

[Τίτλος αναφοράς]

[Όνομα σπουδαστή]

[Όνομα μαθήματος]

[Όνομα εκπαιδευτή]

[20/6/2021]

Πίνακας Περιεχομένων

[Εισαγωγή 3](#_Toc76120226)

[Δημιουργία Γραφικής διεπαφής. 3](#_Toc76120227)

[Δημιουργία backend server 6](#_Toc76120228)

[Προγραμματιστικό περιβάλλον 7](#_Toc76120229)

[Frontend 10](#_Toc76120230)

[Jenkins 12](#_Toc76120231)

[Backend 12](#_Toc76120232)

[Frontend 17](#_Toc76120233)

[Graphical user interface, application, table

Description automatically generated 18](#_Toc76120234)

[Εικόνα 1: Αρχική σελίδα 3](#_Toc76120939)

[Εικόνα 2: Κεντρική σελίδα 4](#_Toc76120940)

[Εικόνα 3: Μενού με διαθέσιμες επιλογές 4](#_Toc76120941)

[Εικόνα 4: Οθόνη αναζήτηση 5](#_Toc76120942)

[Εικόνα 5: Οθόνη για εισαγωγή στοιχείων για αίτηση. 5](#_Toc76120943)

[Εικόνα 6: Περιγραφή των διαθέσιμων πακέτων 6](#_Toc76120944)

[Εικόνα 7: Πρόγραμμα κατά τη λειτουργία 6](#_Toc76120945)

[Εικόνα 8: Μεταβλητές του συστήματος 6](#_Toc76120946)

[Εικόνα 9: Προγραμματιστικό περιβάλλον για την ανάπτυξη της backend εφαρμογής 7](#_Toc76120947)

[Εικόνα 10: Debug μέσα από το περιβάλλον ανάπτυξης 7](#_Toc76120948)

[Εικόνα 11: Παράδειγμα από ένα entity που δημιουργήσαμε. 8](#_Toc76120949)

[Εικόνα 12: Δομή κώδικα 9](#_Toc76120950)

[Εικόνα 13: mvn stage 10](#_Toc76120951)

[Εικόνα 14: Περιβάλλον ανάπτυξη εφαρμογής 11](#_Toc76120952)

[Εικόνα 15: Δομή κώδικα 11](#_Toc76120953)

[Εικόνα 16: Pipelines στο Jenkins 12](#_Toc76120954)

[Εικόνα 17: Πίνακας με τα αποτελέσματα των **pipeline** 12](#_Toc76120955)

[Εικόνα 18: Αποτέλεσμα επιτυχημένης εκτέλεσης 13](#_Toc76120956)

[Εικόνα 19: Αποτελέσματα εκτέλεσης 14](#_Toc76120957)

[Εικόνα 20: Console κατά την διάρκεια εκτέλεσης 15](#_Toc76120958)

[Εικόνα 20: Stage view for pipeline project 18](#_Toc76120959)

[Εικόνα 20: Console κατά την διάρκεια εκτέλεσης 19](#_Toc76120960)

[Εικόνα 20: Docker Hub account 22](#_Toc76120961)

[Εικόνα 20: Github account 22](#_Toc76120962)

# Εισαγωγή

Δημιουργία ενός συστήματος για να υποστηρίζει την δημιουργία μιας αίτησης για συνταξιοδότηση. Ο ασφαλισμένος θα έχει την δυνατότητα αφού συνδεθεί να δει τα διαθέσιμα πακέτα για συνταξιοδότηση και τα δικαιολογητικά που χρειάζεται για κάθε ένα από αυτά. Στην συνέχεια, θα είναι σε θέση να υποβάλει την αίτηση.

Το σύστημα θα υποστηρίζεται από ένα κατάλληλο backend που θα αναλάβει να επικοινωνεί με την βάση και να εξυπηρετεί το frontend σύστημα με ένα RESTAPI.

Ένα σύστημα θα υπάρχει που θα βοηθάει στο συνεχή έλεγχο του συστήματος της μεταγλώττισής, του ελέγχου και των κατάλληλων διαδικασιών για την αυτοματοποίηση της εγκατάστασης του συστήματος.

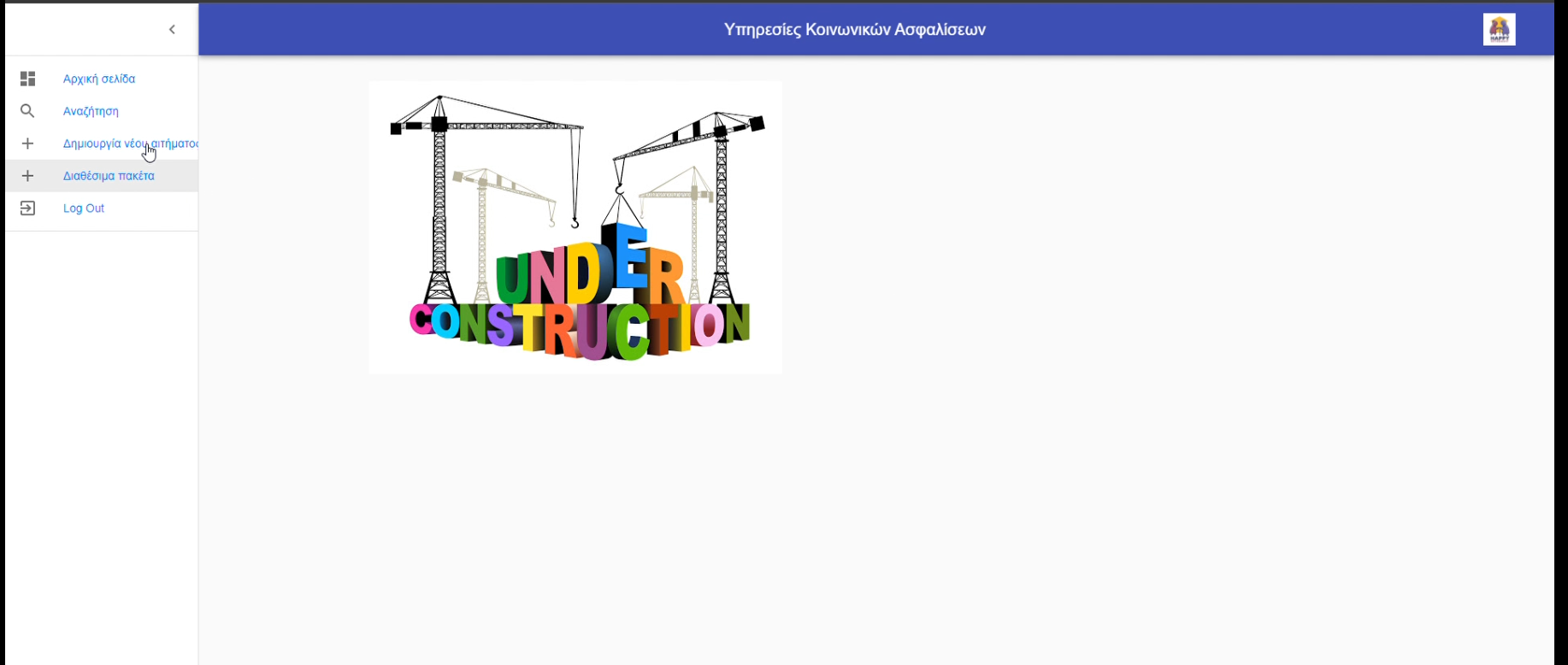
# Δημιουργία Γραφικής διεπαφής.

Το ένα μέρος του τελικού προϊόντος έχει να κάνει με το site το οποίο βλέπει ο χρήστης. Η αρχική οθόνη που βλέπει ο χρήστης είναι στη παρακάτω εικόνα. Στην αρχική οθόνη υπάρχει η δυνατότητα να δώσει το κωδικό του για να συνδεθεί, να ζητήσει να εγγραφεί στο σύστημα αλλά και να επαναφέρει το κωδικό του αν τον έχει ξεχάσει.



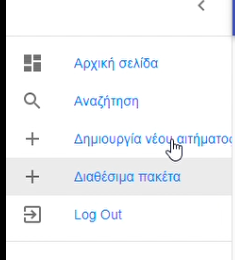
Εικόνα : Αρχική σελίδα

Αφού ο χρήστης συνδεθεί με επιτυχία θα βλέπει την αρχική οθόνη, η οποία θα έχει ένα σύνολο από επιλογές (menu στα αριστερά).



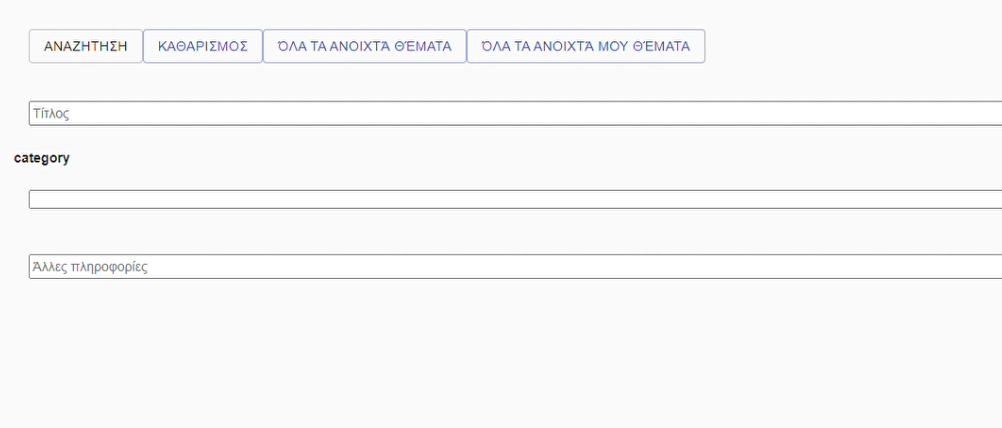
Εικόνα : Κεντρική σελίδα

Οι διαθέσιμες επιλογές φαίνονται στη παρακάτω εικόνα:



Εικόνα : Μενού με διαθέσιμες επιλογές

Η μια λειτουργία που παρέχει το σύστημα είναι η δυνατότητα του χρήστη να αναζητήσει πληροφορία για τα ενεργά του αιτήματα.



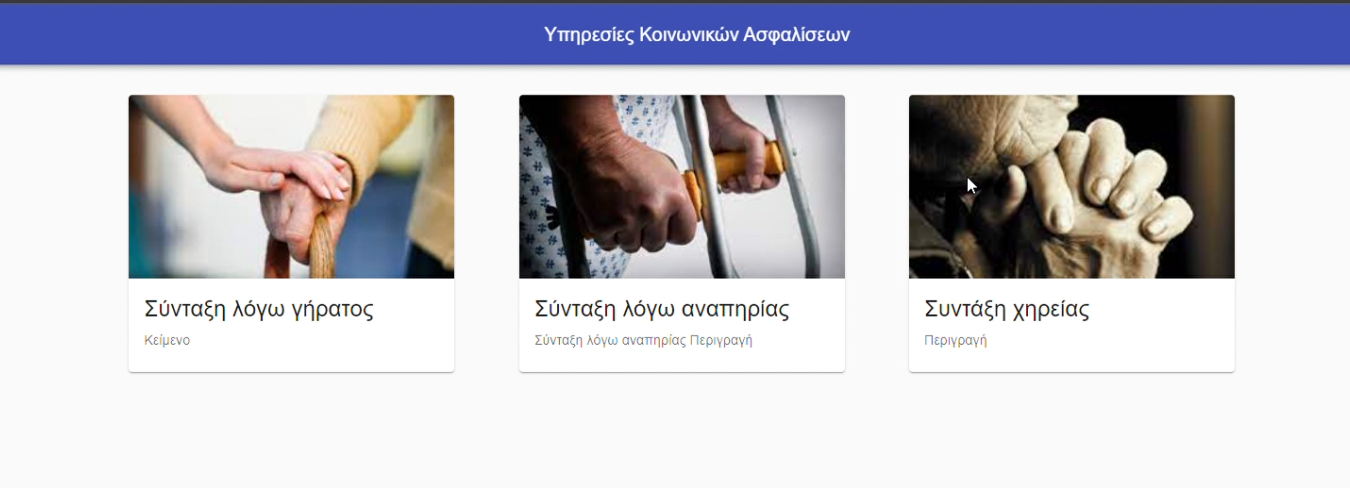
Εικόνα : Οθόνη αναζήτηση

Στην παρακάτω εικόνα φαίνεται το περιβάλλον στο οποίο μπορεί να υποβάλλει την αίτηση για κάποιο επίδομα.



Εικόνα : Οθόνη για εισαγωγή στοιχείων για αίτηση.

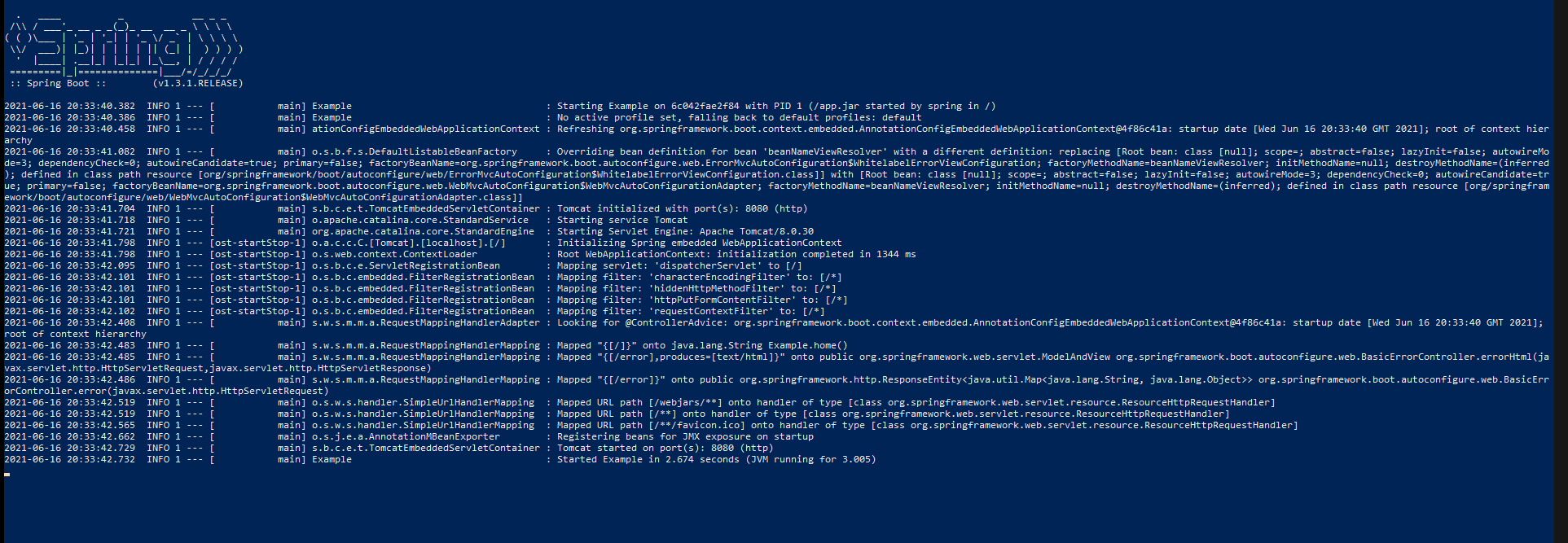
Τέλος στην εικόνα που ακολουθεί φαίνεται το περιβάλλον στο οποίο φαίνονται οι διάφορες παροχές που προσφέρει το σύστημα ασφαλείας.



Εικόνα : Περιγραφή των διαθέσιμων πακέτων

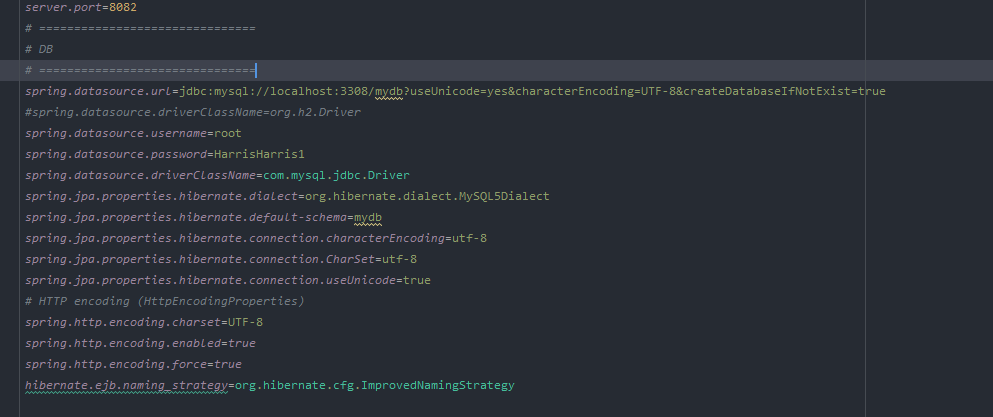
# Δημιουργία backend server

Η εικόνα είναι από το server κατά την διάρκεια τις λειτουργίας του.



Εικόνα : Πρόγραμμα κατά τη λειτουργία

Το πρόγραμμα έχει ένα σύνολο από παραμέτρους τις οποίες το σύστημα μπορεί εύκολα να το παραμετροποιήσει.



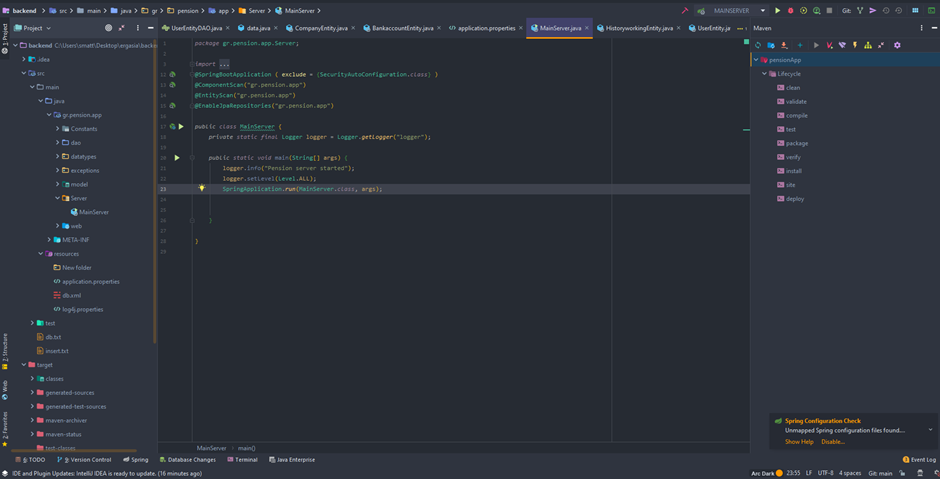
Εικόνα : Μεταβλητές του συστήματος

### Προγραμματιστικό περιβάλλον

Για το προγραμματισμό χρησιμοποιήσαμε σαν περιβάλλον το IdealJ καθώς και για την δημιουργία του backend και του frontend.

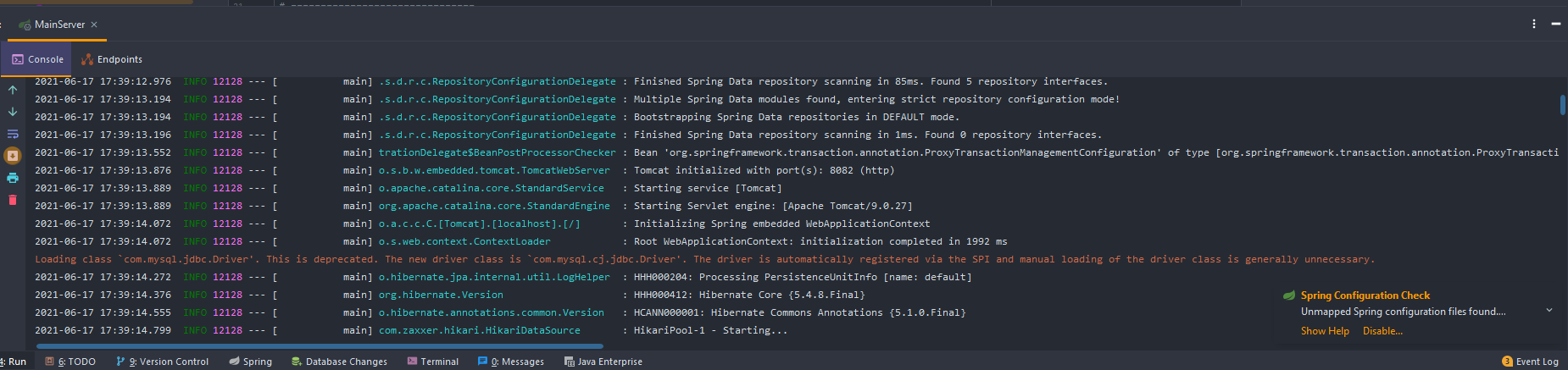
Backend

Στην παρακάτω εικόνα φαίνεται το περιβάλλον κατά την διάρκεια του προγραμματισμού.



Εικόνα : Προγραμματιστικό περιβάλλον για την ανάπτυξη της backend εφαρμογής

Μέσα από το περιβάλλον είχαμε την δυνατότητα να τρέξουμε την εφαρμογή σε debug mode



Εικόνα : Debug μέσα από το περιβάλλον ανάπτυξης

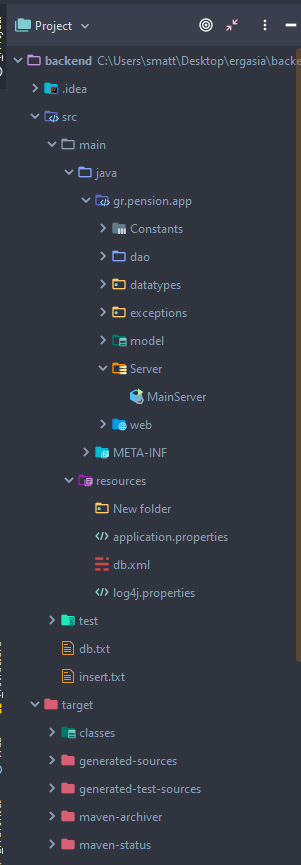
H επικοινωνία της βάσης γίνεται με την χρήση Hibernate. Κάθε πίνακας αντιστοιχεί σε μία κλάση όπως φαίνεται στην επόμενη εικόνα. Μέσα από τις παραμέτρους που έχουμε δώσει στο σύστημα μπορούμε να επιλέξουμε να δημιουργείται αυτόματα η βάση αν δεν βρεθεί.



Εικόνα : Παράδειγμα από ένα entity που δημιουργήσαμε.

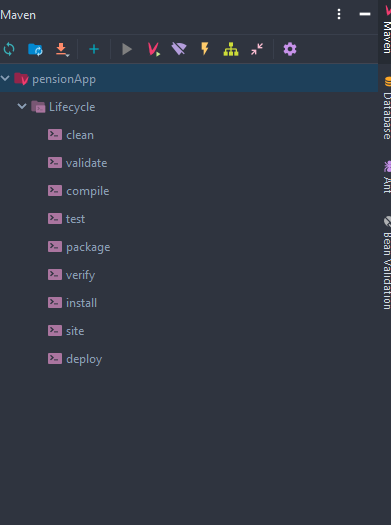
Παρακάτω φαίνεται η οργάνωση του κώδικα σε πακέτα. Πιο σημαντικά θεωρούνται τα εξής:

* DAO: αναλαμβάνει την επικοινωνία με την βάση για την εγγραφή και ανάγνωση δεδομένων.
* Model: Έχει τα διάφορα entities ta
* Web: Το σύνολο από τα web services.



Εικόνα : Δομή κώδικα

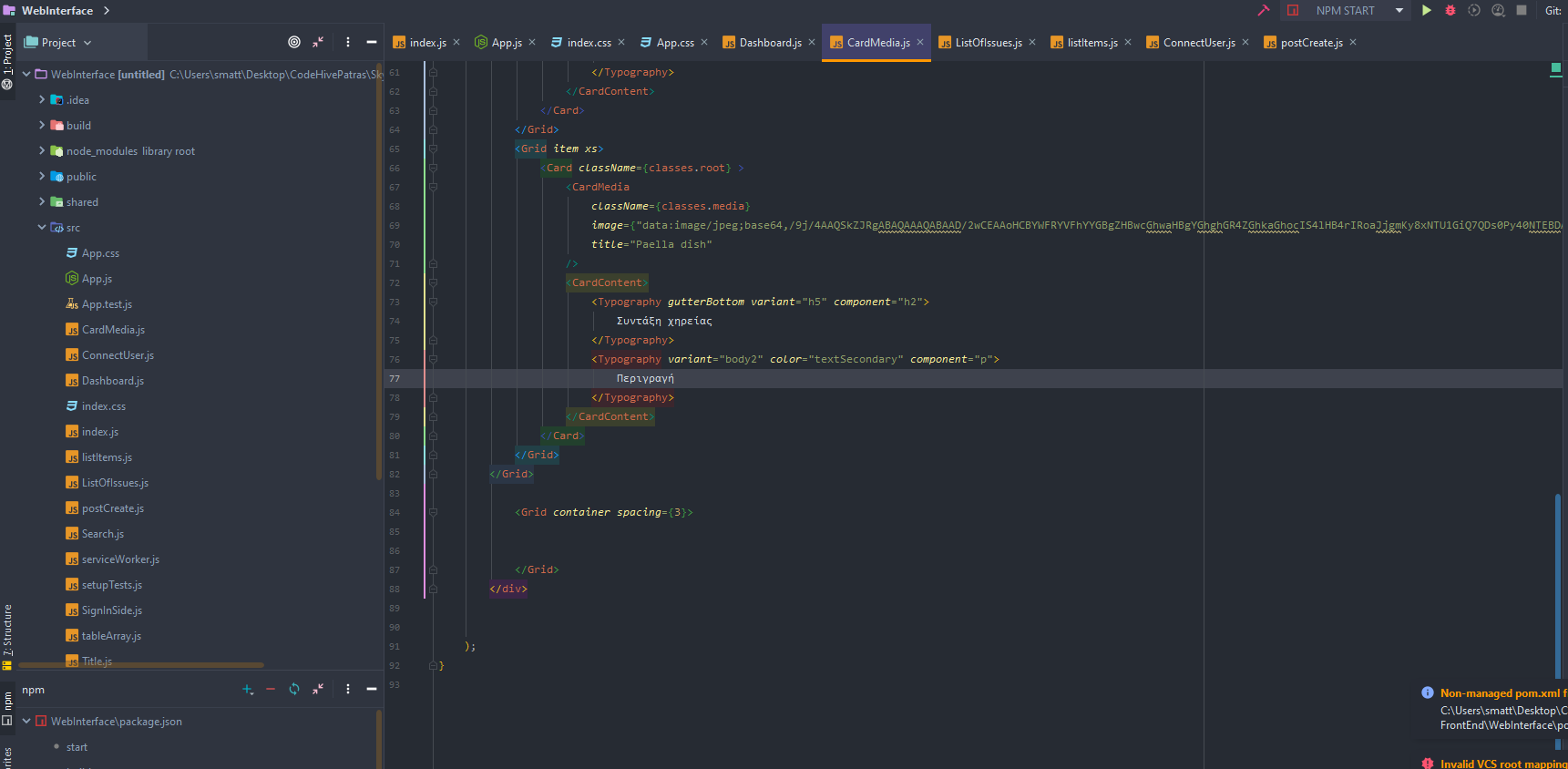
Για το αυτοματισμό του συστήματος επιλέξαμε το maven.



Εικόνα 13: mvn stage

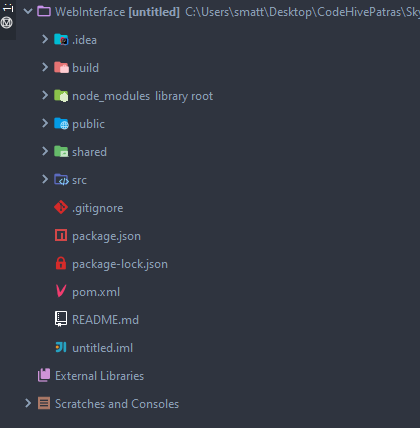
### Frontend

Όπως έχει ήδη αναφερθεί για το προγραμματισμό του frontend επιλέξαμε τη βιβλιοθήκη REACT. H εφαρμογή αυτή είναι ένα σύνολο από σελίδες όπου η κάθε σελίδα αντιστοιχεί και σε ένα component.



Εικόνα : Περιβάλλον ανάπτυξη εφαρμογής

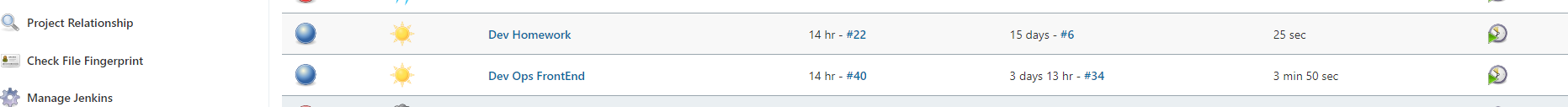
Στη παρακάτω εικόνα φαίνονται η δομή. Στο package.json υπάρχουν όλες οι εξαρτήσεις από εξωτερικές βιβλιοθήκες. Στο node\_modules υπάρχουν οι βιβλιοθήκες που έχουν εξάρτηση και έχουν κατέβει στο σύστημα μας.



Εικόνα : Δομή κώδικα

# Jenkins

Για την αυτοματοποίηση του συστήματος έχουμε φτιάξει ένα Jenkins server. Στο server έχουμε δυο pipeline, ένα για το frontend και ένα για το Backend.

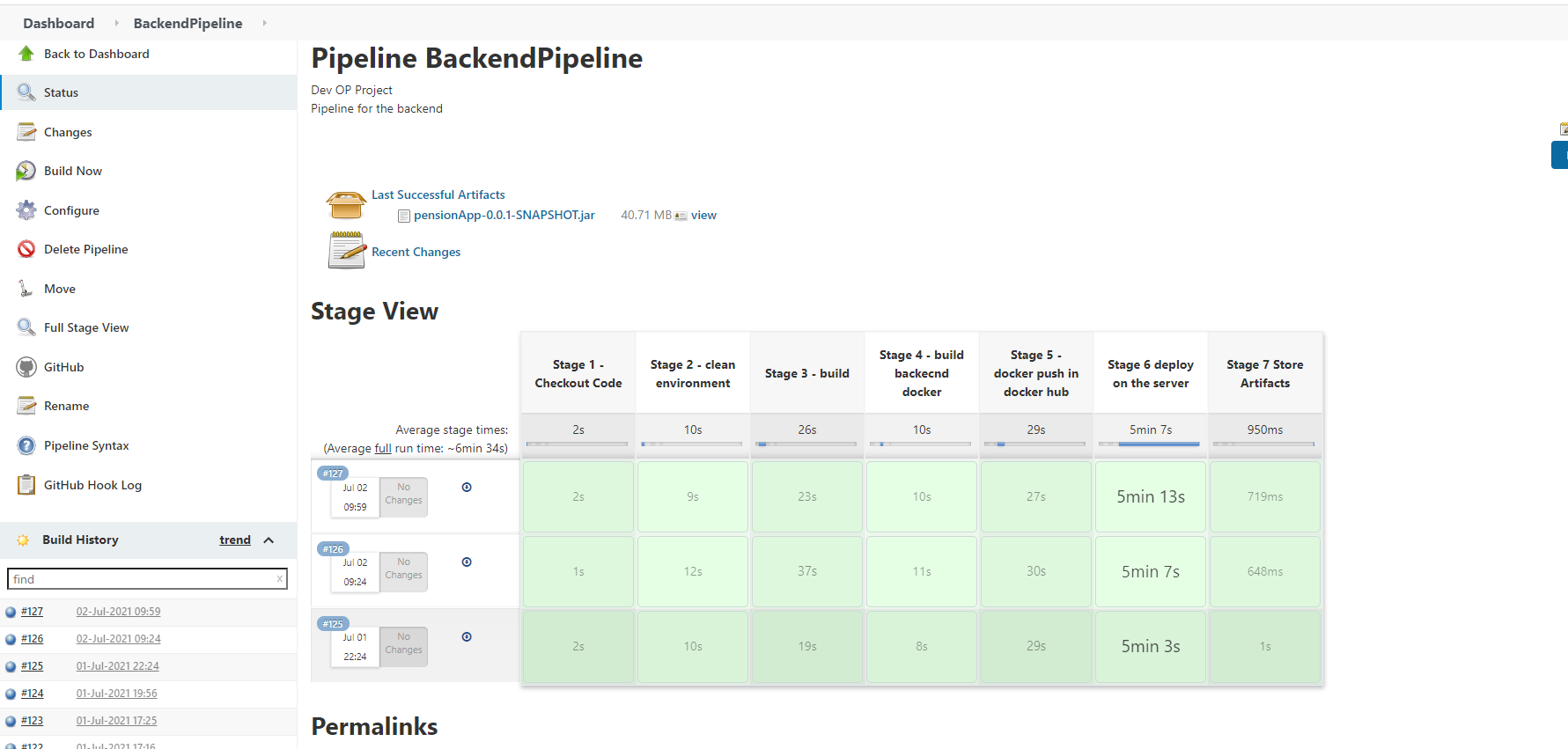


Εικόνα 16: Pipelines στο Jenkins

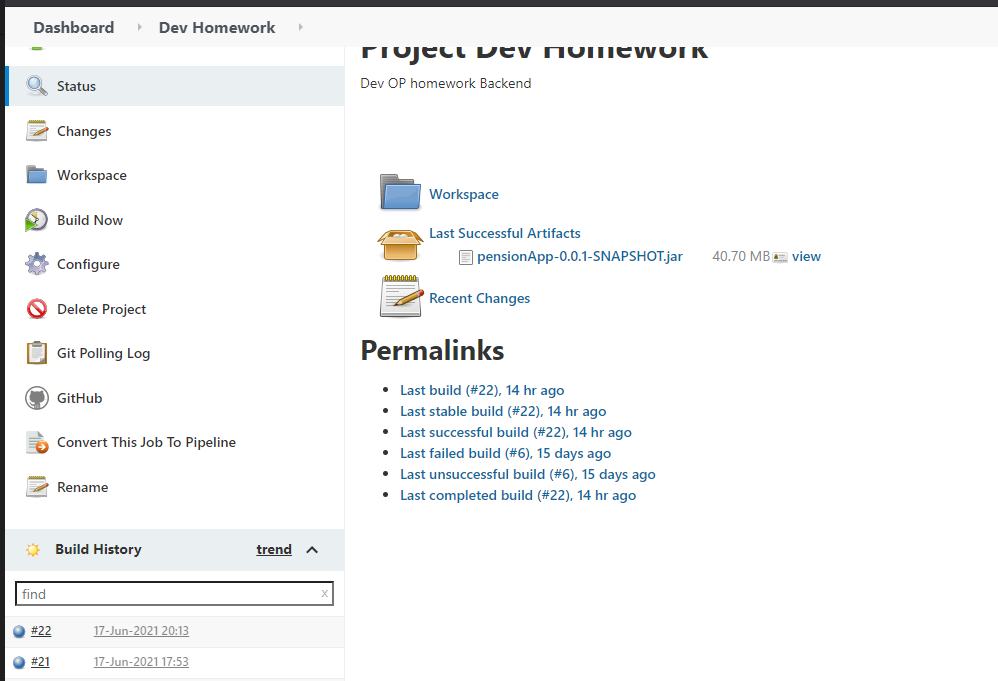
Παρακάτω φαίνεται το αποτέλεσμα από μια επιτυχημένη εκτέλεση για το **pipeline.**

## Backend

`



Εικόνα : Πίνακας με τα αποτελέσματα των **pipeline**



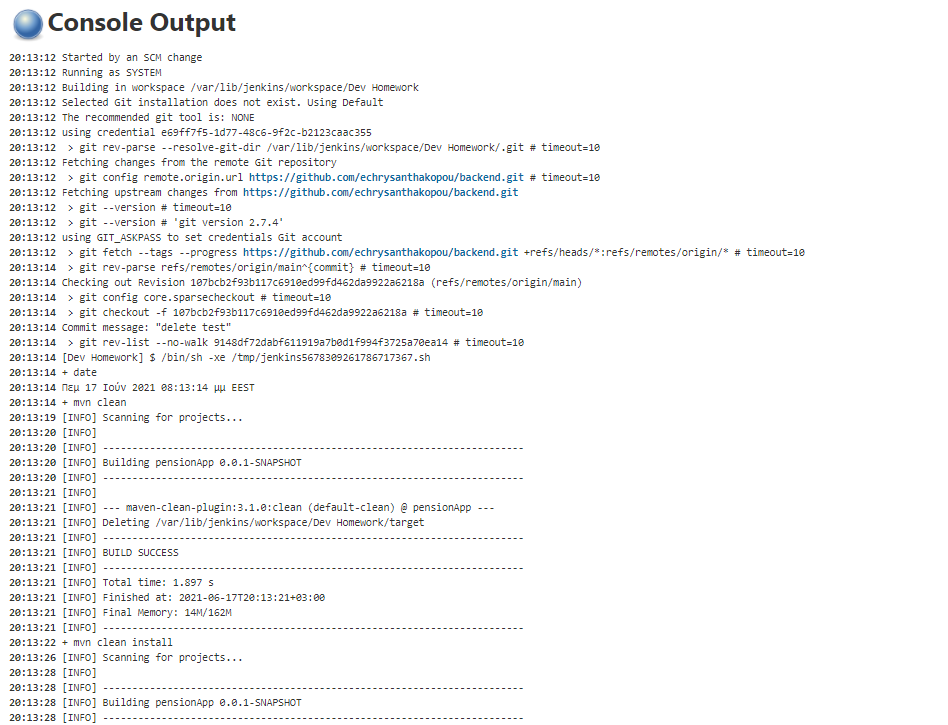
Εικόνα : Αποτέλεσμα επιτυχημένης εκτέλεσης

Για την ενεργοποίηση του pipeline έχει γίνει χρήση του κατάλληλου **trigger ώστε να ενεργοποιείται αυτόματα.**



Εικόνα : Αποτελέσματα εκτέλεσης

Κατά την διάρκεια της εκτέλεσης μπορούμε να δούμε τα αποτελέσματα στην console και να βρούμε πιθανά λάθη.



Εικόνα : Console κατά την διάρκεια εκτέλεσης

|  |
| --- |
| pipeline {    agent any  environment {  VERSION\_TAG = getVersion()  }  stages {  stage ('Stage 1 - Checkout Code') {  steps{  checkout([$class: 'GitSCM', branches: [[name: '\*/main']],  doGenerateSubmoduleConfigurations: false, extensions: [],  submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'e69ff7f5-1d77-48c6-9f2c-b2123caac355',  url: 'https://github.com/echrysanthakopou/backend.git']]])  }  }  stage ('Stage 2 - clean environment') {  steps{  sh " mvn clean"  }  }  stage('Stage 3 - build')  {steps{  sh "mvn clean install"  }}  stage('Stage 4 - build backecnd docker')  {steps{  sh "docker build . -t echrysanthakopou/back:1\_0\_0"  }}  stage('Stage 5 - docker push in docker hub')  {steps{  sh "docker login -u echrysanthakopou -p Evelin1234!@#\$"  sh "docker push echrysanthakopou/back:1\_0\_0"  }}  stage('Stage 6 deploy on the server')  {steps{        ansiblePlaybook installation: 'ansible', playbook: 'deploy.yml'    }}  stage('Stage 7 Store Artifacts')  {steps{  archiveArtifacts allowEmptyArchive: false, artifacts: 'target/\*.jar',  caseSensitive: true, defaultExcludes: true, fingerprint: false,  onlyIfSuccessful: false  }}  }  }  def getVersion(){  def version=sh returnStdout: true, script: 'git rev-parse --short HEAD'  return version  } |
|  |

|  |
| --- |
| ---  - name: "Playing with Ansible and Git"  hosts: localhost  connection: local  # - name: ensure repository key is installed  # apt\_key:  # url: https://download.docker.com/linux/ubuntu/gpg  # state: present  # - name: ensure docker registry is available  # apt\_repository: repo='deb https://download.docker.com/linux/ubuntu bionic stable' state=present  # - name: ensure docker and dependencies are installed  # apt: name=docker-ce update\_cache=yes  # # Uncomment the following to enable insecure registries with Docker  # #- name: ensure docker can use insecure registries in 10.11.0.0/16  # # lineinfile: "dest=/etc/default/docker regexp=^DOCKER\_OPTS line=DOCKER\_OPTS='--insecure-registry 10.11.0.0/16'"  # - service: name=docker state=restarted    tasks:  - name: "jDownload my sql docker"  shell: "docker pull mysql/mysql-server:8.0"  register: "output"  - name: "Remove running mysql"  shell: "docker rm -f mysql-docker"  register: "output"  - name: "Run mysql docker"  shell: "docker run --name=mysql-docker -p 3306:3306 -e MYSQL\_ROOT\_HOST='%' -e MYSQL\_ROOT\_PASSWORD=HarrisHarris1 -e MYSQL\_DATABASE=mydb1 -d mysql/mysql-server:8.0 "  register: "output"  - name: "pull backend docker"  shell: "docker pull echrysanthakopou/back:1\_0\_0"  register: "output"  - name: "remove previous docker "  shell: "docker rm -f back"  register: "output"  - name: "wait for db to initialize"  shell: "sleep 200; "  register: "output"  - name: "deploy"  shell: " docker run --name=back --net=host -p 8082:8082 -d=true echrysanthakopou/back:1\_0\_0 "  register: "output"  - name: "jDownload my mailhog docker"  shell: "docker pull mailhog/mailhog"  register: "output"  - name: "jDownload my mailhog docker"  shell: "docker rm -f mailhog"  register: "output"  - name: "Run mailhog docker"  shell: " docker run -d=true --name=mailhog -p 8025:8025 -p 1025:1025 mailhog/mailhog "  register: "output"  - name: "wait for docker to initialize"  shell: "sleep 60; "  register: "output"  - debug: var=output.stdout\_lines |

# Frontend

# Graphical user interface, application, table Description automatically generated

Εικόνα 21: Stage view for pipeline project

Graphical user interface, text, application

Description automatically generated

Εικόνα : Console κατά την διάρκεια εκτέλεσης

|  |
| --- |
| pipeline {    agent any  environment {  VERSION\_TAG = getVersion()  }  stages {  stage ('Stage 1 - Checkout') {  steps{  checkout([$class: 'GitSCM', branches: [[name: '\*/main']],  doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [],  userRemoteConfigs: [[credentialsId: 'e69ff7f5-1d77-48c6-9f2c-b2123caac355',  url: 'https://github.com/echrysanthakopou/frontEnd']]])  }  }  stage ('Stage 2 - Download dependance') {  steps{  sh "cd frontEnd; npm install"  }  }  stage('Stage 3 -build')  {steps{    sh """ cd frontEnd;npm run build """  }}  stage('stage 4 - package create war')  {steps{    sh " cd frontEnd; mvn package"  }}  stage('stage 5 - create docker')  {steps{  sh "cd frontEnd; docker build . -t echrysanthakopou/front:1\_0\_0"  }}  stage('stage 6 - docker push to docker hub')  {steps{  sh "docker login -u echrysanthakopou -p Evelin1234!@#\$"  sh "cd frontEnd;docker push echrysanthakopou/front:1\_0\_0"  }}  stage('Stage 7 deploy to the server')  {steps{    ansiblePlaybook installation: 'ansible', playbook: 'deploy.yml'      }}  stage('Store Artifacts')  {steps{  archiveArtifacts allowEmptyArchive: false, artifacts:  'frontEnd/target/\*.war', caseSensitive: true, defaultExcludes: true,  fingerprint: false, onlyIfSuccessful: false  }}  }  }  def getVersion(){  def version=sh returnStdout: true, script: 'git rev-parse --short HEAD'  return version  } |

|  |
| --- |
| ---- name: "Playing with Ansible and Git"hosts: localhostconnection: localtasks:- name: "just execute a ls -lrt command"shell: "docker pull echrysanthakopou/front:1\_0\_0"register: "output"- name: "just execute a ls -lrt command1"shell: "docker rm -f front"register: "output"- name: "just execute a ls -lrt command2"shell: "docker run --name=front -p 88:8080 echrysanthakopou/front:1\_0\_0 &"register: "output"- debug: var=output.stdout\_lines |

## Account

Graphical user interface, text, application, email

Description automatically generated

Εικόνα : Docker Hub account

A screenshot of a computer

Description automatically generated

Εικόνα : Github account

Repositories link

* <https://github.com/echrysanthakopou/backend>
* <https://github.com/echrysanthakopou/frontEnd>
* <https://hub.docker.com/u/echrysanthakopou>