) {
  next(createError(404));
});


app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});
```

Sección “Server bootup or server export”

```
//Server bootup or server export
module.exports = app;
```