

# ***MOVIE SELECTOR***

**Παρουσίαση της εργασίας για το μάθημα  
“Διαδίκτυο και Εφαρμογές” 2020 HMMY**

**Αθανάσιος Κουτρούμπας  
Α.Μ: 03116073**

# Περιγραφή Εργασίας

Σκοπός της εργασίας, είναι η δημιουργία μιας διαδικτυακής εφαρμογής στην οποία ο χρήστης θα μπορεί να αναζητήσει ταινίες που βρίσκονται σε streaming πλατφόρμες (Netflix, Hulu, Amazon Prime, Disney+).

Ο χρήστης μπορεί να επιλέξει ως input διάφορα χαρακτηριστικά ταινιών και πολλαπλές streaming πλατφόρμες, ώστε να βρει τις ταινίες που ψάχνει.

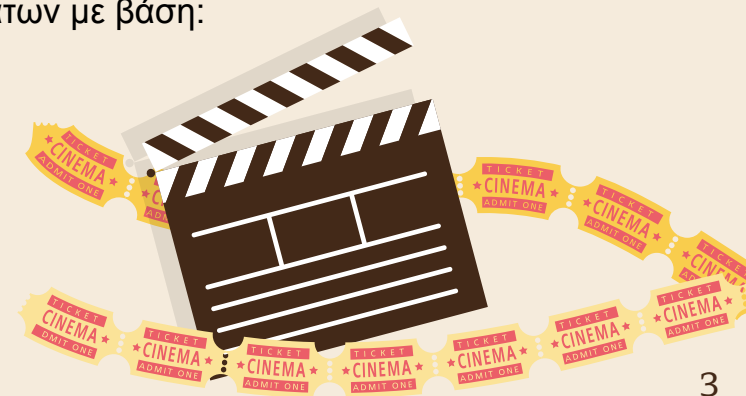
Επίσης επιστρέφονται και στατιστικά για το πλήθος των ταινιών κάθε πλατφόρμας σε μορφή γραφήματος πίτας.

Η εφαρμογή είναι βασισμένη πάνω σε μια βάση δεδομένων με πάνω από 16.000 ταινίες με πληροφορίες για αυτές και επίσης γίνεται σύνδεση με εξωτερικό δωρεάν API ταινιών [TheMovieDB](https://www.themoviedb.org/), από το οποίο παίρνουμε αφίσες για κάθε ταινία που εμφανίζουμε.

# Κύρια Λειτουργικά Χαρακτηριστικά

## Είσοδος

- Η είσοδοι της εφαρμογής είναι είτε:
  - Streaming πλατφόρμα (υποχρεωτικό)
  - Τίτλος ταινίας
  - Όνομα σκηνοθέτη
  - Γλώσσα
  - Είδος
  - Χρονολογία
  - Χώρα παραγωγής
  - Ηλικία
- Καθώς και ο τρόπος αναπαράστασης της σειράς των αποτελεσμάτων με βάση:
  - Τίτλο ταινίας
  - Όνομα σκηνοθέτη
  - Χρονολογία
  - Διάρκεια ταινίας
  - Βαθμολογία στο IMDb
  - Βαθμολογία στο Rotten Tomatoes
  - Αριθμός αποτελεσμάτων ανα σελίδα



# Κύρια Λειτουργικά Χαρακτηριστικά

## Έξοδος

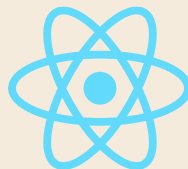
- Η έξοδος της εφαρμογής είναι:
  - Οι αντίστοιχες ταινίες με βάση το input του χρήστη.
  - Σε ποιες πλατφόρμες είναι διαθέσιμες.
  - Φωτογραφία για κάθε ταινία, με σύνδεση με το το API ταινιών [TheMovieDB](https://www.themoviedb.org/).
  - Πληροφορίες για την ταινία όπως (Τίτλος ταινίας, Όνομα σκηνοθέτη, Χρονολογία, Είδος, Γλώσσα, Χώρα παραγωγής, Διάρκεια ταινίας, Βαθμολογία στο IMDb, Βαθμολογία στο Rotten Tomatoes, Ηλικία)
  - Γράφημα με το πλήθος των ταινιών ανά πλατφόρμα.



# Τεχνολογίες Υλοποίησης

## Database

MySQL  
Ver 14.14



## Front End

React  
*create-react-app 3.3.0*

## Back End

Node.js  
v12.14.0



HTML



CSS



## Styling

HTML / CSS  
Bootstrap v4.5.2

## Dataset

By [kaggle](https://www.kaggle.com), scraped from  
[Reelgood.com](https://reelgood.com)



## Charts

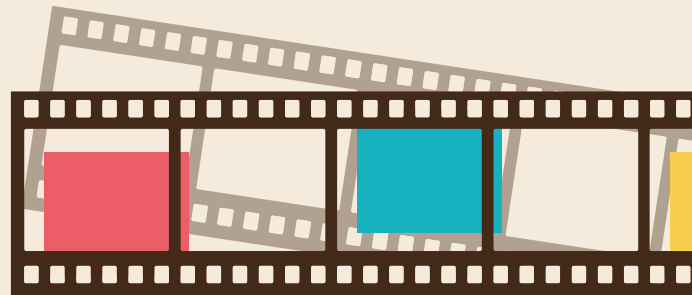
ChartJS

# Οδηγίες Εγκατάστασης

## Προαπαιτούμενα

- **MySQL ή MariaDB server** (> v14.0)
  - Εγκατάσταση για Windows από [εδώ](#) μέσω του XAMPP.
  - Εγκατάσταση για Linux / Windows από τα Official Guides [εδώ](#).
- **Node.js** (> v10.0)
  - Εγκατάσταση για Windows/macOS από [εδώ](#).
  - Εγκατάσταση για Linux μέσω packet manager από [εδώ](#).

Οι οδηγίες εγκατάστασης υπάρχουν αναλυτικά και στο [README](#) στο *GitHub*.



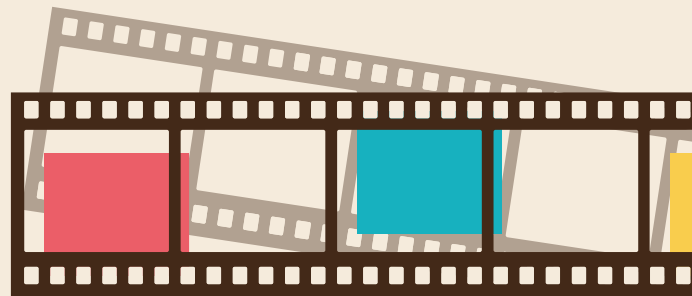
# Οδηγίες Εγκατάστασης

## Κατέβασμα Εφαρμογής

Σε κάποιο φάκελο του συστήματος κάνουμε clone, και κάνουμε cd στον φάκελο της εφαρμογής:

```
git clone https://github.com/thanoskoutr/Appathon-NTUA.git
```

```
cd Appathon-NTUA
```



# Οδηγίες Εγκατάστασης

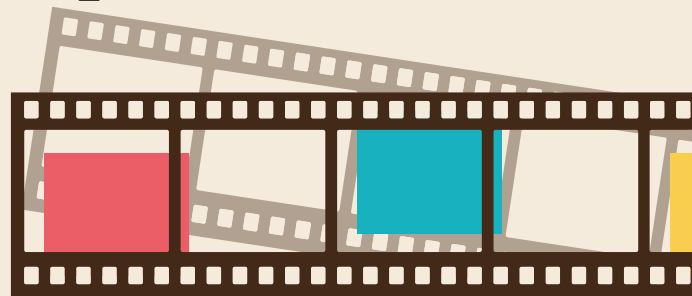
## Εγκατάσταση της Βάσης

Στον φάκελο της εφαρμογής τρέχουμε την παρακάτω εντολή, που δημιουργεί την βάση και τον κατάλληλο πίνακα:

```
mysql -u root -p < ./database/Movies.sql
```

Έπειτα, τρέχουμε την παρακάτω εντολή που περνάει όλα τα δεδομένα στην βάση:

```
mysql -u root -p appathon_03116073 < ./database/appathon_dump.sql
```





# Οδηγίες Εγκατάστασης

## Ρύθμιση Μεταβλητών Περιβάλλοντος

Ανοίγουμε το αρχείο `./back-end/.env.example` ώστε να εισαχθούν οι κατάλληλες τιμές στις μεταβλητές περιβάλλοντος, για την σύνδεση με τη βάση δεδομένων, καθώς και το API Key για το TheMovieDB API.

### Σύνδεση με την βάση

Αντικαθιστούμε τα `DB_USER`, `DB_PASS` με τα credentials του root χρήστη της βάσης:

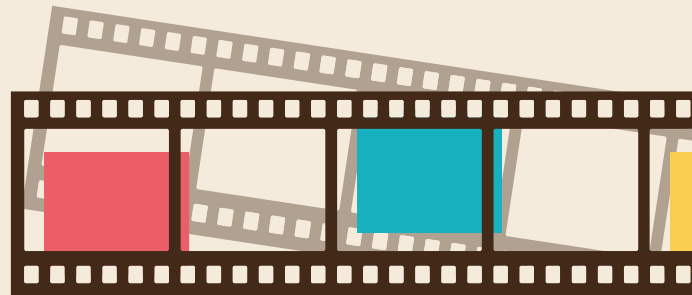
```
DB_HOST=localhost
```

```
DB_USER=root
```

```
DB_PASS=
```

```
DB_NAME=appathon_03116073
```

```
TMDB_API_KEY=
```



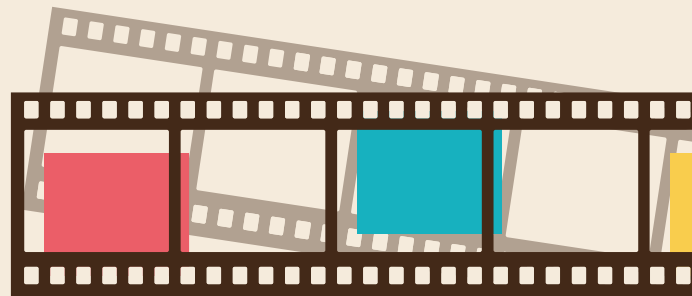
# Οδηγίες Εγκατάστασης

## Ρύθμιση Μεταβλητών Περιβάλλοντος

### Σύνδεση με το API του TMDB

Για το API key του The Movie DB, θα πρέπει να δημιουργηθεί ένας λογαριασμός στο [TMDB](#) και από κει στις ρυθμίσεις του λογαριασμού θα πρέπει να γίνει αίτηση για ένα API key.

Αλλιώς αν δεν προστεθεί κανένα κλειδί στην μεταβλητή `TMDB_API_KEY`, απλά στην εφαρμογή δεν θα εμφανίζονται οι αφίσες των ταινιών.



# Οδηγίες Εγκατάστασης

## Ρύθμιση Μεταβλητών Περιβάλλοντος

### Αλλαγή ονόματος αρχείου

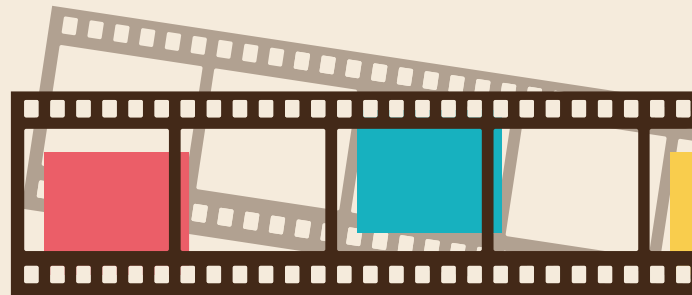
Αφού τα αλλάξουμε κατάλληλα, το αποθηκεύουμε ως `.env`, ώστε να λειτουργήσει σωστά:

#### Linux/macOS:

```
mv ./back-end/.env.example ./back-end/.env
```

#### Windows:

```
RENAME .\back-end\.env.example .env
```



# Οδηγίες Εγκατάστασης

## Αυτόματη Εκκίνηση Εφαρμογής

Για να ξεκινήσει η εφαρμογή αρκεί να τρέξουμε το παρακάτω script σε κάποιο terminal ανάλογα το λειτουργικό σύστημα που έχουμε:

### Linux/macOS:

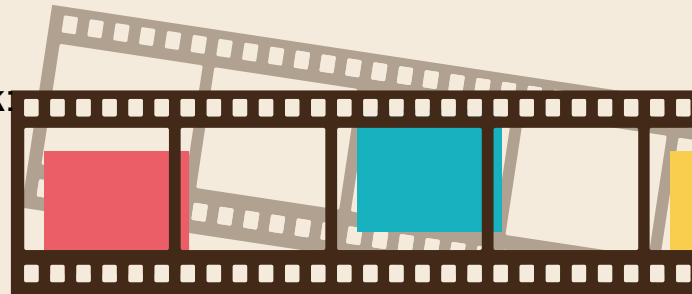
```
./deploy.sh
```

### Windows:

```
deploy.bat
```

Όταν τελειώσει το deploy script, η εφαρμογή θα είναι διαθέσιμη στο **link**:

<http://localhost:5000>



# Οδηγίες Εγκατάστασης

## Χειροκίνητη Εκκίνηση Εφαρμογής

Σε περίπτωση σφάλματος με την αυτόματη εκκίνηση, οι εντολές για να τρέξει η εφαρμογή είναι οι παρακάτω:

### Εκκίνηση back-end:

```
cd ./back-end
```

```
npm install
```

```
npm start
```

### Εκκίνηση front-end:

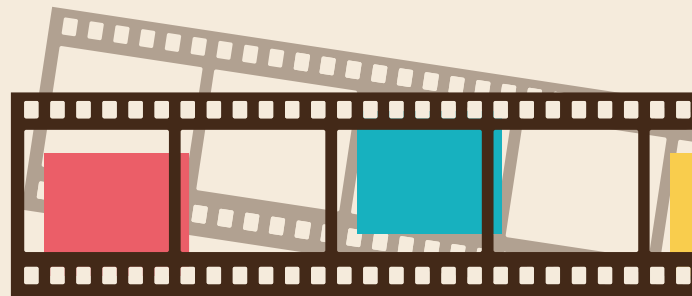
```
cd ./front-end
```

```
yarn install
```

```
yarn build
```

```
npm install -g serve
```

```
serve -s build -l 5000
```



# Επεξήγηση Αρχιτεκτονικής

Η δομή της εφαρμογής έχει χωριστεί σε **3 ξεχωριστά και ανεξάρτητα** κομμάτια μεταξύ τους:

## Database

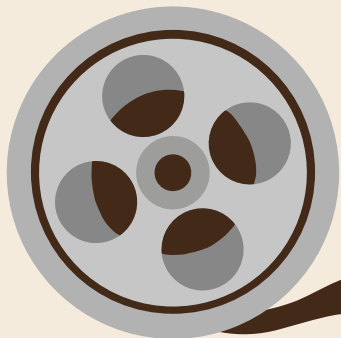
Αποτελείται από την βάση με τον πίνακα με τις ταινίες και τα χαρακτηριστικά τους.

## Back End

Στην ουσία υλοποιείται ένα API, από το οποίο υπάρχουν κατάλληλα Endpoints ώστε να επικοινωνούν εξωτερικές εφαρμογές (π.χ. Το Front End) και να τους στέλνονται τα ζητούμενα δεδομένα από την βάση σε μορφή JSON.

## Front End

Είναι μία εφαρμογή client side, που έχει διάφορες φόρμες και κουμπιά για τις εισόδους του χρήστη και παρέχει ως έξοδο ταινίες και στατιστικά που ζητάει ο χρήστης, με κατάλληλες κλήσεις στο Back End API.





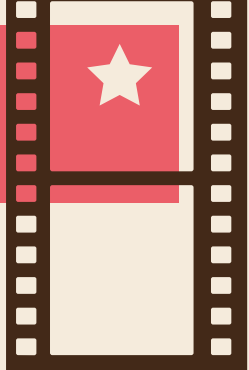
# ***DATABASE***

Η βάση αποτελείται από έναν πίνακα με ονομασία **Movies**, ο οποίος δημιουργήθηκε από κάποιες από τις αντίστοιχες στήλες που είχε το .csv αρχείο που χρησιμοποιήθηκε ως το dataset της εφαρμογής.

Παρακάτω φαίνονται μερικές στήλες και δεδομένα του dataset:

ID	Title	Year	Age	IMDb	Rotten To...	Netflix	Hulu	Prime Vid...	Disney+	Directors
1	Inception	2010	13+	8.8	87%	1	0	0	0	Christopher Nolan
2	The Matrix	1999	18+	8.7	87%	1	0	0	0	Lana Wachowski, Lilly Wachowski
3	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	Anthony Russo, Joe Russo
4	Back to the Future	1985	7+	8.5	96%	1	0	0	0	Robert Zemeckis
5	The Good, the Bad and the Ugly	1966	18+	8.8	97%	1	0	1	0	Sergio Leone

Το κύριο χαρακτηριστικό αυτού του πίνακα, είναι τα πεδία Netflix, Hulu, Prime\_Video, Disney τα οποία παίρνουν τιμές 0 ή 1, αν μια ταινία βρίσκεται στην αντίστοιχη streaming πλατφόρμα.



# ***DATABASE***

Οπότε τοπικά στην MySQL βάση δημιουργήθηκε ο πίνακας με τα πεδία που φαίνονται παρακάτω με τους κατάλληλους τύπους τους.

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
Title	varchar(255)	YES		NULL	
Year	int(4)	YES		NULL	
Age	varchar(255)	YES		NULL	
IMDb	varchar(255)	YES		NULL	
Rotten_Tomatoes	varchar(255)	YES		NULL	
Netflix	tinyint(1)	NO		NULL	
Hulu	tinyint(1)	NO		NULL	
Prime_Video	tinyint(1)	NO		NULL	
Disney	tinyint(1)	NO		NULL	
Directors	varchar(255)	YES		NULL	
Genres	varchar(255)	YES		NULL	
Country	varchar(255)	YES		NULL	
Language	varchar(255)	YES		NULL	
Runtime	int(11)	YES		NULL	



# BACK END - ROUTES

Το Back End, είναι στημένο ως ένα Node-Express App σε Javascript. Η δομή του είναι βασισμένη στο μοντέλο **MVC** (Model - View - Controller). Για αυτό, όπως φαίνεται και παρακάτω ο φάκελος **back-end** χωρίζεται στους υποφακέλους: **controllers**, **models**, **routes**.

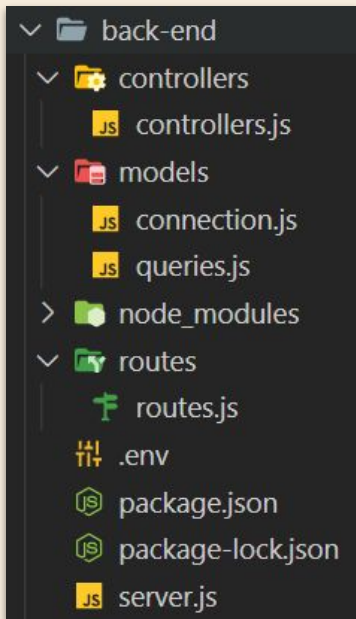
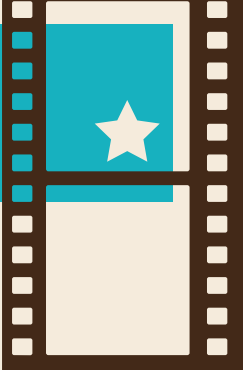
Το αρχείο **server.js**, ουσιαστικά εκκινεί την εφαρμογή και ξεκινά να δέχεται αιτήματα από την **localhost:8000**. Τα αιτήματα προσπαθούν να αντιστοιχιστούν με κάποιο route.

Τα endpoints του API φαίνονται στο αρχείο **routes.js**, το οποίο ουσιαστικά καθορίζει όλα τα routes της εφαρμογής, τα οποία είναι τα παρακάτω:

```
router.get('/TMDB/configuration', controller.TMDBConfiguration);
router.get('/TMDB/search/movie', controller.TMDBSearchMovie);

router.get('/platform', controller.SelectAllPlatforms);
router.get('/platform/statistics', controller.SelectPlatformStatistics);
/* query parameters = { offset, limit, orderBy, order, title, director } */
router.get('/platform/:platform', controller.SelectPlatform);
/* operation = { and, or } */
router.get('/platform/:operation/:platform1/:platform2', controller.SelectPlatforms2);
router.get('/platform/:operation/:platform1/:platform2/:platform3', controller.SelectPlatforms3);
router.get('/platform/:operation/:platform1/:platform2/:platform3/:platform4', controller.SelectPlatforms4);

router.get('/Title/', controller.SelectAllTitles);
router.get('/Directors', controller.SelectAllDirectors);
router.get('/Country', controller.SelectAllCountries);
router.get('/Language', controller.SelectAllLanguages);
router.get('/Year', controller.SelectAllYears);
router.get('/Runtime', controller.SelectAllRuntimes);
router.get('/Age', controller.SelectAllAges);
router.get('/IMDb', controller.SelectAllIMDbScores);
router.get('/Rotten_Tomatoes', controller.SelectAllRottenScores);
router.get('/Genres', controller.SelectAllGenres);
```





# BACK END - CONTROLLERS

Αφού το αίτημα, ταιριάζει με κάποιο route, στέλνεται στον αντίστοιχο controller που βρίσκεται στο αρχείο **controllers.js**. Οι controllers, είναι συναρτήσεις που παίρνουν τα αιτήματα ως μια παράμετρο **req** (request) και επιστρέφουν μια απάντηση στην παράμετρο **res** (response). Αυτή η απάντηση, μπορεί να είναι από ένα απλό HTTP Status Code, όπως κάνουμε στην περίπτωση σφάλματος ή κάποιο αρχείο σε μορφή JSON που περιέχει δεδομένα.

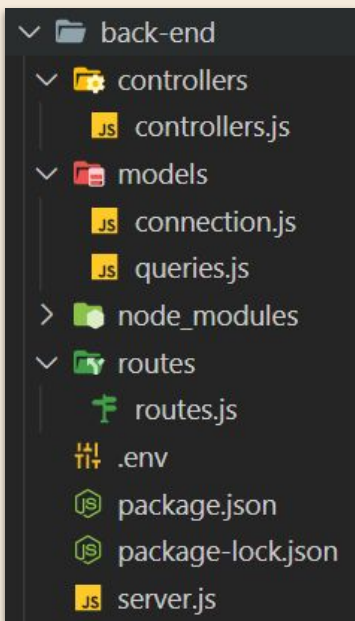
Όλοι οι controllers που έχουν υλοποιηθεί, επιστρέφουν πάντα ένα JSON αρχείο με τα ζητούμενα δεδομένα κάθε αιτήματος, ενώ σε περίπτωση σφάλματος κάποιο Status Code (**400, 403, 404**) με αντίστοιχο κείμενο (**Bad Request, No Data, Not Found**).

Οι controllers, χωρίζονται σε δύο κατηγορίες.

1. Σε αυτούς που κάνουν query στην βάση δεδομένων για να πάρουν τα αποτελέσματα.
2. Σε αυτούς που κάνουν fetch στο API του TMDB (μόνο στα routes **/TMDB/...**).

Επίσης αυτοί που κάνουν query στην βάση μπορούμε να τους χωρίσουμε πάλι σε δύο κατηγορίες:

- A. Σε αυτούς που γυρνάνε όλα τα δεδομένα ενός πεδίου του πίνακα (στα routes **/Title, /Directors, ...**).
- B. Σε αυτούς που ψάχνουν ανά πλατφόρμα / πλατφόρμες.





# BACK END - CONTROLLERS

Οι controllers (κατηγορίας 2), που κάνουν fetch στο TMDb API είναι 2 σε αριθμό:

- Ο **TMDBConfiguration()**, χρησιμοποιείται για να πάρει βασικά στοιχεία για τις υπόλοιπες κλήσεις στο API, όπως το βασικό URL path των φωτογραφιών και τις διαθέσιμες αναλύσεις.
- Ο **TMDBSearchMovie()**, χρησιμοποιείται για να γυρίσει ως αποτέλεσμα την φωτογραφία για μια ζητούμενη ταινία σύμφωνα με τον τίτλο ως είσοδο. Αυτό επιτυγχάνεται με δύο διαδοχικές κλήσεις στο API, μία πρώτη για να βρει από τον τίτλο το path της φωτογραφίας, και μία δεύτερη για να κατεβάσει την φωτογραφία και εν τέλει να την στείλει ως απάντηση.

Οι φωτογραφίες αυτές αποθηκεύονται στον server στον φάκελο **photos**. Προς το παρόν όσο περισσότερα αιτήματα γίνονται τόσο αυξάνονται και οι φωτογραφίες στον server. Σε μια πιο ιδανική υλοποίηση θα μπορούσαν να σβήνονται μετά από κάποιο χρονικό διάστημα. Προς τα παρόν γίνεται χειροκίνητα αν σβήσουμε το φάκελο.

```
controller.TMDBConfiguration
controller.TMDBSearchMovie

controller.SelectAllPlatforms
controller.SelectPlatformStatistics

controller.SelectPlatform

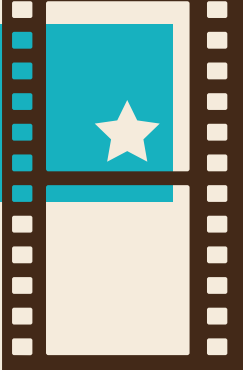
controller.SelectPlatforms2
controller.SelectPlatforms3
controller.SelectPlatforms4

controller.SelectAllTitles
controller.SelectAllDirectors
controller.SelectAllCountries
controller.SelectAllLanguages
controller.SelectAllYears
controller.SelectAllRuntimes
controller.SelectAllAges
controller.SelectAllIMDbScores
controller.SelectAllRottenScores
controller.SelectAllGenres
```

```
fetch(`https://api.themoviedb.org/3/configuration?api_key=${process.env.TMDB_API_KEY}`, {
  method: 'GET',
})
```

```
fetch(`https://api.themoviedb.org/3/search/movie?api_key=${process.env.TMDB_API_KEY}&${title}`, {
  method: 'GET',
})
```

```
fetch(`${base_url}${poster_size}${poster_path}`, {
  method: 'GET',
})
```



# BACK END - CONTROLLERS

Οι controllers (κατηγορίας 1), που κάνουν query στην βάση δεδομένων, όπως είπαμε χωρίζονται στους:

- Οι **SelectAll...()**, οι οποίοι καλούν τις συναρτήσεις της μορφής **SelectAll...Query()**, οι οποίες θα κάνουν τα queries στην βάση δεδομένων για να πάρουν όλα τα δεδομένα από το αντίστοιχο πεδίο και βρίσκονται στο αρχείο **models/queries.js**.
- Οι **SelectPlatform...()**, οι οποίοι καλούν τις συναρτήσεις της μορφής **SelectPlatform...Query()**, οι οποίες θα κάνουν τα queries στην βάση δεδομένων για συγκεκριμένες ταινίες με βάση πολλές παραμέτρους (query parameters) ανάλογα με το σε πόσες πλατφόρμες ψάχνουμε και βρίσκονται στο αρχείο **models/queries.js**.
- Ο **SelectPlatformStatistics()**, ο οποίος καλεί την συνάρτηση **SelectPlatformStatisticsQuery()**, και γυρνάει το πόσες ταινίες βρίσκονται σε κάθε πλατφόρμα και βρίσκεται στο αρχείο **models/queries.js**.

```
controller.TMDBConfiguration
controller.TMDBSearchMovie

controller.SelectAllPlatforms
controller.SelectPlatformStatistics

controller.SelectPlatform

controller.SelectPlatforms2
controller.SelectPlatforms3
controller.SelectPlatforms4

controller.SelectAllTitles
controller.SelectAllDirectors
controller.SelectAllCountries
controller.SelectAllLanguages
controller.SelectAllYears
controller.SelectAllRuntimes
controller.SelectAllAges
controller.SelectAllIMDbScores
controller.SelectAllRottenScores
controller.SelectAllGenres
```

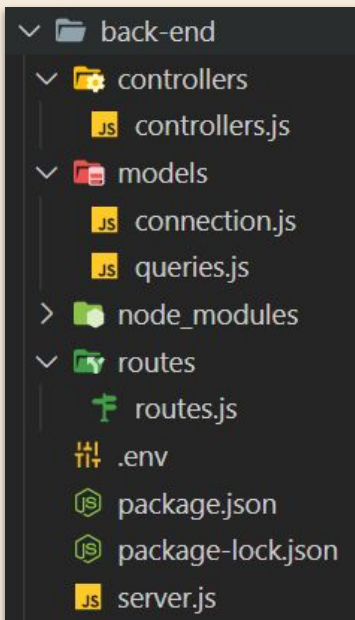
Τα queries αναλύονται παρακάτω στην κατηγορία των **Models**.

# BACK END - MODELS

Ο φάκελος **models**, έχει δύο αρχεία. Το **connection.js** που δημιουργεί την σύνδεση με την βάση δεδομένων και το **queries.js** που περιέχει τα queries που προαναφέρθηκαν.

Η σύνδεση με την MySQL βάση, γίνεται με την βοήθεια του npm πακέτου `mysql` στο οποίο παρέχουμε τις απαραίτητες πληροφορίες για να γίνει η σύνδεση, όπως το `hostname` της βάσης, το `username` και `password` ενός χρήστη της βάσης και το όνομα της βάσης που θέλουμε.

Αυτές οι πληροφορίες επειδή αλλάζουν ανάλογα το σύστημα που βρίσκεται η εφαρμογή και είναι και ευαίσθητες βρίσκονται σε ένα ξεχωριστό `.env` αρχείο το οποίο μας τις παρέχει ως μεταβλητές περιβάλλοντος, όπως φαίνεται παρακάτω:



```
const connection = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASS,
  database: process.env.DB_NAME,
  multipleStatements: true
});
```



# BACK END - MODELS

Στο `queries.js` που περιέχει τα queries που προαναφέρθηκαν, παρακάτω φαίνεται ένα query της μορφής `SelectAll...Query()`:

```
exports.SelectAllDirectorsQuery = (result) => {  
  connection.query(`SELECT DISTINCT Directors FROM Movies WHERE Directors IS NOT NULL ORDER BY Directors ASC`,
```

Επίσης παρακάτω φαίνεται ένα query της μορφής `SelectPlatform...Query()`, στο οποίο αναζητάτε ανάμεσα σε δύο πλατφόρμες είτε με AND είτε με OR ώστε να έχει κάποιο από τα χαρακτηριστικά: *Τίτλο*, *Όνομα σκηνοθέτη*, *Γλώσσα*, *Είδος*, *Χρονολογία*, *Χώρα*, *Ηλικία* και να εμφανιστεί με κάποια *σειρά* και με κάποιο *όριο* και *offset*. Προφανώς μπορεί να είναι κενά ή όλα ή κάποια από αυτά.

```
exports.SelectPlatforms2Query = (operation, platform1, platform2, offset, limit, orderBy, order, title, director, language, genre, year, country, age,  
  connection.query(`SELECT * FROM Movies WHERE (${platform1}=1 ${operation} ${platform2}=1) AND Title LIKE "%${title}%"  
    AND Directors LIKE "%${director}%"  
    AND Language LIKE "%${language}%"  
    AND Genres LIKE "%${genre}%"  
    AND Year LIKE "%${year}%"  
    AND Country LIKE "%${country}%"  
    AND Age LIKE "%${age}%"  
    ORDER BY ${orderBy}='', ${orderBy} ${order} LIMIT ${limit} OFFSET ${offset}`, (err, res) => {
```



# FRONT END

Το Front End, είναι φτιαγμένο με το framework **React** σε Javascript. Η δομή του είναι βασισμένη στο template της **create-react-app**, όπου υπάρχει ένας φάκελος **public** ο οποίος περιέχει το **index.html** που είναι η σελίδα που εμφανίζεται στον browser και ένας φάκελος **src**, ο οποίος έχει όλα τα javascript αρχεία που γράφουμε για την React.

Το αρχείο **index.js**, ουσιαστικά είναι αυτό που περιέχει όλη την εφαρμογή στο front end, καθώς καλεί το **component App (App.js)**, το οποίο με τη σειρά του είναι το ανώτερο component στην εφαρμογή.

```
ReactDOM.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>,  
  document.getElementById('root')  
)
```

Η σύμβαση που έχει ακολουθηθεί είναι ότι κάθε component βρίσκεται σε ξεχωριστό Javascript αρχείο, οπότε όταν αναφερόμαστε στο App Component βρίσκεται στο App.js, και αντίστοιχα όταν αναφερόμαστε στο αρχείο App.js, αναφερόμαστε στο App Component.

# FRONT END

Από το **App.js**, έχουμε τον τίτλο και μια περιγραφή της εφαρμογής μας και καλούμε το **GetConfigurationTMDB** component, το οποίο χρησιμοποιείται για να κάνει κλήση στο TMDB API, για να πάρει τα κατάλληλα configurations που χρειαζόμαστε (**base\_url**, **poster\_sizes**, **poster\_size**). Αυτό θα γίνει μία φορά, όταν φορτώσει η σελίδα, καθώς το fetch γίνεται μέσα στην συνάρτηση **ComponentDidMount()**.

## Movie Selector

This is a small web app, where you can search for movies available on Streaming platforms. You can search and order the movies based on the given attributes below. You can search on multiple Streaming platforms by selecting more than one icon.

Από κει, καλούμε το component **BasicForm**, το οποίο περιέχει την κύρια φόρμα της εφαρμογής στην οποία βάζει ο χρήστης της εισόδους που θέλει.

The screenshot shows the 'Movie Selector' web application. At the top, there's a title 'Select Streaming Platform' followed by four icons: Netflix, Hulu, Prime Video, and Disney+. Below this, there are several search filters: 'Search by Movie Title:' with a text input, 'Search by Director:' with a text input, 'Search by Language:' with a dropdown menu, 'Search by Genre:' with a dropdown menu, 'Search by Year:' with a dropdown menu, 'Search by Country:' with a dropdown menu, and 'Search by Age:' with a dropdown menu. At the bottom, there are three more controls: 'Order:' with a dropdown menu (set to 'Ascending'), 'Sort by:' with a dropdown menu (set to 'Title'), and 'Results per Page:' with a dropdown menu (set to '20'). There are two buttons at the bottom: 'Submit' and 'Statistics'.



# FRONT END

Στο **BasicForm**, έχουμε την κύρια λειτουργία της εφαρμογής μας στην οποία κάνουμε fetch τις επιλογές για τα select inputs (**Language, Genre, Year, Country, Age**), κρατάμε κάθε αλλαγή στις εισόδους του χρήστη μέσω των συναρτήσεων **handlePlatformChange**, **handleChange** για την μπάρα επιλογής πλατφόρμας (**ToggleButtonGroupControlled.js**) και τα select και input fields αντίστοιχα.

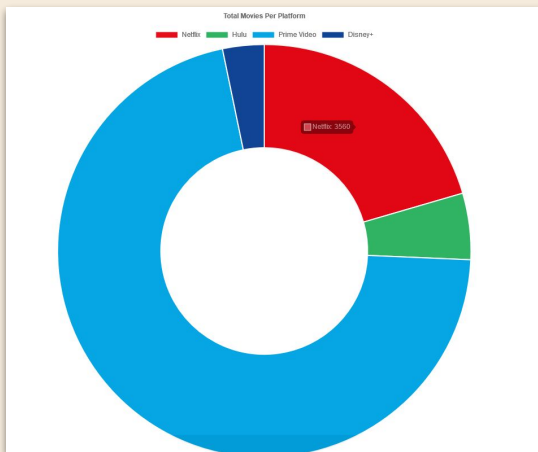
Επίσης, όταν ο χρήστης πατήσει το Submit (ή Statistics) οι είσοδοι που έχει επιλέξει στέλνονται με το κατάλληλο fetch στο back-end, το οποίο επιστρέφει τα αποτελέσματα και τα εμφανίζεται μέσω του component **ShowResults** (ή **ShowStats** αντίστοιχα).

The screenshot shows a web application titled "BasicForm". At the top, there's a section "Select Streaming Platform" with four buttons: NETFLIX, hulu, prime video, and Disney+. Below this, there are several search filters: "Search by Movie Title:" with a text input, "Search by Director:" with a text input, "Search by Language:" with a dropdown menu, "Search by Genre:" with a dropdown menu, "Search by Year:" with a dropdown menu, "Search by Country:" with a dropdown menu, and "Search by Age:" with a dropdown menu. At the bottom, there are three buttons: "Submit", "Statistics", and a "Show Results" button (partially visible). The "Submit" button is highlighted in blue.

# FRONT END

Στο **ShowResults**, για κάθε αποτέλεσμα που δεχόμαστε καλούμε το component **GetPhotoTMDB**, ώστε να ψάξουμε με βάση τον τίτλο φωτογραφία για αυτήν την ταινία και να εμφανίσουμε τα αποτελέσματα μαζί με την φωτογραφία σε μία μορφοποιημένη κάρτα.

Στο **ShowStats**, αφού έχουμε τα αποτελέσματα από το fetch στο αντίστοιχο endpoint του back-end, απλά τα παρουσιάζουμε σε μορφή πίτας. Σε περίπτωση μεγαλύτερου χρονικού πλαισίου, θα είχε γίνει και παραμετροποίηση των στατιστικών ώστε να εμφανίζονται σε κάθε αποτέλεσμα και όχι μόνο συνολικά για όλη την βάση.



Chris D'Elia: No Pain

**Director:** Matt D'Elia

**Year:** 2020

**Genre:** Comedy

**Language:** English

**Country:** United States

**Runtime:** 55

**IMDb Score:** 6

**Rotten Tomatoes Score:** No Data

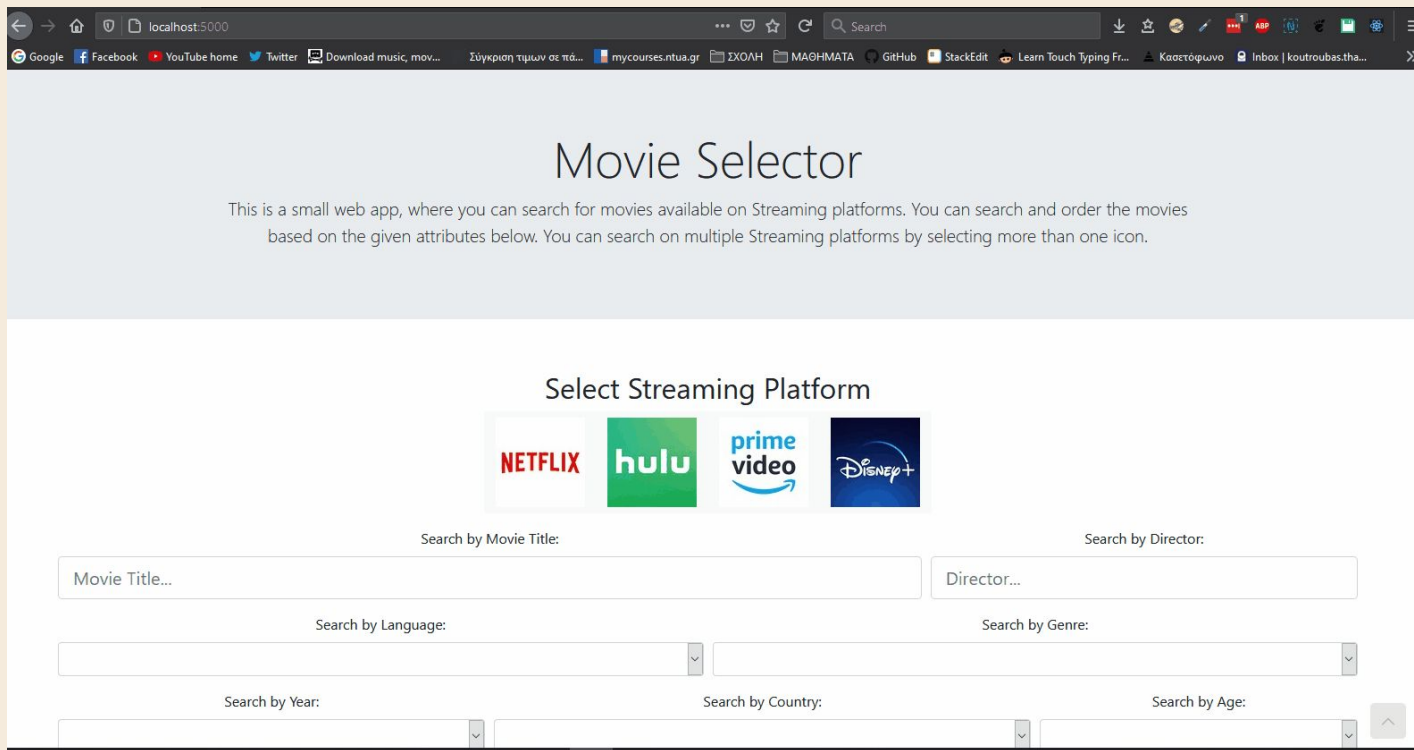
**Age:** No Data

**Platforms:** [Netflix](#)

[Find More](#)

# Παραδείγματα Χρήσης

Επιλέγουμε δύο πλατφόρμες, π.χ. Netflix και Disney+ χωρίς καμία άλλη παράμετρο, με φθίνουσα σειρά με βάση την χρονολογία.

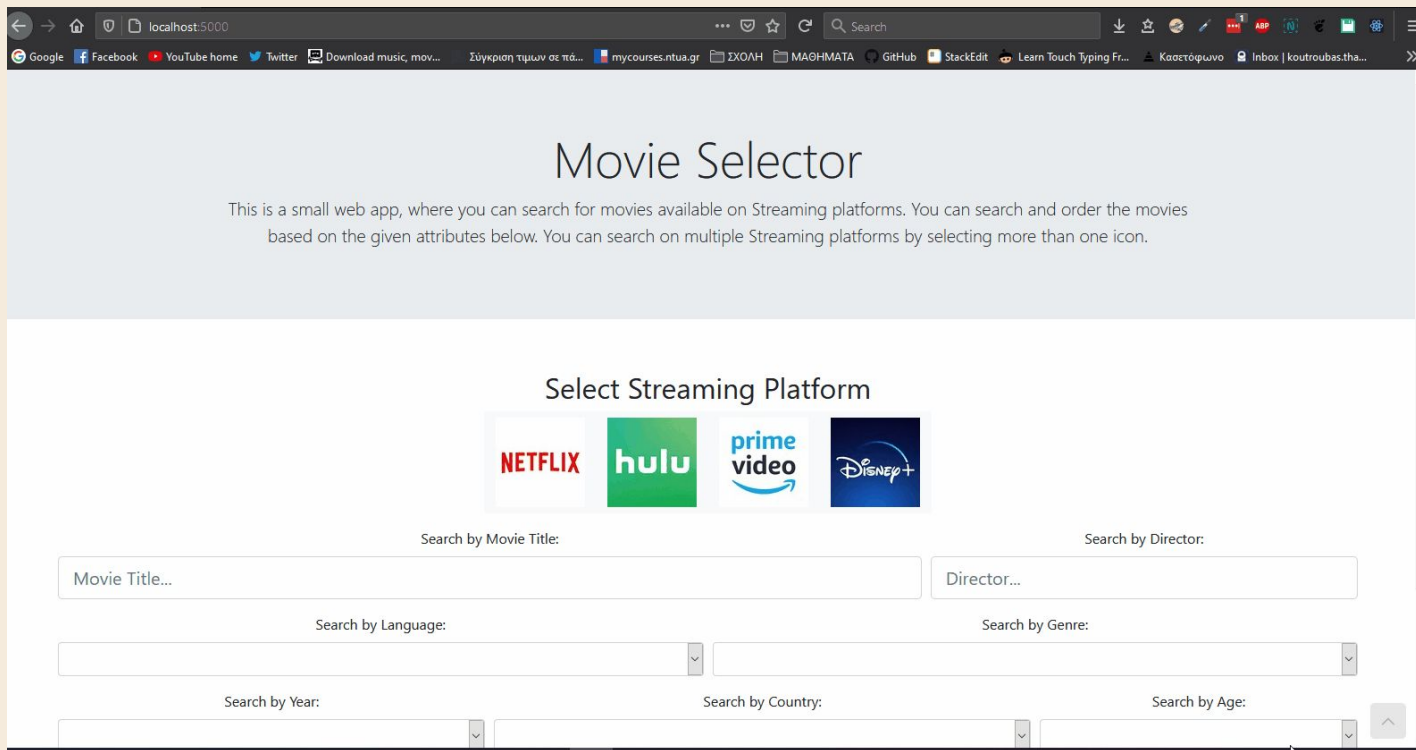


The screenshot shows a web browser window displaying the 'Movie Selector' application. The browser's address bar shows 'localhost:5000'. The application's header has a light blue background with the title 'Movie Selector' and a descriptive paragraph: 'This is a small web app, where you can search for movies available on Streaming platforms. You can search and order the movies based on the given attributes below. You can search on multiple Streaming platforms by selecting more than one icon.' Below the header, there is a section titled 'Select Streaming Platform' with four icons for Netflix, Hulu, Prime Video, and Disney+. Underneath, there are search filters: 'Search by Movie Title:' with a text input field, 'Search by Director:' with a text input field, 'Search by Language:' with a dropdown menu, 'Search by Genre:' with a dropdown menu, 'Search by Year:' with a dropdown menu, 'Search by Country:' with a dropdown menu, and 'Search by Age:' with a dropdown menu. A blue and white director's chair icon is visible on the left side of the browser window.

*Είναι GIF η εικόνα, οπότε μπορεί να μην παίζει.*

# Παραδείγματα Χρήσης

Επιλέγουμε όλες τις πλατφόρμες, και ψάχνουμε τίτλο ταινίας π.χ. Django και βάζουμε τα αποτελέσματα να εμφανίζονται με φθίνουσα σειρά με βάση τις κριτικές του IMDb.



*Είναι GIF η εικόνα, οπότε μπορεί να μην παίζει.*

# Παραδείγματα Σφαλμάτων

Αν δεν επιλέξουμε καμία πλατφόρμα θα μας βγάλει μήνυμα να επιλέξουμε μία τουλάχιστον είσοδο, γιατί δεν μπορεί να ψάξει ταινίας που δεν ανήκουν πουθενά.

Select Streaming Platform

NETFLIX

hulu

prime video

Disney+

Search by Movie Title:

Search by Director:

Movie Title...

Director...

Search by Language:

Search by Genre:

Search by Year:

Search by Country:

Search by Age:

Order:

Ascending

Sort by:

Title

Results per Page:

20

Submit

Statistics

Select Inputs

# Παραδείγματα Σφαλμάτων

Αν δεν βρεθεί κάποιο αποτέλεσμα αναζήτησης πάλι εμφανίζεται μήνυμα που μας ενημερώνει ότι δεν υπάρχει επιπλέον αποτελέσματα.

Search by Movie Title:

Τσιου

Search by Director:

Director...

Search by Language:

Search by Genre:

Search by Year:

Search by Country:

Search by Age:

Order:

Ascending

Sort by:

Title

Results per Page:

20

Submit

Page: 1

Previous Page

Statistics

No Data Found

# Παραδείγματα Σφαλμάτων

Επίσης αν δεν βρεθεί κάποια εικόνα για κάποια ταινία, τότε εμφανίζεται μια default φωτογραφία που ενημερώνει ότι δεν υπάρχει διαθέσιμη εικόνα για την ταινία.



15-Aug

**Director:** Swapnaneel Jaykar

**Year:** 2019

**Genre:** Drama

**Language:** No Data

**Country:** India

**Runtime:** 124

**IMDb Score:** 5.8

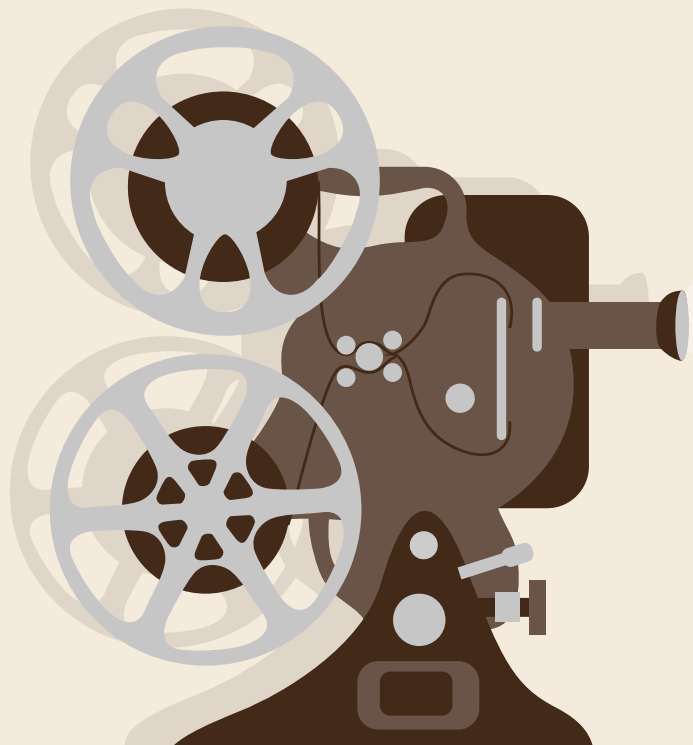


18 Presents

**Director:** Francesco Amato

**Year:** 2020

**Genre:** Drama



# ***THANKS***

Κουτρούμπας Αθανάσιος  
03116073



Project at:

<https://github.com/thanoskoutr/Appathon-NTUA>

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

Please keep this slide for attribution.