PROYECTO 4: APLICACIÓN DE GESTIÓN DE NOTAS

TIEMPO ESTIMADO: 150 MINUTOS

DESCRIPCIÓN

- *Generación de una lista de notas con almacenamiento mediante json y ficheros*

OBJETIVOS

- Repaso de lo visto en proyectos anteriores
- Usar objetos JSON
  - Serializar
  - Deserializar

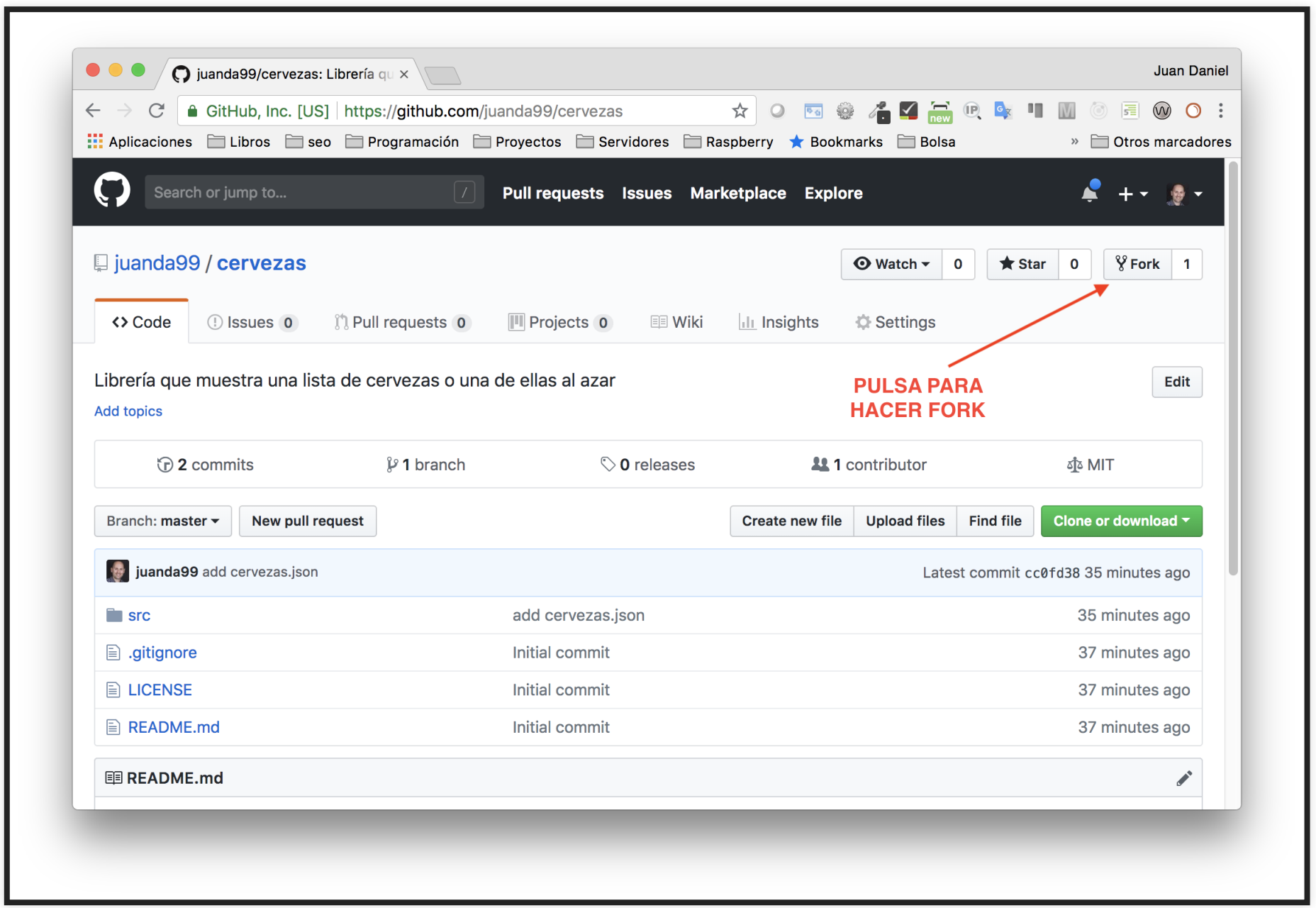
EMPEZAMOS PROYECTO

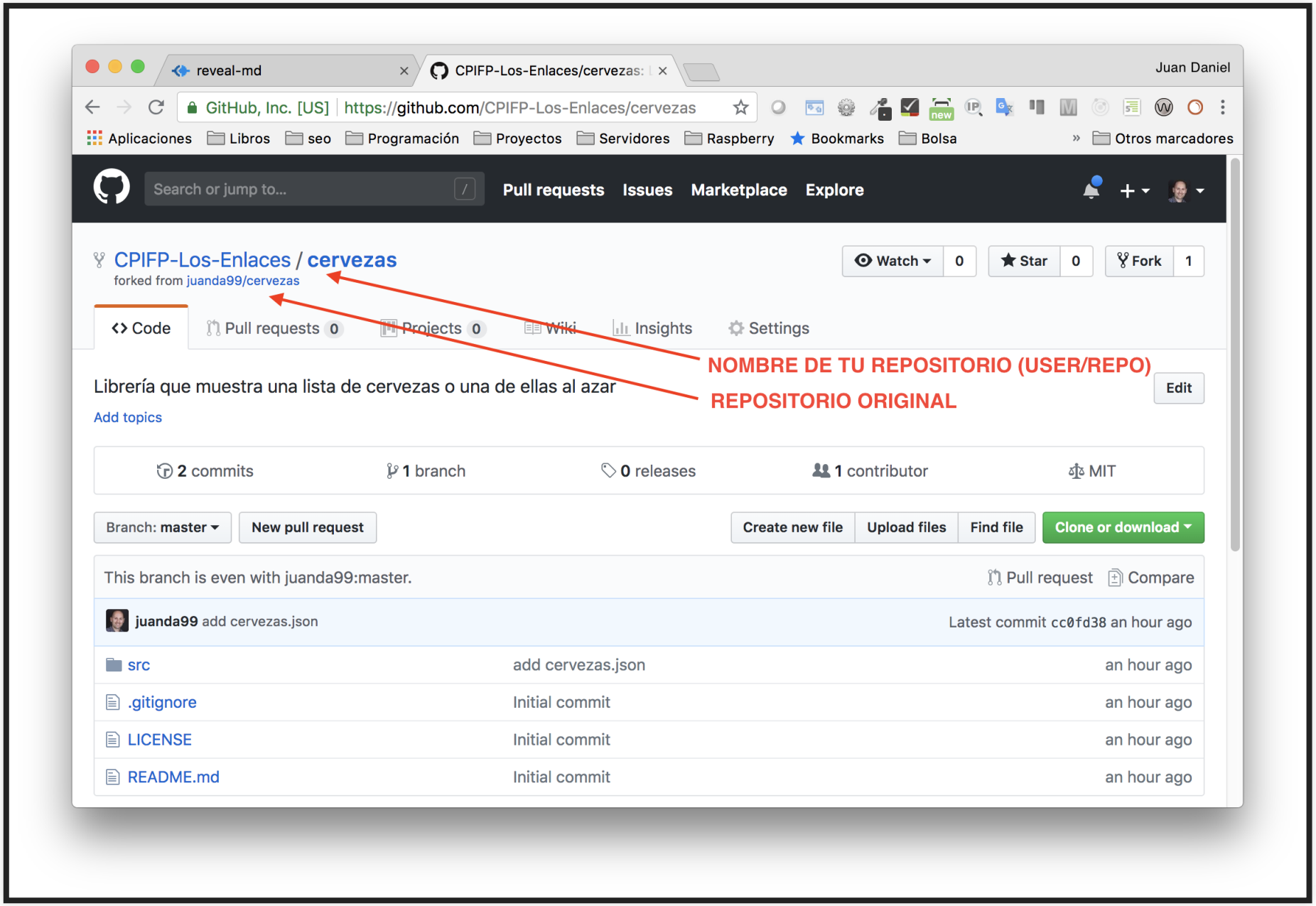
PASOS PREVIOS

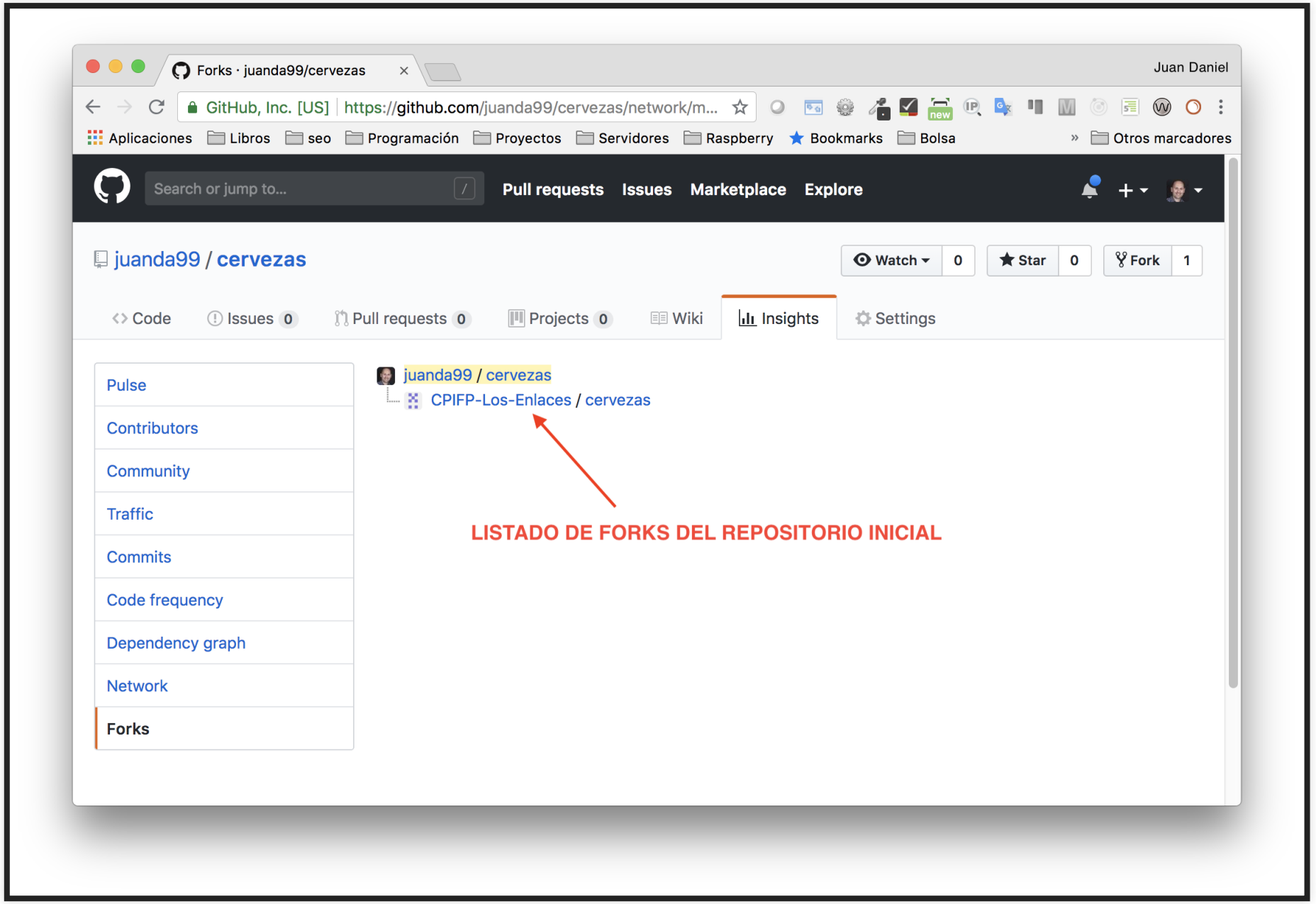
- Pasos similares a la práctica anterior
  - Fork del proyecto
  - Clone al equipo local
  - Configuración del linter
- Vemos capturas de la práctica anterior

CREAR REPOSITORIO EN GITHUB

- Realizamos un fork de mi proyecto
  - Así tendremos el fichero para la práctica
  - .gitignore correctamente configurado
  - Puedo hacer seguimiento de vuestros desarrollos


juanda99/cervezas: Librería qu
Juan Daniel
GitHub, Inc. [US] https://github.com/juanda99/cervezas
Aplicaciones
Libros
seo
Programación
Proyectos
Servidores
Raspberry
Bookmarks
Bolsa
Otros marcadores
Search or jump to...
Pull requests
Issues
Marketplace
Explore
juanda99 / cervezas
Watch 0
Star 0
Fork 1
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Librería que muestra una lista de cervezas o una de ellas al azar
Add topics
PULSA PARA
HACER FORK
Edit
2 commits
1 branch
0 releases
1 contributor
MIT
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
juanda99 add cervezas.json
Latest commit cc0fd38 35 minutes ago
src
add cervezas.json
35 minutes ago
.gitignore
Initial commit
37 minutes ago
LICENSE
Initial commit
37 minutes ago
README.md
Initial commit
37 minutes ago
README.md

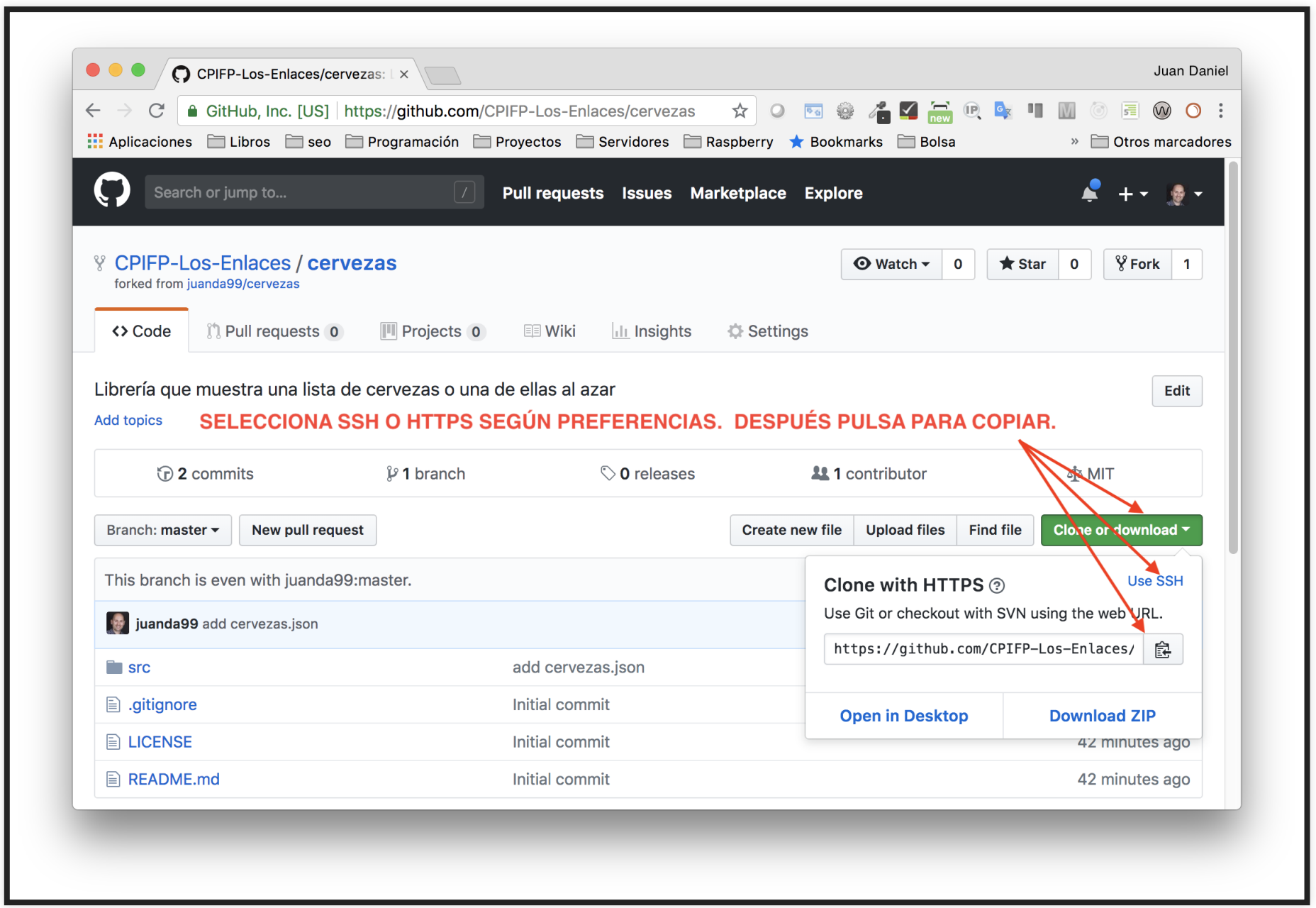

reveal-md
CPIFP-Los-Enlaces/cervezas:
Juan Daniel
GitHub, Inc. [US] https://github.com/CPIFP-Los-Enlaces/cervezas
Aplicaciones Libros seo Programación Proyectos Servidores Raspberry Bookmarks Bolsa Otros marcadores
Search or jump to...
Pull requests Issues Marketplace Explore
CPIFP-Los-Enlaces / cervezas
forked from juanda99/cervezas
Watch 0 Star 0 Fork 1
Code Pull requests 0 Projects 0 Wiki Insights Settings
NOMBRE DE TU REPOSITORIO (USER/REPO)
REPOSITORIO ORIGINAL
Librería que muestra una lista de cervezas o una de ellas al azar
Edit
Add topics
2 commits 1 branch 0 releases 1 contributor MIT
Branch: master New pull request
Create new file Upload files Find file Clone or download
This branch is even with juanda99:master.
Pull request Compare
juanda99 add cervezas.json
Latest commit cc0fd38 an hour ago
src add cervezas.json an hour ago
.gitignore Initial commit an hour ago
LICENSE Initial commit an hour ago
README.md Initial commit an hour ago


Forks · juanda99/cervezas
Juan Daniel
GitHub, Inc. [US] https://github.com/juanda99/cervezas/network/m...
Aplicaciones
Libros
seo
Programación
Proyectos
Servidores
Raspberry
Bookmarks
Bolsa
Otros marcadores
Search or jump to...
Pull requests
Issues
Marketplace
Explore
juanda99 / cervezas
Watch 0
Star 0
Fork 1
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Pulse
Contributors
Community
Traffic
Commits
Code frequency
Dependency graph
Network
Forks
juanda99 / cervezas
CPIFP-Los-Enlaces / cervezas
LISTADO DE FORKS DEL REPOSITORIO INICIAL

CLONAR REPOSITORIO A LOCAL

```
git clone <url proyecto>
```

- La url la copiamos del repo de GitHub (ver captura)
  - ssh normalmente en linux (necesitas importar la clave pública a GitHub)
  - https normalmente en windows / mac


CPIFP-Los-Enlaces/cervezas:
Juan Daniel
GitHub, Inc. [US] https://github.com/CPIFP-Los-Enlaces/cervezas
Aplicaciones Libros seo Programación Proyectos Servidores Raspberry Bookmarks Bolsa Otros marcadores
Search or jump to...
Pull requests Issues Marketplace Explore
CPIFP-Los-Enlaces / cervezas
forked from juanda99/cervezas
Watch 0 Star 0 Fork 1
Code Pull requests 0 Projects 0 Wiki Insights Settings
Librería que muestra una lista de cervezas o una de ellas al azar
Edit
Add topics
SELECCIONA SSH O HTTPS SEGÚN PREFERENCIAS. DESPUÉS PULSA PARA COPIAR.
2 commits 1 branch 0 releases 1 contributor MIT
Branch: master New pull request
Create new file Upload files Find file Clone or download
This branch is even with juanda99:master.
juanda99 add cervezas.json
src add cervezas.json
.gitignore Initial commit
LICENSE Initial commit 42 minutes ago
README.md Initial commit 42 minutes ago
Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/CPIFP-Los-Enlaces/
Open in Desktop
Download ZIP

CREAR PROYECTO

```
cd <url proyecto>
npm init
```

- package_name debe ser único: **no puede haber dos proyectos con el mismo nombre en npm**
- El *entry-point* lo pondremos en *src/index.js*, así separaremos nuesto código fuente de los tests.
- El resto de parámetros con sus valores por defecto
- ¡Ya tenemos nuestro **package.json** creado!

ESTILO DE CÓDIGO

- Puede que colabore más gente en nuestra librería
  - Queremos un estilo uniforme
- Y si nos detecta fallos mejor
- Instalaremos eslint (*D* o *--save-dev*)

```
npm i -D eslint
```

CONFIGURACIÓN DE ESLINT

```
$ node_modules/.bin/eslint --init
? How would you like to configure ESLint?
  Use a popular style guide
? Which style guide do you want to follow?
  Standard
? What format do you want your config file to be in?
  JSON
? Would you like to install them now with npm?
  Yes
```

EMPEZAMOS NUESTRA APP

- Creamos *app.js* como punto de entrada de nuestra aplicación
- Creamos un módulo específico para nuestras notas
  - Servirá para añadir o quitar notas

```
  console.log('Modulo de notas cargado')
```

- Lo cargamos en nuestra aplicación (*app.js*)

```
console.log('Aplicación de notas arrancada.')
const notes = require('./notes')
```

ESTRUCTURA MÓDULO NOTES.JS

- module.exports debe ser un objeto

```
module.exports.addNote = function () {
  console.log('Nueva nota')
  return 'Nueva nota'
}
```

- *app.js* hace el require del módulo y lo almacena en una variable

```
const notes = require('notes')
const notaAñadida = notas.addNote()
```

- ¿Sabrías crear la función removeNote?

SOLUCIÓN

- Notes.js:

```
module.exports.removeNote = function (id) {
  console.log(`Nota borrada con id=${id}`)
  return 'Nota borrada'
}
```

- app.js

```
const notes = require('notes')
const notaAñadida = notas.addNote()
const notaEliminada = notas.removeNote(5)
```

REPASO ES6

- Arrow functions
- Object Literal Property Value Shorthand

ARROW FUNCTIONS

```
const sumarNumeros =  function (a, b) { return a + b }
const sumarNumerosES6 = (a, b) => { return a + b }
const sumarNumerosES6bis = (a, b) => a + b
```

SINTAXIS ARROW FUNCTIONS

- Sintaxis general:

```
(param1, param2, …, paramN) => { statements }
```

- Si solo hay una expresión, se pueden omitir las llaves:

```
(param1, param2, …, paramN) => expression
```

- Si no hay parámetros, paréntesis obligatorios:

```
() => { statements }
```

- Si solo hay un parámetro, parámetros opcionales:

```
singleParam => { statements }
```

ES6: OBJECT LITERAL PROPERTY VALUE SHORTHAND

- Antes (ES5):

```
function createMonster(name, power) {
  return { type: 'Monster', name: name, power: power }
}
```

- Ahora (ES6):

```
function createMonster(name, power) {
  return { type: 'Monster', name, power }
}
```

- Y con arrow functions:

```
const createMonster = (name, power) =>  ({ type: 'Monster', na
}
```

- Esta opción no sería correcta:
  - El compilador piensa que hay un block statement, y no un objeto
  - Retorna *undefined*

```
const createMonster = (name, power) =>  { type: 'Monster', nam
}
```

SOLUCIÓN CON ES6

```
const addNote =  () => {
  console.log('Nueva nota')
  return 'Nueva nota'
}
const removeNote =  id => {
  console.log(`Nota borrada con id: ${id}`)
  return 'Nota borrada'
}
module.exports = {
  addNote,
  removeNote
}
```

INSTALAR MÓDULOS DE TERCEROS

- Necesitamos guardar la referencia a los módulos que guardamos:

```
# este comando ya lo hemos ejecutado previamente
# y tenemos la dependencia eslint guardada
npm init # para tener un package.json & package-lock.json
```

- Configuraremos el script start para que ejecute nuestra aplicación

GUARDAR MÓDULOS DE TERCEROS

- Se guardan en node_modules
- No se deben sincronizar con GitHub (usar *.gitignore*)
- Usar *npm install* para descargarlos

NODEMON

- Es un comando relativo al CLI
- Instalamos

```
npm install --save-dev nodemon # o más corto:
npm i -D nodemon
```

- Cambiaremos en script *npm start* para utilizarlo

INPUT DE UNA APLICACIÓN

- Una aplicación puede obtener el input de distintas formas:
  - Práctica socket.io
    - Mediante websockets
  - Práctica API:
    - Parámetros en URL
    - Parámetros en el header
  - Práctica actual
    - Mediante línea de comandos

INPUT DE CLI

- Los podemos obtener de *process*:

```
console.log(process.argv)
```

  - *argv[0]* es el ejecutable de node
  - *argv[1]* es *app.js*
  - *argv[2]* es el parámetro

EJERCICIO

- Recoger los distintos input que puede recibir la aplicación *if-else* :
  - add: para añadir una nota
  - list: para sacar la lista de notas
  - read: para leer una nota en particular
  - remove: para eliminar una nota
  - cualquier otro caso: avisar de que el comando no se reconoce

SOLUCIÓN CON IF-ELSE

```
console.log('Aplicación de notas arrancada.')
const notes = require('./notes')

var command = process.argv[2]
console.log('Command: ', command)
console.log(process.argv)

if (command === 'add') {
  console.log('Añadiendo nueva nota')
} else if (command === 'list') {
  console.log('Listado de todas las notas')
} else if (command === 'read') {
  console.log('Leyendo notas')
} else if (command === 'remove') {
  console.log('Borrando nota')
```

SOLUCIÓN CON SWITCH

```
console.log('Aplicación de notas arrancada.')
// const notes = require('./notes')

var command = process.argv[2]
console.log('Command: ', command)
console.log(process.argv)

switch (command) {
  case 'add':
    console.log('Añadiendo nueva nota')
    break
  case 'list':
    console.log('Listado de todas las notas')
    break
  case 'read':
```

PROBLEMÁTICA SOLUCIÓN

- Los parámetros pueden llevar argumentos extra:

```
node app.js remove --title="Nota 1"
```

- Los argumentos extra pueden escribirse de múltiples maneras:

```
node app.js remove --title="Nota 1"
node app.js remove --title "Nota 1"
node app.js remove -t="Nota 1"
node app.js remove -t "Nota 1"
node app.js remove --author="Peter"
```

- Nos interesa recibir los argumentos de una forma sencilla
  - Sin tener que crear un parser
  - Si nuestra entrada fuera un objeto sería más sencillo
- Usaremos yargs

USO DE YARGS

- Instalación:

```
npm i -S yargs
```

- Comparar salidas:

```
console.log('Aplicación de notas arrancada.')
const notes = require('./notes')
// const yargs = require('yargs')
// const argv = yargs.argv
const { argv } = require('yargs')

// var command = process.argv[2]
var command = argv._[0]
console.log('Command: ', command)
console.log(process.argv)
console.log('Yargs', argv)
```

EJECUCIÓN

- Comprueba que la salida de estos comandos no aportan grandes diferencias si usamos *yargs* o *process*:

```
node app add
node app add encrypted
```

- Pero si usamos parejas key-value, la cosa cambia:

```
node app add --title="test"
node app add --title "test"
node app add --title test
```

AÑADIR UNA NOTA

- Debemos llamar a la función addNote con dos parámetros: title y body:

```
if (command === 'add') {
  notes.addNote(argv.title, argv.body);
}
```

- La nueva función addNote sería:

```
const addNote = (title, body) => {
  console.log('Nota añadida: ', title, body)
}
```

- Probamos:

```
node app.js add --title=test1 --body="Esta es mi nota de prueb
```

EJERCICIO PROCESAR PARÁMETROS

- Implementa las funciones relativas a las notas, según el siguiente código:

```
if (command === 'add') {
  notes.addNote(argv.title, argv.body);
} else if (command === 'list') {
  notes.getAll();
} else if (command === 'read') {
  notes.getNote(argv.title);
} else if (command === 'remove') {
  notes.removeNote(argv.title);
} else {
  console.log('Comando desconocido');
}
```

SOLUCIÓN PROCESO DE PARÁMETROS

```
console.log('Módulo de notas cargado')
const addNote = (title, body) => {
  console.log('Nota añadida: ', title, body)
}
const getAll = () => {
  console.log('Obtenidas todas las notas')
}
const getNote = (title) => {
  console.log('Obtenida nota: ', title)
}
const removeNote = (title) => {
  console.log('Nota borrada', title)
}
module.exports = {
  addNote,
```

ALMACENAMIENTO DE NOTAS

- Utilizamos objetos para trabajar en nuestro código:

```
const nota1 = {
  title: titulo1,
  body: body1
}
```

- Si se inserta una nueva nota, lo suyo sería guardarla en algún sitio
  - Array de objetos
  - Ficheros / bbdd
  - API

SERIALIZACIÓN

- Para almacenar en ficheros o envio via API debemos utilizar *strings* o *buffers*.
- Se hace necesario serializar nuestros objetos de notas.
- Haremos un par de ejercicios para ver si sabemos:
  - Convertir un JSON a string (escritura de nota)
  - Parsear un string a JSON (lectura de nota)

EJERCICIO SERIALIZACIÓN OBJETO

- Convierte el siguente objeto a un string y muestra su tipo y valor por consola:

```
var persona = {
  nombre: 'Pepe',
  edad: '25'
}
```

- Se utiliza *JSON* para hacer la conversión
- Se utiliza *typeof* para ver el tipo de datos

SOLUCIÓN SERIALIZACIÓN

```
const persona = {
  nombre: 'pepe',
  edad: 25
}
console.log(persona)
const serializedPersona = JSON.stringify(persona)
console.log(serializedPersona)
console.log(typeof persona)
console.log(typeof serializedPersona)
```

EJERCICIO DESERIALIZAR OBJETO

- Convierte el string siguiente a JSON y obten la edad:

```
var personaString = '{"nombre": "Pepe", "edad": 25}'
```

SOLUCIÓN DESERIALIZACIÓN

```
var personaString = '{"nombre": "Pepe", "edad": 25}'
var persona = JSON.parse(personaString)
console.log(persona.edad)
```

IMPLEMENTAR INSERCIÓN DE NOTAS

- Leemos fichero con lista de notas
- Añadimos la nota recibida
- Las volvemos a llevar al fichero
- Serializamos/deserializamos según proceda

```
const fs = require('fs')

const addNote = (title, body) => {
  let notes = []
  const note = {
    title,
    body
  }

  const notesString = fs.readFileSync('notes-data.json')
  notes = JSON.parse(notesString)

  notes.push(note)
  fs.writeFileSync('notes-data.json', JSON.stringify(notes))
}
```

PROBAR INSERCIÓN NOTAS

- Saltará un error porque el fichero *notes-data.json* no existe
- Soluciones:
  - Crear previamente el fichero
  - Gestionar la excepción desde el código

```
node app.js add --title='titulo' --body='texto de la nota'
```

GESTIÓN DE LA EXCEPCIÓN

```
const fs = require('fs')

const addNote = (title, body) => {
  let notes = []
  const note = {
    title,
    body
  }

  try {
    const notesString = fs.readFileSync('notes-data.json')
    notes = JSON.parse(notesString)
  } catch (error) {

  }
```

GESTIÓN DE DUPLICADOS

```
const fs = require('fs')

const addNote = (title, body) => {
  let notes = []
  const note = {
    title,
    body
  }

  try {
    const notesString = fs.readFileSync('notes-data.json')
    notes = JSON.parse(notesString)
  } catch (error) {}
  const duplicateNotes = notes.filter(note => note.title === t
```

FUNCIONES PARA REFACTORIZAR CÓDIGO

- Leer y guardar notas se va a hacer varias veces

```
var fetchNotes = () => {
  try {
    var notesString = fs.readFileSync('notes-data.json');
    return JSON.parse(notesString);
  } catch (e) {
    return [];
  }
}

var saveNotes = (notes) => {
  fs.writeFileSync('notes-data.json', JSON.stringify(notes));
}
```

EJERCICIO REFACTORIZAR

- Utiliza las funciones anteriores para refactorizar el código
- Devuelve la nota creada a *app.js* de modo que la muestre por consola

SOLUCIÓN EJERCICIO REFACTORIZAR

- notes.js:

```
var addNote = (title, body) => {
  var notes = fetchNotes()
  var note = {
    title,
    body
  }
  var duplicateNotes = notes.filter((note) => note.title === t

  if (duplicateNotes.length === 0) {
    notes.push(note)
    saveNotes(notes)
    return note
  }
}
```

- app.js:

```
if (command === 'add') {
  var note = notes.addNote(argv.title, argv.body)
  if (note) {
    console.log('Nota creada')
    console.log(`\tTítulo: ${note.title}`)
    console.log(`\tTexto: ${note.body}`)
  } else {
    console.log('Ya existe una nota con este título');
  }
} else if (command === 'list') {
  notes.getAll()
} else if (command === 'read') {
  notes.getNote(argv.title)
} else if (command === 'remove') {
  notes.removeNote(argv.title)
```

IMPLEMENTAR REMOVENOTE

- Basándote en la función addNote, implementa la función removeNote
- Muestra por consola el resultado (fichero app.js)

```
var removeNote = (title) => {
  // obtener notas
  // utilizar filter para eliminar nota según title
  // guardar notas

  return true // or false
}
```

SOLUCIÓN REMOVENOTE

- notes.js:

```
var removeNote = (title) => {
  var notes = fetchNotes()
  var filteredNotes = notes.filter((note) => note.title !== ti
  saveNotes(filteredNotes)
  return notes.length !== filteredNotes.length
}
```

- app.js

```
else if (command === 'remove') {
  var noteRemoved = notes.removeNote(argv.title)
  var message = noteRemoved ? 'Nota borrada' : 'Nota no encont
  console.log(message)
}
```

IMPLEMENTAR LEER UNA NOTA

- Implementa el método readNote en función del título de la nota
- Si usas la función filter, ten en cuenta que siempre devuele un array
- Refactoriza el código usando la función logNote

```
var logNote = (note) => {
  console.log(`\tTítulo: ${note.title}`)
  console.log(`\tTexto: ${note.body}`)
}
```

SOLUCIÓN LEERNOTA

- note.js:

```
var getNote = (title) => {
  var notes = fetchNotes()
  var filteredNotes = notes.filter((note) => note.title === ti
  return filteredNotes[0]
}
```

- Recuerda exportar la función para utilizarla en app.js

- app.js:

```
else if (command === 'read') {
  var note = notes.getNote(argv.title)
  if (note) {
    console.log('Nota encontrada:')
    notes.logNote(note)
  } else {
    console.log('Nota no entontrada')
  }
}
```

- utiliza logNote en otras partes del código (al añadir nota)

EJERCICIO: LISTADO DE NOTAS

- Implementa la función getAll para obtener todas las notas
- Muestra las notas por consola desde *app.js*

SOLUCIÓN:

- notes.js:

```
var getAll = () => {
  return fetchNotes()
}
```

- app.js:

```
else if (command === 'list') {
  var allNotes = notes.getAll();
  console.log(`Mostrando ${allNotes.length} notas.`);
  allNotes.forEach((note) => notes.logNote(note));
}
```

Y MUCHO MÁS

- yargs avanzado, parámetros requeridos...
  - Ver documentación
- Ejecución al estilo *bash script*:

```
#!/usr/bin/env node
```

¿CONTINUAMOS?

- Siguiente proyecto:
  - Servidor Web mediante Express.js