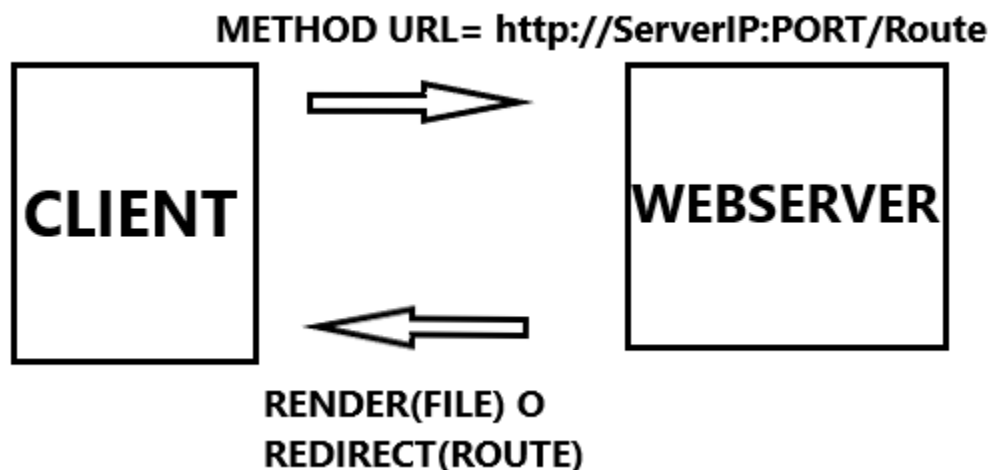


## TUTORIAL E FUNZIONAMENTO HOME.JS

- **OBIETTIVO DEL PROGRAMMA**

- **Realizzare una comunicazione Client → Server**
  - **BROWSER → WEBSERVER**
- **Collegare nella stessa rete i 2 HOST**
  - Tramite **WIFI** oppure **ROUTER** con **cellulare**
- **Creazione webserver http in ascolto su IP:PORT**
  - Spiegato in seguito con apposite foto
- **Richiesta di connessione da parte del Client**
  - Crea l'oggetto session nel Database
- **Registrazione e Login per accedere alle risorse**
  - Dati inseriti nel Database e verificati
- **Accesso alle risorse della presentazione**
  - Una volta autenticati tramite Login



- **INSTALLARE NODEjs** [DOWNLOAD](#)
  - **SOLO LATO SERVER**
  - Permette di realizzare delle webapplication di tipo Client Server, usando come linguaggio di programmazione JavaScript, il quale solitamente è lato Client, ma usando il framework NODE è possibile usarlo anche lato server
- **SCARICARE IL PROGETTO SERVER DAL SEGUENTE LINK** [DOWNLOAD](#)
  - Per scaricare il file si possono usare 2 metodi
    - Download della cartella in **formato zip** dal browser
    - “**git clone URL**” nella cartella desiderata. **Configurazione git** in fondo...
      - URL si ottiene aprendo la tendina verde con scritto Code e copiando il testo suggerito come URL. Deve terminare con **.git!!!**

master 2 branches 0 tags

Go to file Add file Code

This branch is 18 commits ahead, 1 commit behind main. Pull request Compare

DamiFuma02 FINALE db524db 22 ore fa 18 commits		
html-files	FINALE	22 ore fa
models	Modifica PCTO	l'altro ieri
public/immagini	FINALE	22 ore fa
routes	Aggiornamento sessione e FINE	ieri
views	FINALE	22 ore fa
.gitattributes	aggiuo ofile .gitattributes	3 giorni fa
.gitignore	rimozione package	3 giorni fa
APPTUTORIAL.pdf	Aggiornamento Tutorial.pdf	23 ore fa
HOME.js	Aggiornamento TUTORIALejs	23 ore fa
OFFLINE.pptx	Aggiornamento Sitografia	l'altro ieri
README.md	COMMIT COMPLETO	3 giorni fa

Verificare in alto a SX che sia selezionato il “MASTER” e non il “MAIN”

In caso di problemi provare da questo link [download](#), oppure questo [download](#)

## CONFIGURAZIONE LATO SERVER

- I testi inclusi in “ ” indicano un comando da eseguire nel terminale
- START+R** → **cmd.exe** → “**node -v**”
  - Per visualizzare la **versione installata**
  - Per controllare la **corretta installazione**
- Avviare un terminale nella cartella SERVER**, che **contiene** tutti i file del **programma**
- “**Npm init**” per **creare package.json**, nel quale gestire le **proprietà** e le **dipendenze** **necessarie** alla realizzazione del progetto finale.
  - Creazione script** “node app.js” e “nodemon app.js”.

```
{
  "name": "esame",
  "version": "1.0.0",
  "description": "server per esame 2021",
  "main": "HOME.js",
  "scripts": {
    "start": "node HOME.js",
    "devStart": "nodemon HOME.js"
  },
  "author": "Damiano Fumagalli",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.0.1",
    "connect-mongo": "^4.4.1",
    "connect-mongodb-session": "^2.4.1",
    "ejs": "^3.1.6",
    "express": "^4.17.1",
    "express-flash": "0.0.2",
    "express-session": "^1.17.2",
    "method-override": "^3.0.0",
    "mongoose": "^5.12.10",
    "passport-local-mongoose": "^6.1.0"
  },
  "devDependencies": {
    "dotenv": "^8.2.0",
    "nodemon": "^2.0.7"
  }
}
```

- **Dipendenze:** elenco di tutte le **librerie** installate
- **“Npm install ”** e di seguito il nome del pacchetto che si vuole installare, di seguito riportata la lista. Le documentazioni riguardo le librerie sono facilmente reperibili su internet.
  - **“connect-mongo-session”**: per gestire le informazioni sulla **sessione** e memorizzarle nel **database mongoDB connesso**
  - **“SESSION”**: per salvare le **info** degli **user** nelle **diverse pagine**
  - **“FLASH”**: per **inviare** messaggi in caso di **errori** durante il login
  - **“Method-override”**: per la **gestione** del **LOGOUT**
  - **“MONGOOSE”**: **interazione** con **MONGODB**
  - **“EXPRESS”**: **framework** per **web application**
  - **“BCRYPT”**: per **creare** una **comunicazione sicura** con **HASH** delle password.
    - Permette di gestire la **creazione** di **HASH** e **controllo** della **correttezza** di esso.
    - Quando si effettua l’operazione viene **salvata** nella **libreria** la **HASH** e la **SALT** che serve a generare la HASH.

```
//SALT è UN NUMERO CASUALE GENERATO DA UN'APPOSITA
//FUNZIONE IN MODO DA NON AVERE HASH UGUALI
//IN CASO DI VALOR DI TESTO UGUALI
const emailSalt = await bcrypt.genSalt()
const pswSalt = await bcrypt.genSalt()
//GENERAZIONE HASH DEI 2 TESTI INSERITI
const hashedEMAIL = await bcrypt.hash(email, emailSalt)
const hashedPSW = await bcrypt.hash(psw, pswSalt)
//SINTASSI FUNZIONE HASH()
hash(data: string | Buffer, saltOrRounds: string |
number): Promise<string>

The data to be encrypted.

@return — A promise to be either resolved with the encrypted
data salt or rejected with an Error
```

Queste funzioni sono eseguite all'interno del post di registrazione app.post("/")

Queste funzoni sono eseguite all'interno del post del login, per verificare la correttezza dei dati inseriti app.post("/login")

```
// valore booleano che è true se il compare è andato a buon fine
let sameEMAIL = await bcrypt.compare(req.body.email, user.email)
let samePSW = await bcrypt.compare(req.body.password, user.password)
//SINTASSI FUNZIONE compare()
compare(data: string | Buffer, encrypted: string):
Promise<boolean>

The data to be encrypted.

@return — A promise to be either resolved with the comparison
result salt or rejected with an Error
```

Si usano **funzioni asincrone** perché non seguono l’andamento classico delle altri, definite sincrone perchè sono ricorsive, ma vengono eseguite in “parallelo”. Con il parametro **await** si indica al **programma** di **aspettare** il **termine** della **funzione** e quindi il **return** del dato voluto.

Le **funzioni** di **generazione** SALT, HASH e la COMPARE **richiedono** molto **tempo**, a causa della grande sicurezza che abbiamo voluto ottenere generando randomicamente il SALT

- “**npm install –save-dev**”: dipendenze di development (sviluppo)
  - NODEMON: serve a riavviare il server ogni volta che si effettua un cambiamento
  - DOTENV: per gestire variabili d’ambiente all’interno di un file .env

Una volta installate tutte le Librerie si comincia con la configurazione del server

- **Creazione server http**

Grazie a express è possibile gestire facilmente le funzionalità che il server deve avere

```
const express = require("express");
const app = express();
const server = require("http").createServer(app)
PORT = process.env.PORT || 80
ip = "localhost"
server.listen(PORT, ip,
  () => {
    console.log(`SERVER IN ASCOLTO -> (${ip}:${PORT})`)
  });
```

- **Connessione al database MONGODB**

```
const mongoose = require("mongoose")
const session = require("express-session");
const MongoDBSession = require("connect-mongodb-session")(session)
//CONNESSIONE A MONGODB TRAMITE MONGOOSE
mongoPath = process.env.DATABASE_URL
mongoose.connect(mongoPath, {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
const db = mongoose.connection
db.on("error", error => console.error(error))
db.once("open", () => console.log("Connesso al database"))

//definisce dove vengono salvate le informazioni relative
//alla sessione in corso, derivante dal login
const DBSession = new MongoDBSession({
  uri: mongoPath,
  collection: "session"
})
```

- **MONGOOSE: MONGODB** interaction
  - **Creazione** database (free cluster on [mongodb](https://www.mongodb.com) )
  - **Accedere** a un database remoto, impostando un filtro che consenta l'accesso solo alla mia macchina con una whitelist 0.0.0.0
  - **DATABASE** = grande contenitore di dati
  - **COLLECTIONS** = sottocategorie del database nel quale andranno salvati i **documenti**. Perciò 1 database è formato da N collections
  - **DOCUMENTS**: file.json (oggetti) in questo caso gli username che possono avere accesso al mio esame
  - **SCHEMA**: impostazione per identificare uno schema di salvataggio dei documenti, con i relativi **parametri/proprietà**
    - (username, email, password, ID(default in mongoDB) )
- **GESTIONE ROUTES:**
  - le **route** sono i **percorsi** che si vogliono contattare **all'interno dell'url** del server
  - **IP:PORT/routes**
  - **METODI HTML**: corrispondono al metodo di express nel router.
  - **Router.method("ROUTE", callback function() )**
    - **GET**: **visualizzare** la **pagina** scelta, quindi **richiedere** al server
    - **POST**: quando il **client invia** dati **al server** (Registrazione/Login)
    - **PUT**: **aggiornare dati** già **presenti** nel server
    - **DELETE**: **eliminare dati** (**LOGOUT**)
  - Le **route** vengono **gestite in file js separati**, espressi nel comando successivo

```
app.use("/", indexRouter); //DEFINISCE COME RISPONDERE A: IP/route
app.use("/account", accountRouter); //DEFINISCE COME RISPONDERE A: IP/account/route
//vengono impostate le directory per il percorso indicato "/img, /css, /js"
app.use("/img", express.static(__dirname + "public/immagini"))
app.use("/css", express.static(__dirname + "public/styles"))
app.use("/js", express.static(__dirname + "public/scripts"))
```

- **indexRouter.js** : gestisce le **route derivanti da IP:PORT/**  
(NON RICHIEDONO SESSIONE)
  - / : pagina di registrazione
  - /login : pagina di login
- **accountRouter.js** : gestisce le route derivanti da IP:PORT/account/  
(RICHIEDONO SESSIONE)

```
router.get("/", (req, res) => {
  if (req.session.user != null) {
    //se esiste un user loggato ti rimanda alla pagina di info account
    res.redirect("/account")
  } else {
    console.log(req.session)
    res.render("register.ejs", { error: null })
  }
})
```



- contiene tutte le pagine della presentazione
- /infouser : pagina di info user loggato
- /INFO : pagina di info personali

```
router.get("/", (req, res) => {
  if (req.session.user) {
    console.log(`USER LOGGATO: ${req.session.user}`);
    res.render("INFOUSER.ejs", { user: req.session.user });
  } else {
    console.log("NESSUN UTENTE IN SESSIONE")
    res.redirect("/login")
  }
})
```

- Come si nota entrambi hanno come route specificata "/", ma nel primo caso sarà IP:PORT/ mentre nel secondo IP:PORT/account/

HEADERS			SN	Codice e descrizione
General:	Response:	Request:	1	<b>1xx: informativo</b> Significa che la richiesta è stata ricevuta e il processo sta continuando.
Request URL	Server	Cookies	2	<b>2xx: successo</b> Significa che l'azione è stata ricevuta, compresa e accettata con successo.
Request Method	Set-Cookie	Accept-xxx	3	<b>3xx: reindirizzamento</b> Significa che è necessario intraprendere ulteriori azioni per completare la richiesta.
Status Code	Content-Type	Content-Type	4	<b>4xx: errore del client</b> Significa che la richiesta contiene una sintassi errata o non può essere soddisfatta.
Remote Address	Content-Length	Content-Length	5	<b>5xx: errore del server</b> Significa che il server non è riuscito a soddisfare una richiesta apparentemente valida.
Referrer Policy	Date	Authorization		
		User-Agent		
		Referrer		

- **REQUEST:** oggetto che rappresenta la richiesta da parte del client
  - **HEADERS:** contiene i campi sopra indicati
  - **BODY:** tra i tanti documenti contiene anche i dati inseriti dall'utente Client
    - **Name, email, Password**
    - Il nome dell'input deve corrispondere al campo da cercare all'interno del REQUEST.BODY

```
router.post("/login", async(req, res) => {
  let datiInseriti = {
    username: req.body.username,
    email: req.body.email,
    password: req.body.password
  };
});
```

- **name:** username inserito
- **email:** email inserita in chiaro
- **role:** ADMIN o GUEST

```
if (req.body.email == process.env.ADMIN_EMAIL) {
  //controlla se la mail inserita corrisponde a quella da ADMIN
  console.log("UTENTE AMMINISTRATORE")
  //salva le info relative alla sessione (user loggato)
  req.session.user = {
    email: req.body.email, //si salva la mail in chiaro
    name: user.username,
    role: "ADMIN"
  }
} else {
  console.log("UTENTE OSPITE")
  req.session.user = {
    email: req.body.email, //si salva la mail in chiaro
    name: user.username,
    role: "GUEST"
  }
}
```

- **RESPONSE:** oggetto che contiene la **risposta del server**

- **HEADERS:** elementi elencati sopra

```
res.redirect("/login")
```

- **REDIRECT:** rimanda a una route specificata `response.redirect(route)`

- **RENDER:** per inviare e visualizzare una pagina indicata

- è possibile passare dei dati tramite `file.json`
- Nel caso in esame si passa l'oggetto `res.session.user` come parametro `user`

```
res.render("INFOUSER.ejs", { user: req.session.user });
```

```
<table align="center">
  <tr>
    <td id="title" colspan="2">
      <% if(user.role == "ADMIN") { %>
        
      <% } else { %>
        
      <% } %>
      <h1 align="center">CIAO
      <%= user.name %>!
    </h1>
    </td>
  </tr>
  <tr>
    <td>USERNAME: </td>
    <td>
      <%= user.name %>
    </td>
  </tr>
</table>
```

"INFOUSER.ejs"

```
if (req.body.email == process.env.ADMIN_EMAIL) {  
  //controlla se la mail inserita corrisponde a quella da ADMIN  
  console.log("UTENTE AMMINISTRATORE")  
  //salva le info relative alla sessione (user loggato)  
  req.session.user = {  
    email: req.body.email, //si salva la mail in chiaro  
    name: req.body.username,  
    role: "ADMIN"  
  }  
} else {  
  console.log("UTENTE OSPITE")  
  req.session.user = {  
    email: req.body.email, //si salva la mail in chiaro  
    name: req.body.username,  
    role: "GUEST"  
  }  
}
```

```
} catch (err) {  
  console.log(err)  
  res.status(500).send(err)  
  //in caso di errore non calcolato si manda un codice di errore al client,  
  // il quale dovrà ricaricare la pagina e riprovare  
}
```

Si invia uno stato di 500 per indicare che si è verificato un errore lato server.  
E come testo si invia l'errore (err) riscontrato.

- **AVVIARE IL SERVER:** nel caso di problemi con la ricezione della richiesta accedere al firewall e aprire la porta desiderata, quella su cui il server sarà in ascolto.

Server.listen(PORT)

- **“npm run script”**

- start: avvia “node HOME.js”
  - avvia il programma HOME.js
  - per stoppare premere Ctrl + C
    - Chiederà se si desidera fermare il processo. Rispondere S(si) / N(no)
- devStart: avvia “nodemon HOME.js”
  - avvia il programma HOME.js ma tramite nodemon
    - ogni volta che si effettua una modifica al server e si salva il server riparte automaticamente
    - questo è utile in fase di sviluppo (development)



- **USER FRIENDLY:**

- Ho cercato di rendere il codice il più leggibile possibile, commentando alcune operazioni per renderle più chiare al lettore
- Pagine WEB semplici e d'impatto per visualizzare al meglio le informazioni riguardanti la presentazione.

- **Linguaggi utilizzati:**

- **HTML: Hyper Textual Markup Language:** linguaggio utilizzato per la presentazione delle pagine
- **Javascript:** linguaggio di programmazione utilizzato per gestire al meglio gli script e le risorse inserite dall'utente via HTML
- **PHP:** per gestire lato client, quindi all'interno delle pagine HTML, le informazioni passate dal server
  - Nell'esempio riportato precedentemente per effettuare il controllo tra ADMIN e GUEST si usa la sintassi di PHP <% operazione %>
- **EJS:** motore di visualizzazione delle pagine HTML, permette una migliore gestione dei parametri da passare tra server e client

```
//queste 2 funzioni servono per gestire l'indirizzamento corretto verso i file ejs  
//i quali andranno visualizzati dal client  
app.set('view-engine', 'ejs');  
app.set("views", path.join(__dirname, "views"));
```

- **Configurazione GIT**

- **GR:** global repository, si trova su github.com al link indicato all'inizio del documento, da cui è stato scaricato il progetto intero
- **LR:** local repository, Copia del GR nella macchina locale. Si aggiorna con il commit
- **WD:** working directory, cartella locale su cui io programmatore sto lavorando sul mio PC
- **SA:** Staging Area, livello intermedio nel quale i file sono stati aggiunti per poi essere salvati in LR dopo il commit

- **Configurazione del terminale per usare la GIT BASH, installata dal seguente link [GIT-DOWNLOAD](#)**

Una volta installato git è possibile avviare la BASH nella cartella desiderata (WD)

- **CREAZIONE ACCOUNT E GR dal sito [GITHUB.com](#)**

- Dalla gitbash digitare I seguenti comandi:

- Configurazione account da usare nel pc locale (deve essere quello registrato su github.com)
  - <git config --global user.name "USERNAME">
  - <git config --global user.email "EMAIL" >

- Selezionare la cartella dove usare la WD
  - <git init>: inizializza un Repository Locale (LR) vuoto

```
damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER
$ git init
Initialized empty Git repository in C:/TESINA ESAME/GITSERVER/.git/

damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        COME FUNZIONA.odt
        HOME.js
        README.md
        models/
        package-lock.json
        package.json
        public/
        routes/
        views/

nothing added to commit but untracked files present (use "git add" to track)
```

- Inserire I FILE del progetto all'interno della cartella appena configurata come LR
  - <git status>: mostra lo stato degli elementi nella cartella (MASTER), cioè quella appena ottenuta dalla fase di clone
    - Se sono scritti in rosso vuol dire che sono UNTRACKED, cioè non ancora aggiunti in SA
    - Se sono verdi sono aggiunti in SA ma non ancora salvati in LR
    - MODIFIED: sono modificati rispetto alla versione salvata in LR
    - STAGED: modificato rispetto al LR ma è già stato salvato in SA
    - COMMITTED: salvato in LR
  - git add FILE: aggiunge il file da WD a SA, quindi lo rende riconoscibile da git

```
damia@DamianoPC MINGW64 /c/TESINA ESAME/MONGODB WORKING/SERVER-NODE (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        new file:   COME FUNZIONA.odt
        new file:   HOME.js
        new file:   models/model.js
        new file:   package-lock.json
        new file:   package.json
        new file:   public/immagini/3-4-INDUTRIA.png
        new file:   public/immagini/3-4-industria.png
        new file:   public/immagini/5g-timeline.jpg
        new file:   public/immagini/5g.jpg
        new file:   public/immagini/5gvelocity.jpg
        new file:   public/immagini/FUMAGALLI-DAMIANO.jpeg
        new file:   public/immagini/PCTO.png
        new file:   public/immagini/SOTTORETE.jpg
        new file:   "public/immagini/Smart thinking 1\342\200\231intelligenza
        new file:   "public/immagini/Smart thinking 1\342\200\231intelligenza
```

- <git commit -m "MESSAGGIO">: salva in LR

```
damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git commit -m "COMMIT COMPLETO"
[master (root-commit) 46d1ca4] COMMIT COMPLETO
111 files changed, 4687 insertions(+)
create mode 100644 .gitignore
create mode 100644 COME FUNZIONA.odt
create mode 100644 HOME.js
create mode 100644 README.md
create mode 100644 models/model.js
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 public/immagini/3 4 INDUSTRIA.png
create mode 100644 public/immagini/3-4-industria.png
create mode 100644 public/immagini/5g-timeline.jpg
create mode 100644 public/immagini/5g.jpg
create mode 100644 public/immagini/5gvelocity.jpg
create mode 100644 public/immagini/FUMAGALLI-DAMIANO.jpeg
create mode 100644 public/immagini/PCTO.png
```

- <git lfs install>: dopo aver installato il programma Large File Storage da [qui](#), eseguire questi comandi per permettere a git di salvare file di dimensioni elevate, superiori allo standard

```
damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git lfs install
Updated git hooks.
Git LFS initialized.

damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git lfs track "*.mp4"
Tracking "*.mp4"

damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git add .gitattributes

damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git commit -m "aggiunto file .gitattributes"
[master 71844a0] aggiu ofile .gitattributes
1 file changed, 1 insertion(+)
create mode 100644 .gitattributes
```

- <git remote add origin GITURL>: aggiunge al parametro origin la URL del GR
- <git push origin master>: salva i dati presenti in LR nel GR

```
damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git remote add origin https://github.com/DamiFuma02/SERVER-NODE.git

damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git push origin master
Uploading LFS objects: 100% (1/1), 68 MB | 525 KB/s, done.
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 85.25 KiB | 2.84 MiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/DamiFuma02/SERVER-NODE.git
71844a0..6066a14 master -> master
```

git rm --cache FILE>: rimuove il file indicato solamente dal GR, non dalla WD

- <git rm FILE>: rimuove da WD

```
damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git rm --cached APPTUTORIAL.docx
rm 'APPTUTORIAL.docx'

damia@DamianoPC MINGW64 /c/TESINA ESAME/GITSERVER (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    APPTUTORIAL.docx
        new file:   APPTUTORIAL.pdf
```

















- <git clone URL>: copia fi file trovati all'URL indicato all'interno della WD

```
damia@DamianoPC MINGW64 /c/TESINA ESAME/Nuova cartella (master)
$ git clone https://github.com/DamiFuma02/SERVER-NODE.git
Cloning into 'SERVER-NODE'...
remote: Enumerating objects: 296, done.
remote: Counting objects: 100% (296/296), done.
remote: Compressing objects: 100% (212/212), done.
Receiving objects: 9% (27/296), 9.67 MiB | 139.00 KiB/s
```

## • STRUTTURA DELLE CARTELLE

- **NODE-SERVER**: cartella “ROOT” scaricata da GIT
  - **Html-files**: file HTML static, cioè non interattivi
  - **Models**: contiene il modello degli oggetti da inserire nel DB
  - **Public**: contiene tutti i file statici
    - Immagini: immagini e video
    - Styles: contiene i modelli di stile CSS applicati ad HTML
  - **Routes**: contiene i router configurati
    - Index.js = indexRouter
    - Account.js = accountRouter
  - **Views**: contiene I file EJS (HTML dinamico) da renderizzare nelle routes
    - **It**: contiene i file della presentazione in italiano (richiede sessione)
    - **Eng**: contiene i file del PCTO scritti in inglese (richiede sessione)
    - Altri FILE base, quali **Login**, **Registrazione**, **Tutorial** e **INFOUSER**
  - **.git attributes**: file di configurazione nel quale è inserita l'opzione di commit di file grandi
  - **.Gitignore**: indica quali file vengono ignorati da git
  - **APPTUTORIAL.pdf**: questo file che si sta leggendo
  - **HOME.js**: main del programma, quindi quello che viene eseguito
  - **OFFLINE.pptx**: presentazione PowerPoint di riserva
  - **README**: file di info generato da git

- **WORKING DIRECTORY = LOCAL REPOSITORY:** la cartella del PC locale sul quale si stà lavorando, quindi modificando il programma

 <b>html-files</b>	28/05/2021 16:35	Cartella di file	
 <b>models</b>	27/05/2021 21:36	Cartella di file	
 <b>node_modules</b>	26/05/2021 15:39	Cartella di file	
 <b>public</b>	29/05/2021 22:47	Cartella di file	
 <b>routes</b>	26/05/2021 15:39	Cartella di file	
 <b>views</b>	26/05/2021 15:39	Cartella di file	
 <b>.env</b>	22/05/2021 15:33	File ENV	1 KB
 <b>.gitattributes</b>	26/05/2021 16:02	Documento di testo	1 KB
 <b>.gitignore</b>	28/05/2021 21:12	Documento di testo	1 KB
 <b>APPTUTORIAL.docx</b>	29/05/2021 22:56	Documento di Mi...	1.259 KB
 <b>APPTUTORIAL.pdf</b>	29/05/2021 15:36	Microsoft Edge P...	1.156 KB
 <b>HOME.js</b>	29/05/2021 22:47	File di origine Java...	3 KB
 <b>OFFLINE.pptx</b>	27/05/2021 21:35	Presentazione di ...	15.841 KB
 <b>package.json</b>	23/05/2021 21:57	JSON File	1 KB
 <b>package-lock.json</b>	23/05/2021 21:57	JSON File	87 KB
 <b>README.md</b>	28/05/2021 19:07	File di origine Mar...	1 KB

- **.env** : contiene le variabili d'ambiente, perciò quelle che saranno salvate solamente nel LR
  - **SESSION-SECRET:** sequenza di caratteri che serve a criptare le info della sessione
  - **PORT:** porta sulla quale il server sarà in ascolto
  - **DATABASE\_URL:** per accedere al mongoDB
  - **ADMIN\_EMAIL**

Per richiedere l'utilizzo delle variabili d'ambiente è necessario il seguente comando

```
if (process.env.NODE_ENV !== "production") {
  require("dotenv").config();
}
```




**porta = process.env.PORT || 80**

Se esistono le variabili di ambiente si usano quelle altrimenti al posto di dare errore si assegna 80

- **APPTUTORIAL.docx:** file WORD che sarà convertito poi in pdf per una maggiore sicurezza
- **Package.json & package-lock.json:** sono I file generati quando si eseguono i comandi di NODE, quali "npm init" o "npm install pacchetto"
- **Node\_modules:** cartella contenente tutte le librerie installate, circa 200MB. Diventa troppo oneroso salvarlo nel GR.  
Anch'esso viene generato automaticamente quando si installa un pacchetto




- **GLOBAL REPOSITORY:** cartelle MASTER presente su github.com

 master ▾  2 branches  0 tags

[Go to file](#) [Add file ▾](#) [Code ▾](#)

This branch is 18 commits ahead, 1 commit behind main. [Pull request](#) [Compare](#)

 **DamiFuma02 FINALE** db524db 22 ore fa 🕒 18 commits

html-files	FINALE	22 ore fa
models	Modifica PCTO	l'altro ieri
public/immagini	FINALE	22 ore fa
routes	Aggiornamento sessione e FINE	ieri
views	FINALE	22 ore fa
.gitattributes	aggiuo ofile .gitattributes	3 giorni fa
.gitignore	rimozione package	3 giorni fa
APPTUTORIAL.pdf	Aggiornamento Tutorial.pdf	23 ore fa
HOME.js	Aggiornamento TUTORIAL.js	23 ore fa
OFFLINE.pptx	Aggiornamento Sitografia	l'altro ieri
README.md	COMMIT COMPLETO	3 giorni fa

Come si nota ci sono meno files.

Ciò perché ho voluto rendere il programma “universale”, cioè il più possibile adattabile alle esigenze di chi lo scarica e lo vuole usare.

Le **scritte centrali** corrispondono al **messaggio** inserito nella fase di **commit**.

Premendo su **compare** è possibile vedere le **differenze** tra una **versione vecchia e nuova**, quindi di **2 commit differenti**.

Questo è uno dei **vantaggi principali** della **piattaforma GIT** perciò ho deciso di utilizzarla.