# Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

**Μεταπτυχιακή Διατριβή**

| | |
|---|---|
| Τίτλος Διατριβής | FULL STACK ΣΥΣΤΗΜΑ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ ΓΙΑ ΜΕΤΑΣΧΗΜΑΤΙΣΜΟ ΕΙΚΟΝΩΝ ΣΕ ΑΝΑΠΑΡΑΣΤΑΣΕΙΣ CARTOON<br><br>FULL STACK ARTIFICIAL INTELLIGENCE SYSTEM TO TRANSFORM IMAGES TO CARTOON REPRESENTATION |
| Ονοματεπώνυμο Φοιτητή | Τάντη Στυλιανή |
| Πατρώνυμο | Παναγιώτης |
| Αριθμός Μητρώου | ΜΠΣΠ17065 |
| Επιβλέπων | Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής |

Ημερομηνία Παράδοσης **Μάιος 2021**

Τριμελής Εξεταστική Επιτροπή

| (υπογραφή) | (υπογραφή) | (υπογραφή) |
|---|---|---|
| Αλέπης Ευθύμιος<br>Αναπληρωτής Καθηγητής | Βίρβου Μαρία<br>Καθηγήτρια | Πατσάκης Κωνσταντίνος<br>Αναπληρωτής Καθηγητής |

# Περιεχόμενα

**Πίνακας εικόνων** ..... **3**

**Περίληψη** ..... **4**

**Abstract** ..... **5**

**Πίνακας Συντομεύσεων** ..... **6**

**1. Εισαγωγή** ..... **7**

**2. Ανασκόπηση πεδίου** ..... **8**

*2.1. Νευρωνικά Δίκτυα* ..... *8*

*2.2. Tensor Flow* ..... *10*

*2.3. Παρόμοιες εφαρμογές* ..... *10*

**3. Παρουσίαση και χρήση** ..... **12**

**4. Αρχιτεκτονική** ..... **16**

*4.1. Υποσύστημα 1 – User Interface – Android App* ..... *17*

*4.2. Server Side υποσυστήματα* ..... *21*

*4.2.1. Υποσύστημα 2 – Web server* ..... *21*

*4.2.1.1 Hypertext Transfer Protocol - HTTP* ..... *22*

*4.2.1.2 HTTP Request* ..... *23*

*4.2.1.3 HTTP Response* ..... *24*

*4.2.2 Web server Code* ..... *26*

*4.2.3 Υποσύστημα 3 – Artificial Intelligence* ..... *28*

*4.3 Versioning – Τεκμηρίωση* ..... *32*

**5 Συμπεράσματα και μελλοντικές επεκτάσεις** ..... **34**

**Βιβλιογραφία** ..... **35**

# Πίνακας εικόνων


| | | |
|---|---|---|
| 1 | Εικόνα 1. Υποσυστήματα που υλοποιούνται και δομούν την εφαρμογή | 7 |
| 2.1 | Εικόνα 2. Απεικόνιση φυσικού και τεχνητού νευρώνα | 9 |
| 2.1 | Εικόνα 3. Βιολογικό Νευρωνικό δίκτυο / Τεχνητό Νευρωνικό δίκτυο | 9 |
| 2.3 | Εικόνα 4. Aging Booth / εφαρμογή μεταβολής εικόνα στο πεδίο της ηλικίας. | 11 |
| 2.3 | Εικόνα 5. Εφαρμογές αλλοίωσης χαρακτηριστικών. | 11 |
| 3 | Εικόνα 6. Αρχική εικόνα εφαρμογής | 12 |
| 3 | Εικόνα 7. Περιβάλλον λήψης εικόνας | 12 |
| 3 | Εικόνα 8. Απεικόνιση λήψης | 13 |
| 3 | Εικόνα 9. Ένδειξη αναμονής | 13 |
| 3 | Εικόνα 10. Απεικόνιση τελικής εικόνας | 14 |
| 3 | Εικόνα 11. Μετασχηματισμός σύνθεσης φρούτων. | 14 |
| 3 | Εικόνα 12. Μετασχηματισμός εικόνας σαλονιού. | 15 |
| 3 | Εικόνα 13. Μετασχηματισμός σύνθεσης με λούτρινα ομοιώματα ζώων. | 15 |
| 4 | Εικόνα 14. Υποσυστήματα που υλοποιούνται και δομούν την εφαρμογή | 16 |
| 4 | Εικόνα 15. Τεχνολογίες ανά υποσύστημα | 16 |
| 4.1 | Εικόνα 16. Εργαλεία προσομοίωσης. | 17 |
| 4.1 | Εικόνα 17. App Layout | 18 |
| 4.1 | Εικόνα 18. Διάγραμμα κλάσεων βασικών οντοτήτων εφαρμογής. | 19 |
| 4.1 | Εικόνα 19. Απεικόνιση δομής JAVA όπως οργανώνεται στο περιβάλλον ανάπτυξης | 19 |
| 4.2 | Εικόνα 20. Γραφική απεικόνιση υποσυστημάτων που φιλοξενεί το Linux VM | 21 |
| 4.2.1.1 | Εικόνα 21. Σχέδιο στοιχείων επικοινωνίας HTTP | 22 |
| 4.2.1.2 | Εικόνα 22. Παράδειγμα Request Headers HTTP | 23 |
| 4.2.1.3 | Εικόνα 23. Παράδειγμα GET HTTP Request | 25 |
| 4.2.1.3 | Εικόνα 24. Response του παραπάνω παραδείγματος | 25 |
| 4.2.2 | Εικόνα 25. Κεντρικό αρχείο κώδικα υποσυστήματος 2 / Typescript | 27 |
| 4.2.3 | Εικόνα 26. Επίπεδα και αλληλεπίδραση στοιχείων του μοντέλου | 28 |
| 4.2.3 | Εικόνα 27. Ζεύγος εκπαίδευσης / οδός | 29 |
| 4.2.3 | Εικόνα 28. Ζεύγος εκπαίδευσης / φαγητά | 29 |
| 4.2.3 | Εικόνα 29. Ζεύγος εκπαίδευσης / εστιατόριο | 29 |
| 4.2.3 | Εικόνα 30. Ζεύγη εκπαίδευσης / άνθρωποι | 30 |
| 4.2.3 | Εικόνα 31. Ζεύγη εκπαίδευσης / φιγούρες | 30 |
| 4.2.3 | Εικόνα 32. Ζεύγη εκπαίδευσης / πρόσωπα | 30 |
| 4.2.3 | Εικόνα 33. Κεντρικό αρχείο κώδικα υποσυστήματος 3/ Python | 31 |
| 4.3 | Εικόνα 34. Απεικόνιση διακλαδώσεων παράλληλης ανάπτυξης | 32 |

## Περίληψη

Η παρούσα εργασία αποτελεί την διπλωματική διατριβή στα πλαίσια του μεταπτυχιακού «Προηγμένα συστήματα πληροφορικής» του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς. Το αντικείμενό της βρίσκεται στην τομή των συστημάτων Mobile computing και της Μηχανικής Μάθησης / Τεχνητής νοημοσύνης. Σκοπός είναι η ανάπτυξη εφαρμογής για οικοσύστημα Android η οποία θα δίνει τη δυνατότητα στο χρήστη να λαμβάνει φωτογραφίες με την κάμερα της συσκευής του , να την αποστέλλει στον εξυπηρετητή (server), να υλοποιείται μετασχηματισμός με χρήση εκπαιδευμένου δικτύου Τεχνητής Νοημοσύνης και να επιστρέφει το αποτέλεσμα του μετασχηματισμού. Αντικείμενο του μετασχηματισμού της εφαρμογής είναι η μετατροπή πραγματικής λήψης σε ισοδύναμη Cartoon αναπαράσταση.

## Abstract

The present document is the graduate thesis in the context of Master Degree (M.Sc.) in “Advanced Informatics and Computing Systems” of the Department of Informatics of the University of Piraeus. The objective of this work lies in the joint of Mobile Computing systems and Artificial (AI) domain. The purpose of this dissertation is the development of an Android ecosystem application that will enable the user to capture pictures using his device camera, send it to the system server, apply the transformation using a trained Neural Network and retrieve the result of this transformation. The transformed image will be a Cartoon representation of the real-world image.

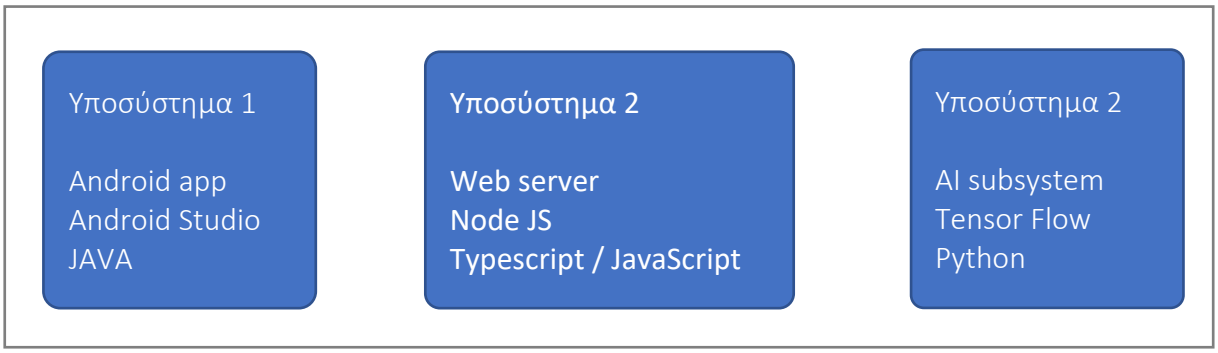
## Πίνακας Συντομεύσεων

| | |
|---|---|
| AI | Artificial Intelligence |
| cGAN | Conditional Generative Adversarial Networks |
| ΝΔ | Νευρωνικά Δίκτυα |
| HTTP | Hypertext Transfer Protocol |
| ΤΝΔ | Τεχνητά Νευρωνικά Δίκτυα |
| VM | Virtual Machine |

# 1. Εισαγωγή

Ο τομέας της πληροφορικής παρουσιάζει ραγδαία εξέλιξη σε όλους τους τομείς που τον αφορούν. Από το υλικό (hardware) που υποστηρίζει όλες τις λειτουργίες τις οποίες αναπτύσσονται για τις ανάγκες των χρηστών, μέχρι συστήματα διαχείρισης πολύπλοκων συστημάτων, αλγορίθμων και μαθηματικών απεικονίσεων. Έτσι δίνεται η δυνατότητα για ανάπτυξη επιλύσεων προβλημάτων ή αναγκών με όλο και πιο σύνθετα μέσα.

Η εργασία αυτή κινείται στον χώρο της τεχνητής νοημοσύνης, υλοποιώντας ένα σύστημα μετασχηματισμού φωτογραφιών του χρήστη, σε εκδόσεις cartoon. Η υλοποίηση αποτελείται από 3 διακριτά υποσυστήματα τα οποία χρησιμοποιών το καθένα διαφορετικές πλατφόρμες και τεχνολογίες.




**Εικόνα 1. Υποσυστήματα που υλοποιούνται και δομούν την εφαρμογή**

Το πρώτο υποσύστημα είναι η εφαρμογή Android η οποία δίνει στο χρήστη τη δυνατότητα να κάνει λήψη φωτογραφίας να τη στείλει στο server για επεξεργασία, να λάβει το αποτέλεσμα και να το απεικονίσει / αποθηκεύσει.

Το δεύτερο υποσύστημα είναι η υπηρεσία web server που αναλαμβάνει να υλοποιήσει όλες τις λειτουργίες μεταφοράς δεδομένων από τη φορητή συσκευή στο server, να την στείλει στο υποσύστημα Artificial Intelligence (AI), να παρακολουθεί της εξέλιξη και να την στείλει στο χρήστη όταν είναι έτοιμη.

Το τρίτο υποσύστημα είναι το υποσύστημα Τεχνητής Νοημοσύνης το οποίο υλοποιεί το μετασχηματισμό της εικόνας. Υλοποιείται με χρήση της πλατφόρμας Tensor Flow σε γλώσσα Python, ενώ το μοντέλο μετασχηματισμού είναι το White-Box-Cartoonization το οποίο μπορεί να βρεθεί στο repository (1)

Η εφαρμογή έχει δομηθεί με έμφαση τον επιμερισμό λειτουργιών (modularity) ώστε να είναι δυνατή η επέκταση. Για παράδειγμα, η εφαρμογή Android είναι “agnostic” ως προς το μοντέλο που υλοποιείται. Άρα αν αλλάξουμε μόνο το υποσύστημα 3, μπορούμε να υλοποιήσουμε διαφορετικό μετασχηματισμό από άλλο σύστημα τεχνητής νοημοσύνης και όχι μόνο. Έτσι μπορούμε να πούμε πως κατά μια έννοια δεν. Μιλάμε για εφαρμογή που υλοποιεί τον συγκεκριμένο μετασχηματισμό, αλλά σύστημα που μπορεί να φιλοξενήσει διαφορετικά μοντέλα με αλλαγή μόνο του υποσυστήματος 3.

# 2. Ανασκόπηση πεδίου

Τα συστήματα τεχνητής νοημοσύνης (2) βρίσκονται όλο και περισσότερο στην καθημερινότητά μας, καθώς καταφέρουν να δώσουν λύση σε προβλήματα που οι κλασσικές αλγοριθμικές προσεγγίσεις αποτυγχάνουν ή έχουν μικρό βαθμό επιτυχίας, ακόμα και στα πιο σημαντικά ζητήματα όπως η πανδημία του Covid19 (3). Επιπροσθέτως, η ανάπτυξη των φορητών συσκευών και η πρόοδος σε επεξεργαστική ισχύ, μνήμη αλλά και δίκτυα, δίνουν τη δυνατότητα στον κάθε ένα να μπορεί να έχει ένα πλήρη υπολογιστή μαζί του , και επιπλέον να μπορεί να εφαρμόζει πολύπλοκα μοντέλα για να ανταποκριθεί σε καθημερινές του ανάγκες. Κλασσικά παραδείγματα αποτελούν τα συστήματα βοήθειας των μεγαλύτερων παγκόσμιων κολοσσών υπηρεσιών πληροφοριακών συστημάτων.

- Google : Google Assistant (4)
- Apple : Siri (5)
- Amazon : Alexa (6)

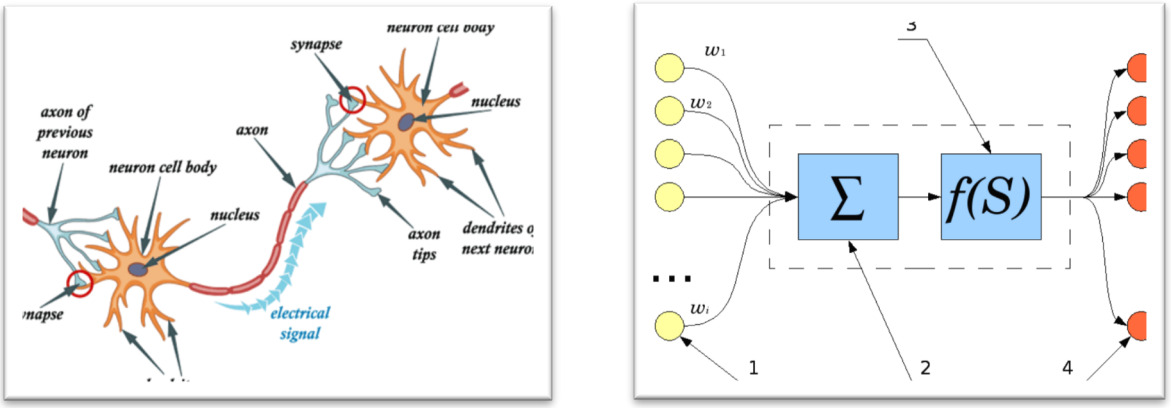
Τα συστήματα αυτά χρησιμοποιούν τεχνητή νοημοσύνη για να κάνουν ένα μεγάλο όγκο λειτουργιών όπως, αναγνώριση φωνής ή επεξεργασία πολυμέσων ενώ τα συναντάμε σε υπολογιστές, κινητές συσκευές αλλά και συσκευές διαχείρισης smart home (Home pod, Google Home, Amazon echo). Φυσικά οι διεργασίες δεν εκτελούνται τοπικά στις συσκευές αυτές αλλά εξυπηρετούνται από κεντρικούς διακομιστές για εξοικονόμηση πόρων. Την προσέγγιση αυτή ακολουθεί και η παρούσα εργασία.

## 2.1. Νευρωνικά Δίκτυα

Η βάση της τεχνητής νοημοσύνης είναι η θεωρία των Νευρωνικών Δικτύων (7). Η ιδέα είναι πως προσπαθούμε να βρούμε τη λύση σε ένα πρόβλημα χωρίς να γνωρίζουμε τη θεωρία που συσχετίζει τα δεδομένα εισόδου και εξόδου, αλλά λειτουργώντας όπως ο εγκέφαλος που προσπαθεί να ανιχνεύσει από τα δεδομένα σχέσεις. Γι’ αυτό και αναφέρονται πολλές φορές ως black box.

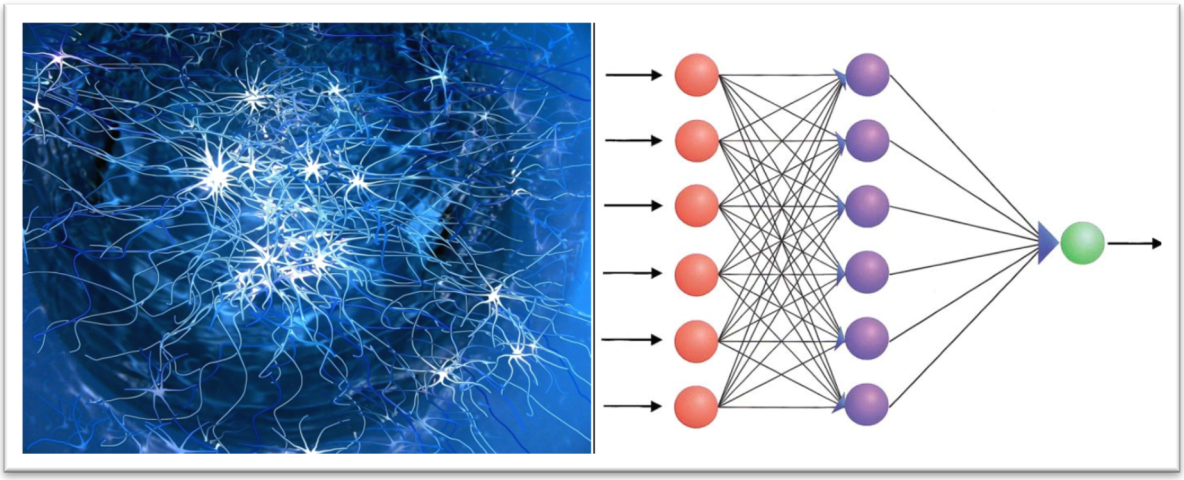
Τα ΝΔ προσομοιάζουν τον τρόπο που διαχειρίζεται ο εγκέφαλος τις πληροφορίες και χρησιμοποιούνται λόγω της ιδιότητάς τους να ανιχνεύουν πολυδιάστατες - μη γραμμικές συναρτήσεις. Η ιδέα ανάπτυξης των Τεχνιτών Νευρωνικών Δικτύων, στηρίζεται στη δομή των βιολογικών Νευρωνικών δομών του εγκεφάλου. Ένα βιολογικό Νευρωνικό Δίκτυο αποτελείται από χημικές συνδέσεις και λειτουργίες Νευρώνων. Κάθε Νευρώνας μπορεί να συνδέεται με πολλούς άλλους και να δημιουργούνται έτσι πολύπλοκα πλέγματα. Εκτιμάται πως ο ανθρώπινος εγκέφαλος αποτελείται από περίπου 10 δις Νευρώνες.

Ο Νευρώνας (Βιολογικός και Τεχνητός) αποτελούν στοιχειώδεις μονάδες επεξεργασίας πληροφορίας. Ο Τεχνητός Νευρώνας μπορεί να γίνει κατανοητός μέσα από τη σύγκριση με τον Βιολογικό. Και στις δύο περιπτώσεις οι συνδέσεις μεταξύ δύο Νευρώνων (Συνάψεις) μεταφέρουν το σήμα και εκεί γίνεται άθροιση όλων των σημάτων εισόδου. Αν η συνολική ένταση της εισόδου ξεπερνάει ένα κατώτατο όριο (κατώφλι θ), ο Νευρώνας γίνεται στιγμιαία ενεργός και παράγει στιγμιαία ένα παλμό ό οποίος μεταφέρεται μέσα από τον Άξονα σε συνάψεις με άλλους Νευρώνες.




**Εικόνα 2. Απεικόνιση φυσικού και τεχνητού νευρώνα**

Σε κάθε περίπτωση, η ουσιώδης διεργασία που συμβαίνει είναι η προσπάθεια ανίχνευσης σχέσης σημάτων εισόδου με γνωστές τιμές εξόδου για να μπορέσει το σύστημα να εκπαιδευτεί. Φυσικά ένα Νευρωνικό Δίκτυο δεν αποτελείται μόνο από ένα Νευρώνα, αλλά από πολλαπλά επίπεδα / συστοιχίες αυτών για να μπορέσει να ανιχνεύσει μη γραμμικές σχέσεις.



**Εικόνα 3. Βιολογικό Νευρωνικό δίκτυο / Τεχνητό Νευρωνικό δίκτυο**

Συνήθως οι νευρώνες είναι οργανωμένοι σε επίπεδα. Έχουμε ένα επίπεδο εισόδου, ένα επίπεδο εξόδου και ενδιάμεσα κρυφά επίπεδα (κανένα, ένα ή και περισσότερα). Στην εικόνα 3 απεικονίζεται ένα δίκτυο 3 επιπέδων (εισόδου, κρυφό, εξόδου).

Εκτός από τον αριθμό των επιπέδων, ένα ΝΔ χαρακτηρίζεται σχετικά με τον τρόπο σύνδεσης από επίπεδο σε επίπεδο. Έτσι διακρίνονται τα Πλήρως συνδεδεμένα ΤΝΔ (fully connected), στα οποία κάθε νευρώνας συνδέεται με όλους τους νευρώνες του επόμενου επιπέδου, και Μερικώς συνδεδεμένα ΤΝΔ (partially connected), στα οποία υπάρχουν νευρώνες που δεν συνδέονται με όλους τους νευρώνες του επόμενου επιπέδου. Ανάλογα με τη φορά σύνδεσης από επίπεδο σε επίπεδο, συναντώνται δύο ομάδες δικτύων που ορίζουν τις δύο μεγάλες κατηγορίες ΤΝΔ. Η πρώτη κατηγορία ορίζεται από τα δίκτυα με απλή ανατροφοδότηση (feedforward) στα οποία δεν υπάρχουν συνδέσεις μεταξύ νευρώνων ενός επιπέδου και νευρώνων προηγούμενου επιπέδου.

Η δεύτερη κατηγορία είναι τα δίκτυα με ανατροφοδότηση (feedback ή recurrent), στα οποία υπάρχουν συνδέσεις μεταξύ νευρώνων ενός επιπέδου και νευρώνων προηγούμενων επιπέδων.

## 2.2. Tensor Flow

Η παραπάνω περιγραφόμενη θεώρηση, στηρίζεται σε μαθηματικό υπόβαθρο το οποίο είναι δύσκολο να κατανοήσει κανείς χωρίς την κατάλληλη εκπαίδευση. Η κοινότητα όμως έχει ανάγκη από τρόπους επίλυσης προβλημάτων που να είναι ρεαλιστικά εφαρμόσιμη. Έτσι εμφανίστηκαν διάφορες προσπάθειες ανάπτυξης συστημάτων επίλυσης Νευρωνικών Δικτύων και λοιπών μεθόδων Τεχνητής Νοημοσύνης από οργανισμούς που έχουν το κατάλληλο δυναμικό. Χαρακτηριστική περίπτωση αποτελεί η πλατφόρμα Tensor Flow που έχει δημιουργήσει η google. Όπως προσδιορίζεται στη σχετική σελίδα, το Tensor Flow είναι μία end-to-end πλατφόρμα ανοιχτού κώδικα για μηχανική μάθηση. Αποτελείται από ένα σύνολο εργαλείων, βιβλιοθηκών και πόρων που επιτρέπει σε ερευνητές να αναπτύξουν εύκολα και να θέσουν σε λειτουργία εφαρμογές μηχανικής μάθησης.

Το Tensor Flow ένα μεγάλο πλήθος από κατηγορίες προβλημάτων όπως ταξινόμηση εικόνων, εμπλουτισμό δεδομένων, κατάτμηση εικόνων, ανίχνευση αντικειμένων, αναγνώρηση κειμένου, εξαγωγή keywords από εικόνες, αναγνώριση ήχου και πολλά ακόμα (8).

Το toolset συνεχώς εμπλουτίζεται με αποτέλεσμα κατά τη συγγραφή να υποστηρίζει Python, Javascript, mobile (Android, iOs) (9).

## 2.3. Παρόμοιες εφαρμογές

Η παρούσα εργασία ασχολείται με την ανάπτυξη συστήματος μετασχηματισμού φωτογραφίας που λαμβάνει ο χρήστης με φορητή συσκευή android σε αντίστοιχη με cartoon στοιχεία. Ο μετασχηματισμός αυτός εντάσσεται στην ευρύτερη οικογένεια των image-to-image μετασχηματισμών. Οι μετασχηματισμοί με τη σειρά τους θεμελιώνονται σε μια οικογένεια δικτύων που λέγεται cGAN (Conditional Generative Adversarial Networks). Η περιγραφή αυτού ξεφεύγει από τα πλαίσια της παρούσας εργασίας (10) (11).

Η πιο κλασική μορφή τέτοιων εφαρμογών έχουν τη μορφή μετασχηματισμού με έμφαση στην ηλικία. Πρόκειται για εφαρμογές που δίνοντας την φωτογραφία του χρήστη παράγεται μια ισοδύναμη που αναφέρεται σε μεγαλύτερη η μικρότερη ηλικία. Μία τέτοια εφαρμογή είναι η “Aging Booth” (12). Στην εικόνα 4 φαίνεται μια αντιπροσωπευτική μετατροπή στο πεδίο της ηλικίας, όπως είναι διαθέσιμη στη σελίδα της εφαρμογής


**Εικόνα 4. Aging Booth / εφαρμογή μεταβολής εικόνα στο πεδίο της ηλικίας.**

Με την ίδια λογική, έχουν αναπτυχθεί και διατίθενται εφαρμογές που μετασχηματίζουν εικόνες σε διάφορα πεδία, όπως για παράδειγμα το βάρος (13) του εικονιζόμενου, την τριχοφυΐα (14) και άλλα. Η εικόνα 5 απεικονίζει τις αντίστοιχες περιπτώσεις. Όλα τα παραπάνω ανήκουν στην ίδια κατηγορία μετασχηματισμών με χρήση Νευρωνικών Δικτύων.

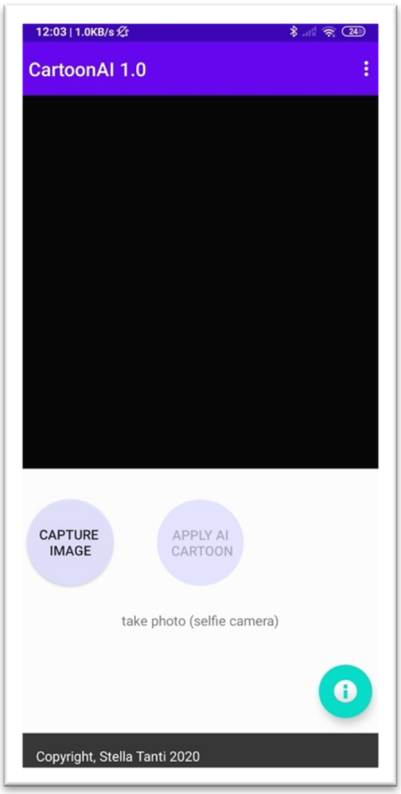

**Εικόνα 5. Εφαρμογές αλλοίωσης χαρακτηριστικών.**

Στη συγκεκριμένη κατηγορία μετασχηματισμού που πραγματεύεται το παρόν, υπάρχουν προϊόντα στο internet τα οποία έχουν παρόμοια λειτουργικότητα όπως:

- Cartoonize.net - https://www.cartoonize.net/onlinecartoonizer/
- ToonApp (15)

# 3. Παρουσίαση και χρήση

Η εφαρμογή αναπτύχθηκε με γνώμονα την απλότητα και ευχρηστία, Το περιβάλλον που καλείται να χειριστεί ο χρήστης είναι δομημένο με τρόπο που να είναι αυτονόητη η ροή ενεργειών που πρέπει να εκτελέσει χρήστης ώστε να φτάσει στο τελικό παραγόμενο προϊόν.

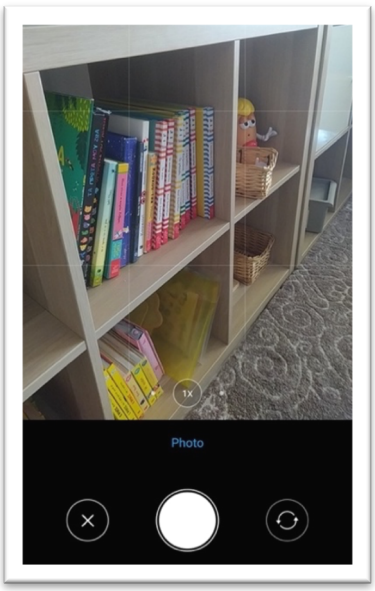



Στην εικόνα απεικονίζεται το περιβάλλον όπως διαμορφώνεται κατά την πρώτη εκκίνηση. Στο επάνω μέρος υπάρχει ο τίτλος και η έκδοση στης εφαρμογής (CartoonAI 1.0). Ακολουθεί χώρος που έχει προβλεφθεί για την απεικόνιση της φωτογραφίας. Κατά την εκκίνηση εμφανίζεται με μαύρο υπόβαθρο για να υποδείξει πως είναι σε κατάσταση αναμονής.

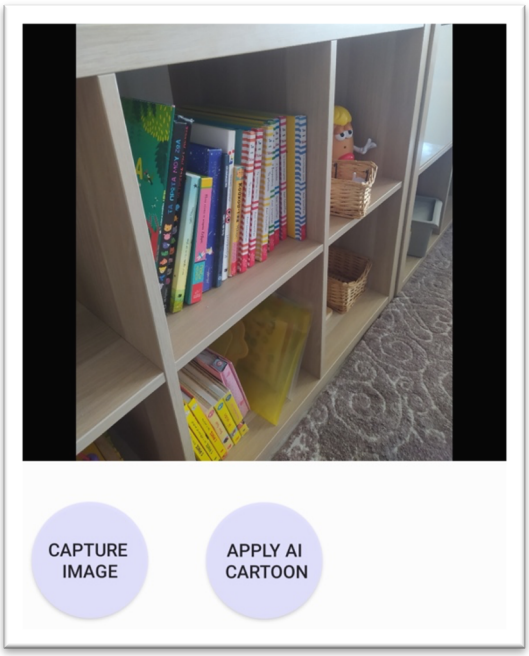
Κάτω από τον υποδοχέα της εικόνας υπάρχουν σε οριζόντια διάταξη δύο κυκλικά κουμπιά. Το πρώτο έχει τίτλο “CAPTURE IMAGE” και πατώντας το ανοίγει η κάμερα για να γίνει λήψη της φωτογραφίας που θα χρησιμοποιηθεί. Το Δεύτερο κουμπί είναι ανενεργό καθώς χρησιμοποιείται αφού γίνει λήψη της φωτογραφίας. Στο κάτω μέρος υπάρχει ένα κουμπί πληροφοριών, που όταν το πατήσουμε εμφανίζει μπάρα με πληροφορίες copyright.

**Εικόνα 6. Αρχική εικόνα εφαρμογής**

Η πρώτη ενέργεια που καλείται να κάνει ο χρήστης είναι να πατήσει το κουμπί για τη λήψη της φωτογραφίας. Αμέσως ανοίγει το περιβάλλον λήψης, το οποίο είναι το προεπιλεγμένο της κάθε συσκευής. Οι τυπικές ρυθμίσεις είναι η εναλλαγή μεταξύ εμπρόσθιας και οπίσθιας κάμερας. Ανάλογα την επιλογή, προκύπτει και η αντίστοιχη φωτογραφία. Συνήθως η οπίσθια κάμερα δημιουργεί μεγαλύτερες εικόνες και έτσι και ο χρόνος επεξεργασίας είναι μεγαλύτερος.



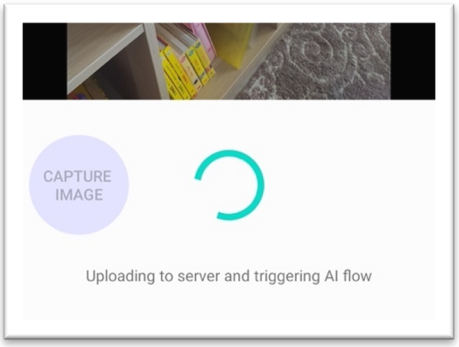

**Εικόνα 7. Περιβάλλον λήψης εικόνας**




**Εικόνα 8. Απεικόνιση λήψης**

Αφού ο χρήστης κάνει την λήψη της φωτογραφίας που επιθυμεί να μετασχηματίσει, η εφαρμογή επανέρχεται στο περιβάλλον χειρισμού. Απεικονίζεται η φωτογραφία στον κατάλληλο χώρο και ενεργοποιείται η επιλογή “APPLY AI CARTOON”. Σε αυτή τη φάση μπορεί να επιλέξει εκ νέου φωτογραφία και να αγνοήσει την τρέχουσα λήψη, επιλέγοντας εκ νέου “CAPTURE IMAGE” ή να προχωρήσει στο μετασχηματισμό.

Όταν ο χρήστης επιλέξει να προχωρήσει στο μετασχηματισμό, αρχίζει η διεργασία και εμφανίζεται ένα spinner bar για να ειδοποιήσει το χρήστη ότι η εφαρμογή είναι σε κατάσταση αναμονής καθώς υλοποιείται η διαδικασία στα διάφορα στάδιά της. Όσο ο χρήστης βρίσκεται σε αναμονή μια σειρά διεργασιών συμβαίνουν.

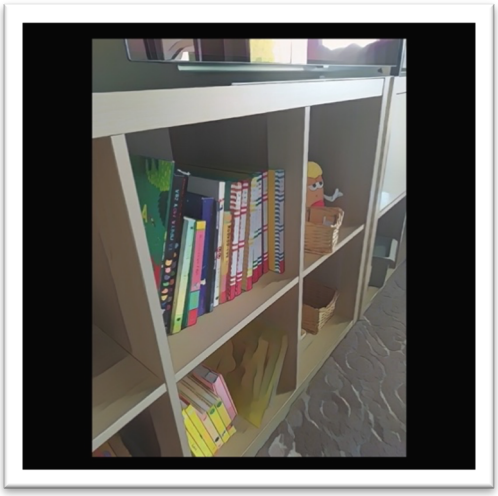



**Εικόνα 9. Ένδειξη αναμονής**

Αρχικά η εικόνα στέλνεται στο server χρησιμοποιώντας το κατάλληλο POST request. Ο server παραλαμβάνει και επιστρέφει ένα id, ώστε να γνωρίζει η εφαρμογή android πως μπορεί να ζητήσει πληροφορίες σχετικά με την κατάσταση στην οποία βρίσκεται η επεξεργασία. Τώρα η εφαρμογή εισέρχεται σε κατάσταση αναμονής όσο συμβαίνει ο μετασχηματισμός στο server.

Ο server αφού παραλάβει την εικόνα, την στέλνει στο υποσύστημα AI για να γίνει ο μετασχηματισμός. Στο μεσοδιάστημα, η εφαρμογή κάθε 5 δευτερόλεπτα ρωτάει το server με χρήση του id αν έχει ολοκληρωθεί η διαδικασία.

Όταν πάρει θετική απάντηση, κατεβάζει το νέο αρχείο στη συσκευή. Στη συνέχεια το απεικονίζει αλλά και το αποθηκεύει στο image gallery της φορητής μας συσκευής.



**Εικόνα 10. Απεικόνιση τελικής εικόνας**

Ακολουθούν ενδεικτικά ζεύγη αρχικής – μετασχηματισμένης εικόνας, όπως προέκυψαν από τη χρήση της εφαρμογής. Χρησιμοποιήθηκαν πολλές εικόνες και επελέχθη η παρουσίαση κάποιων χαρακτηριστικών εξ αυτών.

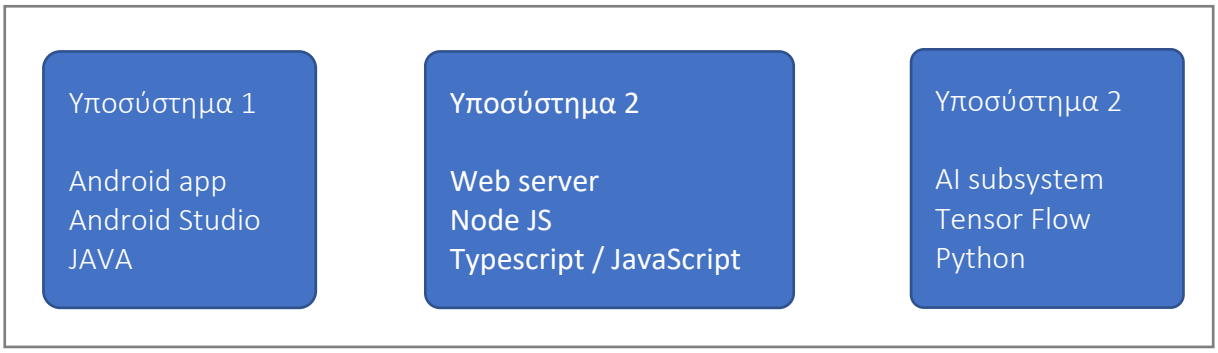
**Εικόνα 11. Μετασχηματισμός σύνθεσης φρούτων.**


**Εικόνα 12. Μετασχηματισμός εικόνας σαλονιού.**

**Εικόνα 13. Μετασχηματισμός σύνθεσης με λούτρινα ομοιώματα ζώων.**

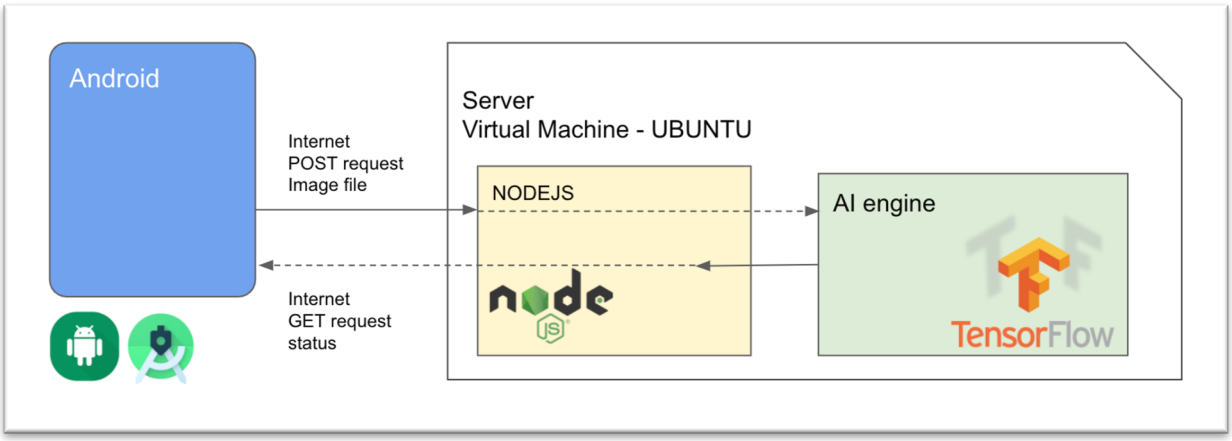
# 4. Αρχιτεκτονική

Η εργασία αυτή κινείται στον χώρο της τεχνητής νοημοσύνης, υλοποιώντας ένα σύστημα μετασχηματισμού φωτογραφιών του χρήστη, σε εκδόσεις cartoon. Η υλοποίηση αποτελείται από 3 διακριτά υποσυστήματα τα οποία χρησιμοποιών το καθένα διαφορετικές πλατφόρμες και τεχνολογίες.




**Εικόνα 14. Υποσυστήματα που υλοποιούνται και δομούν την εφαρμογή**

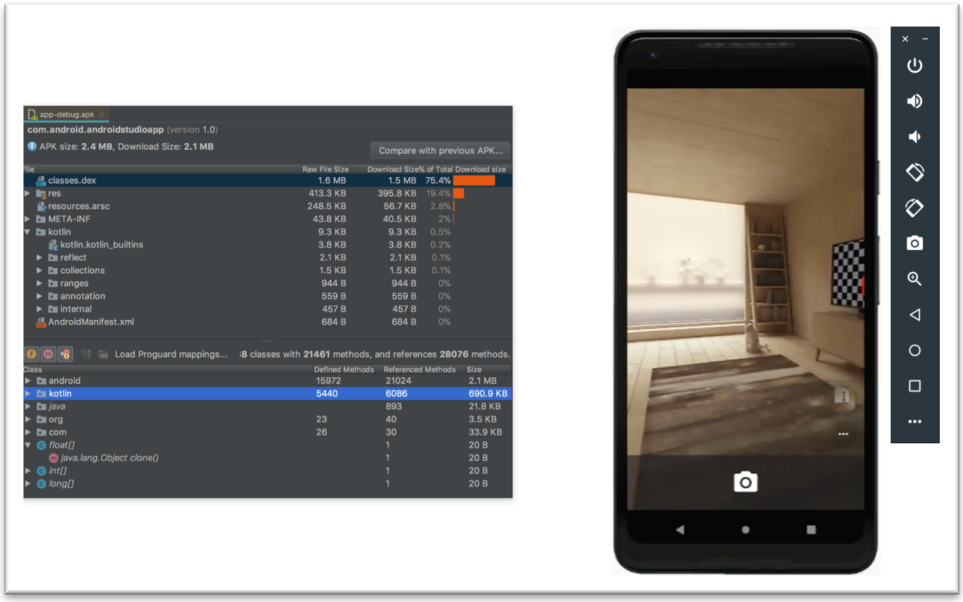
Το υποσύστημα 1, είναι δομημένο στο περιβάλλον Android Studio και χρησιμοποιεί γλώσσα JAVA. Πρόκειται για την Client Side πλευρά της εφαρμογής και υλοποιεί την διεπαφή με τον χρήστη. Το Υποσύστημα 2 αναλαμβάνει να χειριστεί τις απαιτήσεις διακίνησης δεδομένων μέσα από το διαδίκτυο. Χρησιμοποιεί τη πλατφόρμα NodeJS ενώ ο κώδικάς είναι γραμμένος σε γλώσσα Typescript. Το υποσύστημα 3 αναλαμβάνει το μετασχηματισμό της εικόνας και υλοποιείται σε πλατφόρμα Tensor Flow με κώδικα γραμμένο σε Python.




**Εικόνα 15. Τεχνολογίες ανά υποσύστημα**

## 4.1. Υποσύστημα 1 – User Interface – Android App

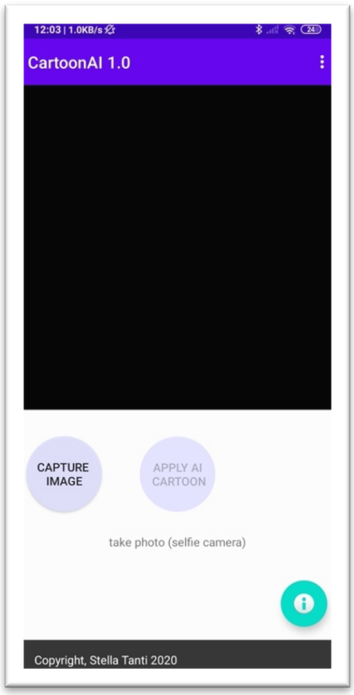
Το υποσύστημα 1, αποτελεί την διεπαφή λειτουργίας που απευθύνεται στο χρήστη. Πρόκειται για εφαρμογή Android, οπότε έχει υλοποιηθεί σε περιβάλλον Android Studio 4.0.1 (16). Το Android Studio είναι ένα σύνολο εργαλείων που συστηματοποιεί και διευκολύνει την ανάπτυξη εφαρμογών για συσκευές που λειτουργούν υπό το λειτουργικό σύστημα Android. Βασικό συστατικό του είναι ο επεξεργαστής κώδικα όπου γίνεται η ανάπτυξη του κώδικα με παράλληλη χρήση εργαλείων εκσφαλμάτωσης. Ενσωματώνει εξομοιωτή συσκευών καθώς το υπό ανάπτυξη λογισμικό πρέπει να δοκιμάζεται σε διαφορετικές εκδόσεις του λογισμικού Android και να προσαρμόζεται σε διαφορετικό μέγεθος οθόνης. Ο εξομοιωτής δίνει τη δυνατότητα ορισμού συσκευών με διαφορετικά χαρακτηριστικά.




**Εικόνα 16. Εργαλεία προσομοίωσης.**

Το παραπάνω λογισμικό διαθέτει δύο εναλλακτικές γλώσσες προγραμματισμού. Η πρώτη είναι η ευρέως διαδεδομένη γλώσσα **JAVA** (17) που συναντάται σε όλο το εύρος των εφαρμογών που μπορεί να συναντήσει κανείς. Είναι αντικειμενοστραφής γλώσσα και ανακοινώθηκε επίσημα το 1995 από την Sun. Σχεδιάστηκε να platform independent να εκτελείται δηλαδή σε κάθε είδους λειτουργικό σύστημα κάνοντας χρήση ενός ενδιάμεσου επιπέδου μετάφρασης προς το λειτουργικό σύστημα, το JVM (Java Virtual Machine).

Η άλλη εναλλακτική επιλογή είναι η πιο νέα γλώσσα προγραμματισμού **Kotlin** (18) που εισήχθη από τη JetBrains. Είναι πάλι γλώσσα που ακολουθεί αντικειμενοστραφή προσέγγιση και στηρίζεται στο JVM (Java Virtual Machine). ς Java, Scala, C#, Groovy, είναι συμβατή με την Java ™ και Android ™, έχει σχεδιαστεί για να τρέχει σαν εγγενής κώδικας στα iOS και macOS.




Στη παρούσα εργασία, έγινε χρήση της γλώσσας προγραμματισμού JAVA έναντι της Kotlin καθώς κρίθηκε πως είναι σαφώς πιο ώριμη και δοκιμασμένη, ενώ παράλληλα υπάρχει μια πολύ μεγάλη κοινότητα που προσφέρει τεκμηρίωση και πληροφορίες στα πλαίσια της γλώσσας. Η Kotlin λόγω της πιο πρόσφατης ύπαρξής της καθίσταται πιο δύσκολη στην εύρεση υποστήριξης.

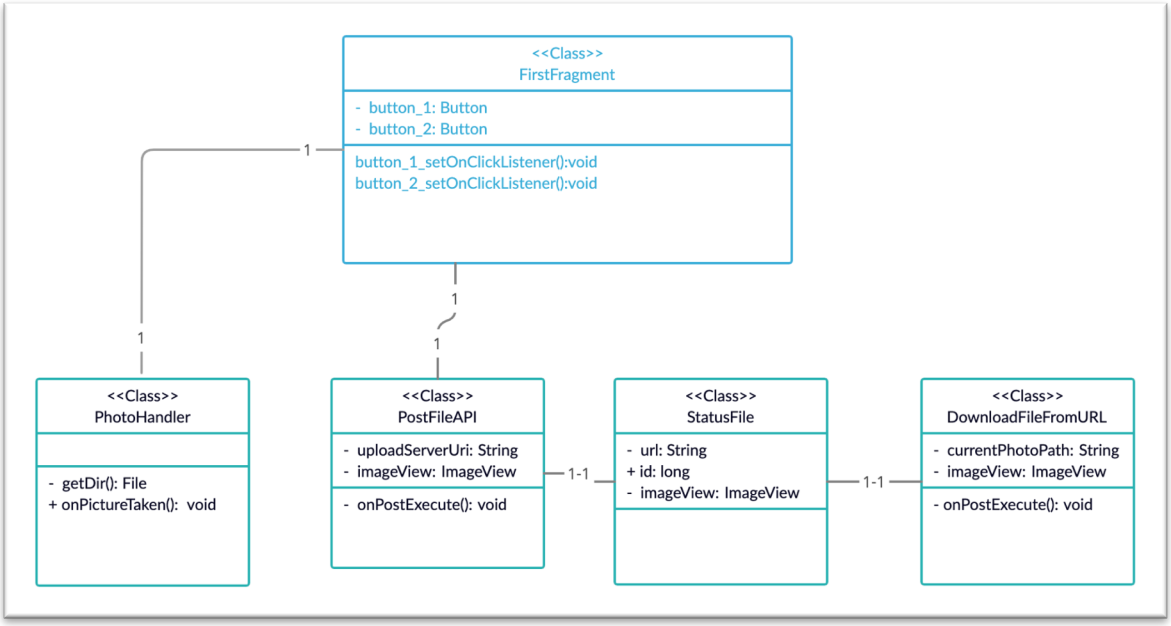
Η εφαρμογή αναπτύχθηκε με γνώμονα την απλότητα και ευχρηστία. Το περιβάλλον που καλείται να χειριστεί ο χρήστης είναι δομημένο με τρόπο που να είναι αυτονόητη η ροή ενεργειών που πρέπει να εκτελέσει χρήστης ώστε να φτάσει στο τελικό παραγόμενο προϊόν.

**Εικόνα 17. App Layout**

Στην εικόνα 17 απεικονίζεται το περιβάλλον όπως διαμορφώνεται κατά την πρώτη εκκίνηση. Στο επάνω μέρος υπάρχει ο τίτλος και η έκδοση στης εφαρμογής (CartoonAI 1.0). Ακολουθεί χώρος που έχει προβλεφθεί για την απεικόνιση της φωτογραφίας. Κατά την εκκίνηση εμφανίζεται με μαύρο υπόβαθρο για να υποδείξει πως είναι σε κατάσταση αναμονής. Στην ενότητα 3, «Παρουσίαση και χρήση», γίνεται εκτενής αναφορά στην λειτουργία την εφαρμογής και το γραφικό της περιβάλλον.

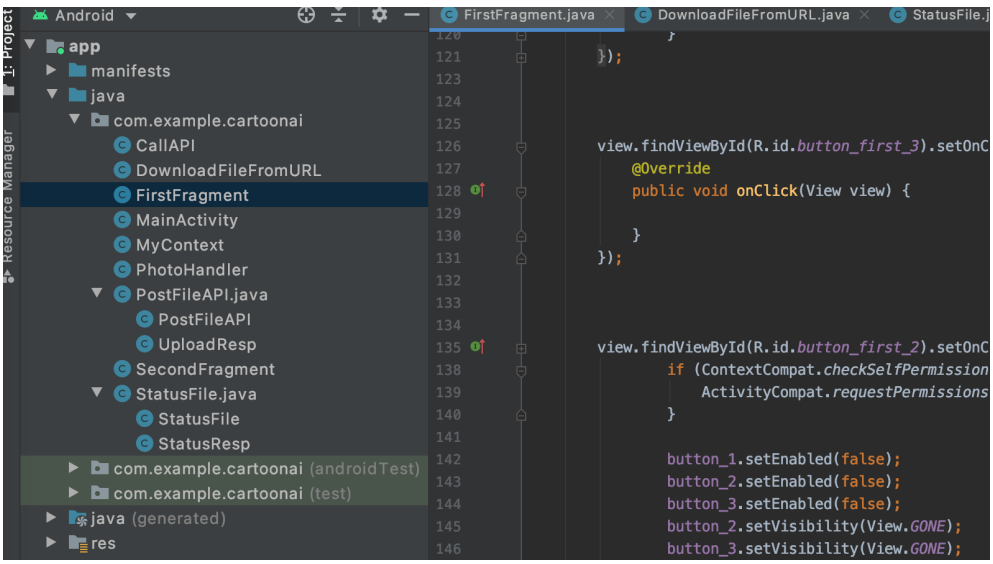
Αναπτύσσοντας την εφαρμογή, προέκυψαν 5 βασικές κλάσεις. Η Πρώτη κλάση είναι το view της εφαρμογής (**FirstFragment**) και έχει δύο αντικείμενα που το συνοδεύουν. Αυτά δεν είναι άλλα από τα δύο κουμπιά (button_1, button_2) που εξυπηρετούν τις λειτουργίες λήψης την φωτογραφίας και μεταφόρτωσης στο server, μαζί με τις αντίστοιχες συναρτήσεις που κάνουν trigger τις διεργασίες.

Στο διάγραμμα κλάσεων που ακολουθεί (εικόνα 18), απεικονίζονται οι βασικές κλάσεις όπως αυτές αναπτύχθηκα στο περιβάλλον του Android Studio σε JAVA. Επίσης ακολουθεί εικόνα δομής JAVA όπως οργανώνεται στο περιβάλλον ανάπτυξης (εικόνα 16).




**Εικόνα 18. Διάγραμμα κλάσεων βασικών οντοτήτων εφαρμογής.**

Η Δεύτερη κλάση -**PhotoHandler**- εξυπηρετεί τη λήψη φωτογραφίας και χειρίζεται με κατάλληλο τρόπο το hardware για να γίνονται οι απαραίτητες λειτουργίες (εναλλαγή εμπρόσθιας – οπίσθιας, προσωρινή αποθήκευση, ενημέρωση συστήματος για ολοκλήρωση, render στο κατάλληλο control).




**Εικόνα 19. Απεικόνιση δομής JAVA όπως οργανώνεται στο περιβάλλον ανάπτυξης.**

Αφού το σύστημα ενημερωθεί πως η φωτογραφία είναι διαθέσιμη, και ύστερα από επιλογή του χρήστη, η κλάση **PostFileAPI** ξεκινάει τη διαδικασία uploading. Κατά την ολοκλήρωση αποδίδεται από το backend ένα id για να είναι δυνατή η παρακολούθηση της κατάστασης της διεργασίας. Αυτό επιτυγχάνεται με την κλάση **StatusFile** η οποία να πέντε δευτερόλεπτα «ρωτάει» τον server για την κατάσταση της διεργασίας (pending/ complete).

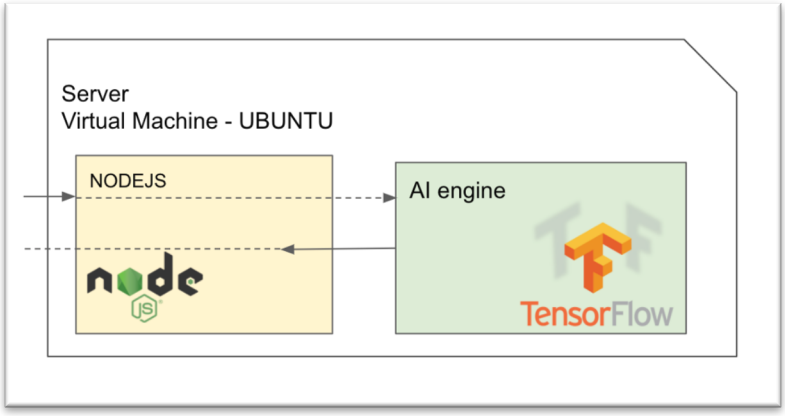
Όταν πάρει απάντηση κατάστασης complete, ο έλεγχος περνάει στην κλάση **DownloadFileFromURL** η οποία πέρα από το κατέβασμα, επιφορτίζεται να ενημερώσει το User Interface, να αποθηκεύσει την μετασχηματισμένη φωτογραφία και να την κάνει διαθέσιμη μέσα από το Photo Gallery του λειτουργικού συστήματος.

Οι διαδικασίες upload, download, status κλπ, αποτελούν χειριστές κλήσεων http για την αλληλεπίδραση μέσα από το internet. Οπότε ενσωματώνουν χειριστές δικτυακών πρωτοκόλλων κατά τα πρότυπα επικοινωνίας μέσα από το διαδίκτυο (POST / GET requests). Επίσης, λόγω των αναμονών της ολοκλήρωσης διεργασιών, έντονη και απαραίτητη είναι η χρήση ασύγχρονων τεχνικών. Αυτό σημαίνει πως ενώ το σύστημα στέλνει μια εντολή ή καλεί μια συνάρτηση, αντί να περιμένει την απάντηση ή το αποτέλεσμα διακόπτοντας την ροή εκτέλεσης, συνεχίζει την εκτέλεση του προγράμματος και ορίζει μία callback συνάρτηση η οποία θα εκτελεστεί όταν ολοκληρωθεί η ασύγχρονη διεργασία.

## 4.2. Server Side υποσυστήματα

Όπως έχει συζητηθεί στο κεφάλαιο 4, η παρούσα εφαρμογή για την υποστήριξη των απαιτητικών διεργασιών μηχανικής μάθησης, υποστηρίζεται από εξυπηρετητή με πρόσβαση στο διαδίκτυο ο οποίος αναλαμβάνει τις πρόσθετες διεργασίες. Για τις ανάγκες της εργασίας αναπτύχθηκε ένα container Linux Ubuntu 18, σε virtualization περιβάλλον Proxmox (19).

Η επιλογή χρήσης ενός container (μία διαφορετικής προσέγγισης Virtual Machine) έγινε με σκοπό το περιβάλλον να είναι απομονωμένο από άλλα λογισμικά και να λειτουργεί ως ένα κλειστό σύστημα (sandbox) το οποίο μπορεί εύκολα να μεταφερθεί ολόκληρο, να αποθηκευτεί και να ανακτηθεί. Το λειτουργικό σύστημα Linux επιλέχθηκε επειδή δεν εισάγει προβλήματα αδειοδότησης – κόστους, ενώ παράλληλα είναι πλήρως δοκιμασμένο ως server side software host με ισχυρή παρουσία υποστήριξης από την παγκόσμια κοινότητα. Τέλος, η μόνη πρόσβαση που χρειάστηκε να ανοίξει , είναι αυτή της πόρτας 80 που αποτελεί την πόρτα http (και 443 για https).




**Εικόνα 20. Γραφική απεικόνιση υποσυστημάτων που φιλοξενεί το Linux VM**

Το Ubuntu Virtual Machine φιλοξενεί τα δύο υποσυστήματα υποστήριξης τα οποία περιγράφονται στη συνέχεια.

### 4.2.1. Υποσύστημα 2 – Web server

Το υποσύστημα 2 αποτελεί το Web Server της εφαρμογής κατά τις ανάγκες που έχουν περιγραφεί παραπάνω. Πρόκειται για εκείνο λογισμικό που είναι υπεύθυνο για τον χειρισμό των κλήσεων που γίνονται με χρήση του διαδικτύου, την μεταφόρτωση και αποστολή αρχείων, την επικοινωνία με το υποσύστημα 3 αλλά και την διαχείριση τοπικά των αρχείων.

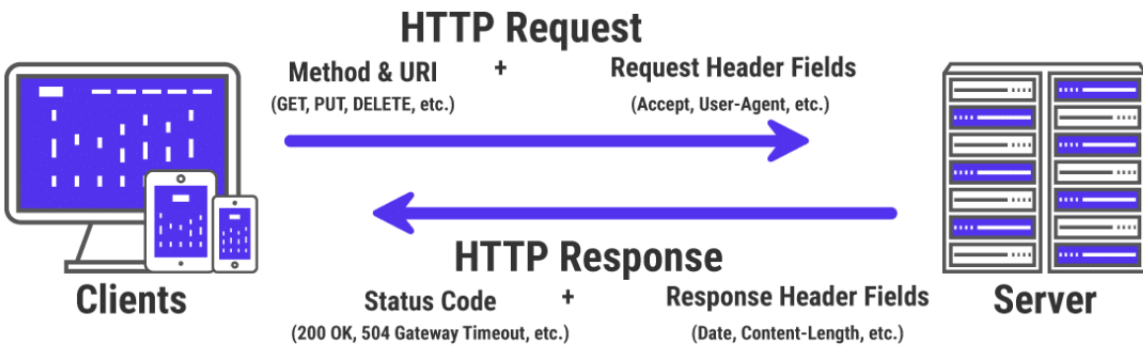
Το λογισμικό πάνω στο οποίο γίνεται η ανάπτυξη λέγεται **NodeJS** (20). Είναι μι εξέλιξη της μηχανής JavaScript (21) του google chrome (V8) το οποίο αναπτύσσει και συντηρεί η Google. Το βασικό της πλεονέκτημα είναι πως η γλώσσα ανάπτυξης που χρησιμοποιεί είναι η JavaScript και έτσι αφενός ο κώδικας μπορεί να έχει κοινά στοιχεία με το κώδικα που γράφουμε για σελίδες στο internet και αφετέρου το μοντέλο και η μορφή δεδομένων που χρησιμοποιεί είναι κοινή (json) και έτσι δεν υπάρχει ανάγκη για διαχείριση και μετατροπή από μορφή σε μορφή που πολλές φορές μπορεί να είναι επίπονη αλλά και πηγή σφαλμάτων.

Ως γλώσσα ανάπτυξης χρησιμοποιήθηκε η **TypeScript** (22). Η γλώσσα αυτή είναι μία παραλλαγή της JavaScript, η οποία δημιουργήθηκε για να καλύψει κενά αυτής με βασικότερο την έλλειψη τύπων. Έτσι η TypeScript μπορεί να θεωρηθεί ως μία βελτιωμένη JavaScript που με χρήση του κατάλληλου compiler μεταφράζεται σε JavaScript για να τρέξει σε NodeJS.

#### 4.2.1.1 Hypertext Transfer Protocol - HTTP

Για την επικοινωνία με χρήση του διαδικτύου αξιοποιείται το πρωτόκολλο Hypertext Transfer Protocol (HTTP) (23). Πρόκειται για το κύριο Πρωτόκολλο που χρησιμοποιείται από τους browsers για την επικοινωνία με τους αντίστοιχους servers – εξυπηρετητές, με πρώτες περιπτώσεις χρήσεις από την τεκμηριωμένη έκδοση 1.1 από το 1997.

Η βασική ιδέα στηρίζεται στην δομή Ερώτημα – Απάντηση (Request – Response) με την προϋπόθεση ύπαρξης φυσικής πρόσβασης μέσα από δίκτυο (εικόνα 17). Έτσι όταν υπάρχει ανάγκη για πρόσβαση σε ένα πόρο (κείμενο, εικόνα, ήχος, κλπ.). Η διαδικασία είναι




**Εικόνα 21. Σχέδιο στοιχείων επικοινωνίας HTTP**

- Ένας web browser (ή οποιοσδήποτε άλλος χρήστης του πρωτοκόλλου όπως μία εφαρμογή κινητού) υποβάλλει ένα HTTP request στον server.
- server δεχόμενος το ερώτημα, εκτελεί μια σειρά ενεργειών ανάλογα με τις οδηγίες που έχει από το πρόγραμμά του.
- Ο server επιστρέφει τα δεδομένα μέσα από το δίκτυο.

#### 4.2.1.2 HTTP Request

Ένα HTTP ερώτημα (http request) είναι ένα μπλοκ πληροφορίας που συχνά ονομάζουμε μήνυμα (Request message). Η βασική του δομή είναι :

- Γραμμή ερωτήματος

  ***GET*** */images/logo.png HTTP/1.1*

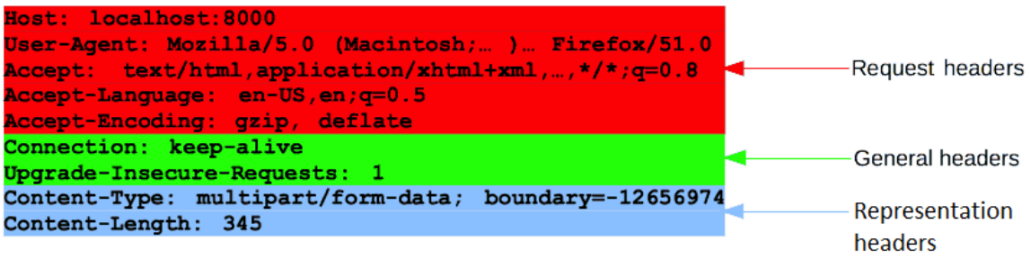
  Στο παράδειγμα ζητείται να χρησιμοποιηθεί η μέθοδος ερωτήματος GET για να επιστραφεί η αντίστοιχη εικόνα. Εκτός από τη μέθοδο GET υπάρχουν και άλλοι που θα αναφερθούν στη συνέχεια.

- Headers ερωτήματος (24)

  Στοιχεία που ορίζουν παραμέτρους του ερωτήματος όπως

  *Accept-Language*

  *Content-Type*




**Εικόνα 22. Παράδειγμα Request Headers HTTP**

- Μία κενή γραμμή
- Προαιρετικό σώμα μηνύματος (δεδομένα που θέλουμε να στείλουμε στον server)

Για να υπάρχει μεγάλη ευελιξία, όπως είδαμε παραπάνω ορίζεται σε κάθε ερώτημα και η μέθοδος. Πρόκειται για προσδιορισμό της ενέργειας που ζητείται να γίνει στον επιθυμητό πόρο. Ο πόρος αυτός μπορεί να είναι ένα αρχείο που ήδη υπάρχει στο server η που ζητείται να δημιουργηθεί.

Οι μέθοδοι που έχουν οριστεί είναι

- **GET**
  Ζητάει το ερώτημα μια αναπαράσταση του πόρου (εικόνα, ήχος, κείμενο, κλπ.) . Ένα τέτοιο ερώτημα πρέπει μόνο να ανακτήσει δεδομένα και να μην έχει καμία άλλη επίδραση στον πόρο.

- **HEAD**
  Ζητάει το ερώτημα μια αναπαράσταση του πόρου (εικόνα, ήχος, κείμενο, κλπ.) όπως και το GET αλλά χωρίς τα δεδομένα. Χρησιμοποιείται συνήθως για ανάκτηση μεταδεδομένων.

- **POST**
  Ζητάει το ερώτημα να καταχωρηθεί η απεσταλμένη οντότητα (δεδομένα – αρχείο) στον server σε υπάρχουσα εγγραφή. Μπορεί να είναι για παράδειγμα ένα postσε ένα blog.

- **PUT**
  Όπως και το POST αλλά με τη διαφορά πως ζητείται η δημιουργία νέου πόρου.

- **DELETE**
  Χρησιμοποιείται για να διαγραφεί ένας πόρος.

- **TRACE**
  Η μέθοδος αυτή ζητάει την επαναποστολή από το server του request ώστε να εντοπιστούν τυχόν αλλαγές που έχουν γίνει από ενδιάμεσους server.

- **OPTIONS**
  Η μέθοδος αυτή τις μεθόδους που υποστηρίζει ο server για τον συγκεκριμένο πόρο.

- **PATCH**
  Η μέθοδος αυτή ζητάει μερική τροποποίηση του πόρου.

#### 4.2.1.3 HTTP Response

Ένα HTTP ερώτημα (http request) όταν αποσταλεί στον εξυπηρετητή στον οποίο απευθύνεται, αναμένει την αντίστοιχη απάντηση (HTTP Response). Ακριβώς όπως και το HTTP Request, έτσι και η αντίστοιχη απάντηση είναι δομημένη σύμφωνα με το πρότυπο που ορίζει το πρωτόκολλο HTTP. Ποιο συγκεκριμένα, η απάντηση αποτελείται από τα εξής δομικά στοιχεία:

- Γραμμή κατάστασης (status line)

  ***HTTP/1.1 200 OK***

  Στο παράδειγμα αναφέρεται επιτυχής ολοκλήρωση διαδικασίας. Το βασικό στοιχείο είναι ο κώδικας κατάστασης (status code). Υπάρχουν γενικές οδηγίες για τους κωδικούς αλλά μπορούν να χρησιμοποιηθούν και κωδικοί εκτός αυτών. Σε γενικές γραμμές ορίζονται με βάση το πρώτο ψηφίο.

  - 1xx Informational
  - 2xx Successful (πχ 200)
  - 3xx Redirection
  - 4xx Client error (πχ 400 Bad Request)
  - 5xx Server error (πχ 500 Internal server error)

- Headers απάντησης (Response headers)

- Μία κενή γραμμή

- Προαιρετικό σώμα μηνύματος επιστροφής (δεδομένα που επιστρέφει ο server)

Στις παρακάτω εικόνες φαίνεται ένα απλό παράδειγμα HTTP επικοινωνίας με το HTTP GET Request (εικόνα 23) το οποίο ζητάει το κείμενο μιας Ιστοσελίδας και στη συνέχεια την αντίστοιχη απάντηση (HTTP Response) (εικόνα 24) να επιστρέφει το περιεχόμενο της ιστοσελίδας “www.example.com” .

```
GET / HTTP/1.1
Host: www.example.com
```

**Εικόνα 23. Παράδειγμα GET HTTP Request**

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 155
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World, this is a very simple HTML document.</p>
  </body>
</html>
```

**Εικόνα 24. Response του παραπάνω παραδείγματος**

### 4.2.2 Web server Code

Το υποσύστημα 2, το οποίο υλοποιεί το Web server κομμάτι της εφαρμογής, επιτρέπει τρεις κλήσεις. Μία για το upload της φωτογραφίας, μία για την ερώτηση status και μία για την αποστολή της μετασχηματισμένης εικόνας. Το κεντρικό αρχείο κώδικα υλοποιείται από 29 μόλις γραμμές όπως φαίνεται στη σχετικό screenshot (εικόνα 21).

Τα βασικά λειτουργικά block κώδικα περιλαμβάνουν:

- Εισαγωγή βιβλιοθήκης διαχείρισης web requests (**express**), 1-6

  Η βιβλιοθήκη αυτή επιτρέπει στην εφαρμογή μας να λειτουργεί ως web server, να μπορεί δηλαδή να δέχεται κλήσεις μέσα από το διαδίκτυο και να ρυθμίζει όλες τις παραμέτρους που εξασφαλίζουν τη συμβατότητα με τα πρότυπα http / https.

- Αρχικοποίηση και ορισμός μέγιστου μεγέθους αρχείου (**50MB**), 8-11

  Για λόγους ασφάλειας και εξασφάλισης διαθεσιμότητας πόρων, η κλήση που δέχεται τη φωτογραφία εισόδου θέτει περιορισμό στο επιτρεπόμενο μέγεθος αυτής. Στην εφαρμογή έχουμε βάλει όριο τα 50 MB, τιμή που υπερκαλύπτει το μέγεθος μιας τυπικής φωτογραφίας που έχει δημιουργηθεί από φορητή συσκευή, και αφήνει αρκετό περιθώριο ώστε να είναι future proof το σύστημα σε περιπτώσεις συσκευών με πολύ μεγάλη διάσταση αισθητήρα (έχουμε δει μέχρι και 108 Megapixel μέχρι στιγμής σε κινητό τηλέφωνο)

- Ορισμός **headers** για τα response, 14-19

  Όπως αναφέρθηκε στην ενότητα του πρωτοκόλλου HTTP , τα headers προσδιορίζουν παραμέτρους λειτουργίας του συστήματος. Εδώ ζητάμε μεταξύ άλλων να γίνονται δεκτές οι κλήσεις από κάθε client (browser ή κινητό).

src > TS index.ts > ...

```ts
import * as express from 'express';
import * as FileUpload from 'express-fileupload';
import * as bodyParser from 'body-parser';
import { fileStatus, fileUpload } from './upload';

let expressServer = express();

expressServer.use(bodyParser.json({ limit: '50mb' }));
expressServer.use(FileUpload({
    limits: { fileSize: 500 * 1024 * 1024 }, useTempFiles: true
}));


expressServer.use((_req, res, next) => {
    console.log(_req.originalUrl);
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Headers', 'Authorization, Content-Type');
    next();
});

expressServer.post('/upload', fileUpload);
expressServer.get('/upload', fileUpload);
expressServer.get('/status/:id', fileStatus);

expressServer.use('/file/res', express.static(__dirname + '/../tmp'));

expressServer.listen(5000, function () {
    console.log(`Server is listening on 5000`);
});
```

**Εικόνα 25. Κεντρικό αρχείο κώδικα υποσυστήματος 2 / Typescript**

- Ορισμός POST/GET request για **upload,** 21-22
- Ορισμός **status** request, 23
- Άνοιγμα πρόσβασης για το **download** των αρχείων, 25
- **Εκκίνηση** web server στη θύρα 5000.

Σημειώνεται εδώ πως ενώ η εφαρμογή ακούει στην θύρα 5000, το gateway του host λειτουργικού συστήματος κάνει αναδρομολόγηση στην τυπική θύρα http/https. Η αναφορά στην τεχνική αυτή ξεφεύγει από το σκοπό του παρόντος.

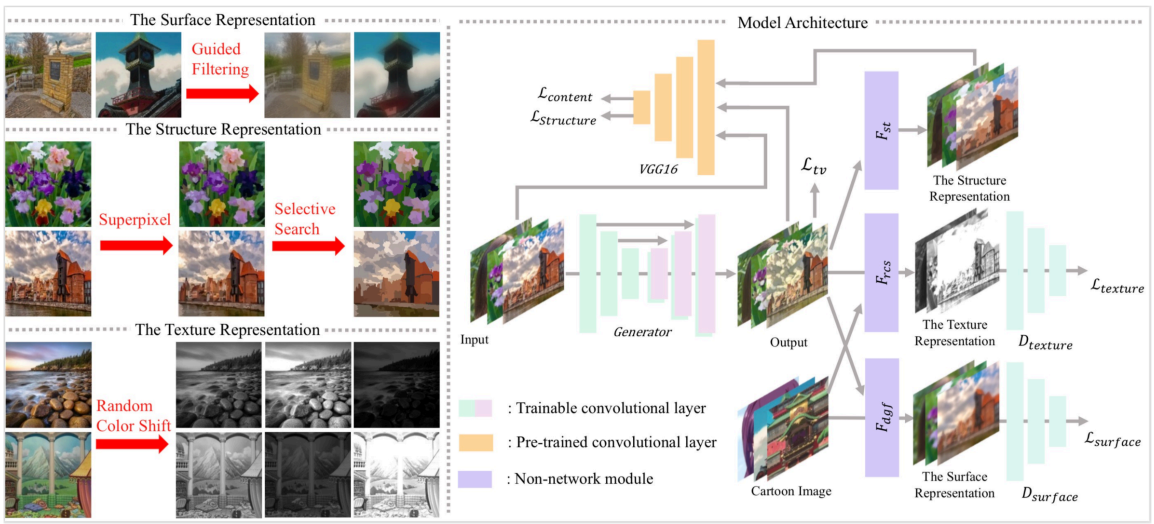
### 4.2.3 Υποσύστημα 3 – Artificial Intelligence

Το υποσύστημα 3 είναι υπεύθυνο για τον καθαυτό μετασχηματισμό της εικόνας από την αρχική λήψη στην cartoon έκδοση. Η πλατφόρμα στην οποία υλοποιείται το αντίστοιχο νευρωνικό δίκτυο είναι το Tensor Flow το οποίο έχει αναπτυχθεί από τη Google. Το Tensor Flow είναι μία end-to-end πλατφόρμα ανοιχτού κώδικα για μηχανική μάθηση. Αποτελείται από ένα σύνολο εργαλείων, βιβλιοθηκών και πόρων που επιτρέπει σε ερευνητές να αναπτύξουν εύκολα και να θέσουν σε λειτουργία εφαρμογές μηχανικής μάθησης.

Το Tensor Flow ένα μεγάλο πλήθος από κατηγορίες προβλημάτων όπως ταξινόμηση εικόνων, εμπλουτισμό δεδομένων, κατάτμηση εικόνων, ανίχνευση αντικειμένων, αναγνώριση κειμένου, εξαγωγή keywords από εικόνες, αναγνώριση ήχου και πολλά ακόμα.

Για τον προγραμματισμό του υποσυστήματος χρησιμοποιούμε τη γλώσσα προγραμματισμού Python, η οποία είναι πολύ διαδεδομένη στο χώρο της έρευνας με έντονο στοιχείο μαθηματικών μετασχηματισμών και απεικονίσεων.

Το μοντέλο που εφαρμόζεται αναπτυχθεί το 2020 (1) και έχει δημοσιευτεί με τίτλο «Learning to Cartoonize Using White-box Cartoon Representations». Σκοπός της εργασίας ήταν η ανάπτυξη ενός μοντέλου μηχανικής μάθησης για δημιουργία Cartoon αναπαραστάσεων. Χρησιμοποιείται η κατηγορία εκμάθησης / μοντελοποίησης General Adversarial Network (GAN). Συνολικά χρησιμοποιεί τρία επίπεδα εκμάθησης ένα για την αναπαράσταση της επιφάνειας, ένα για την αναπαράσταση των δομών και ένα για την αναπαράσταση των υφών. Στην εικόνα 22 φαίνονται τα τρία επίπεδα καθώς ο τρόπος αλληλεπίδρασης κάθε στοιχείου.




**Εικόνα 26. Επίπεδα και αλληλεπίδραση στοιχείων του μοντέλου**

Η διαδικασία δημιουργίας του μοντέλου, εκτός από τον ορισμό του δικτύου και των επιπέδων, περιλαμβάνει και την εκπαίδευση. Δίνεται δηλαδή στο σύστημα ένας μεγάλος όγκος δεδομένων εικόνων αρχικών και τελικών, ώστε το σύστημα να καταφέρει να προσδιορίσει τις σχέσεις που επιτρέπουν την δημιουργία των Cartoon αναπαραστάσεων από μια αρχική φυσική εικόνα. Για να επιτευχθεί επιτυχώς η διαδικασία της εκπαίδευσης, απαιτείται μεγάλος όγκος δεδομένων αλλά και προσεκτική επιλογή όσο αφορά τις κατηγορίες αλλά και την ποιότητα αυτών. Στη διαδικασία εκπαίδευσης του συγκεκριμένου

μοντέλου χρησιμοποιήθηκαν 10000 εικόνες από πρόσωπα και 10000 εικόνες από τοπία. Ενδεικτικά μερικά ζεύγη δεδομένων παρατίθεται.

**Εικόνα 27. Ζεύγος εκπαίδευσης / οδός**

**Εικόνα 28. Ζεύγος εκπαίδευσης / φαγητά**

**Εικόνα 29. Ζεύγος εκπαίδευσης / εστιατόριο**

**Εικόνα 30. Ζεύγη εκπαίδευσης / άνθρωποι**

**Εικόνα 31. Ζεύγη εκπαίδευσης / φιγούρες**

**Εικόνα 32. Ζεύγη εκπαίδευσης / πρόσωπα**

Στην εφαρμογή της εργασίας, η εκπαίδευση του συστήματος έγινε μία φορά για να παραχθεί το αρχείο που περιγράφει το Νευρωνικό Δίκτυο. Η διαδικασία αυτή διήρκησε περισσότερο από 30 λεπτά σε υπολογιστή με 40GB RAM, CPU i7, GPY NVIDIA 2060 RTX. Πρόκειται για ιδιαίτερα απαιτητική διαδικασία, που όμως γίνεται μια φορά για την παραγωγή του αρχείο που περιγράφει το μοντέλο.

Το υποσύστημα 3 έχοντας έτοιμο το αρχείο που περιγράφει το μοντέλο, για κάθε μία νέα φωτογραφία που λαμβάνει, εφαρμόζει το μετασχηματισμό. Η διεργασία αυτή, αντίθετα με την εκπαίδευση, διαρκεί μερικά δευτερόλεπτα κάθε φορά, ανάλογα με το μέγεθος της φωτογραφίας. Η βασική συνάρτηση μετασχηματισμού ακολουθεί στο επόμενο σχήμα.

```python
def cartoonize(load_folder, save_folder, model_path):
    input_photo = tf.placeholder(tf.float32, [1, None, None, 3])
    network_out = network.unet_generator(input_photo)
    final_out = guided_filter.guided_filter(input_photo, network_out, r=1, eps=5e-3)

    all_vars = tf.trainable_variables()
    gene_vars = [var for var in all_vars if 'generator' in var.name]
    saver = tf.train.Saver(var_list=gene_vars)

    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    sess = tf.Session(config=config)

    sess.run(tf.global_variables_initializer())
    saver.restore(sess, tf.train.latest_checkpoint(model_path))
    name_list = os.listdir(load_folder)
    for name in tqdm(name_list):
        try:
            print(save_folder)
            print(name)
            load_path = os.path.join(load_folder, name)
            save_path = os.path.join(save_folder, name)
            image = cv2.imread(load_path)
            image = resize_crop(image)
            batch_image = image.astype(np.float32)/127.5 - 1
            batch_image = np.expand_dims(batch_image, axis=0)
            output = sess.run(final_out, feed_dict={input_photo: batch_image})
            output = (np.squeeze(output)+1)*127.5
            output = np.clip(output, 0, 255).astype(np.uint8)
            cv2.imwrite(save_path, output)
        except:
            print('cartoonize {} failed'.format(load_path))

```

**Εικόνα 33. Κεντρικό αρχείο κώδικα υποσυστήματος 3/ Python**

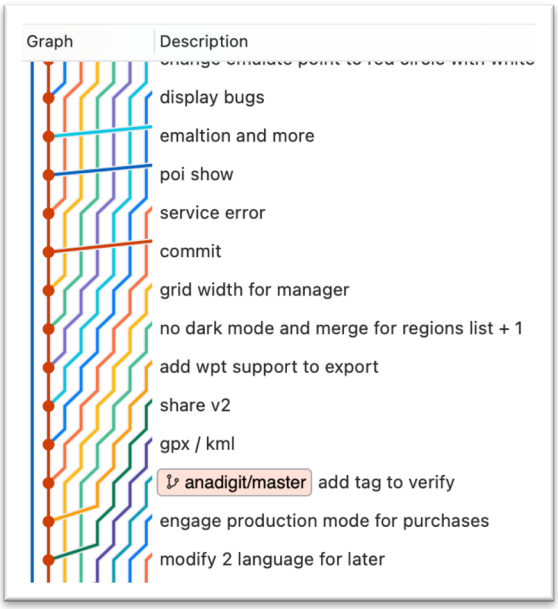
Παρατηρούμε πως δέχεται τρεις παραμέτρους. Τον φάκελο στον οποίο βρίσκεται η προς μετασχηματισμό φωτογραφία, τον φάκελο στον οποίο θα αποθηκεύσει το τελικό αρχείο και το path για το αρχείο που περιγράφει το μοντέλο. Αφού σαρώσει τον φάκελο για όλες τις φωτογραφίες , επαναλαμβάνει τη διαδικασία για κάθε μία από αυτές. Φορτώνει τους συντελεστές του μετασχηματισμού και υλοποιεί τη διαδικασία.

## 4.3 Versioning – Τεκμηρίωση

Η τρέχουσα εργασία αποτελείται από τρία υποσυστήματα, όπως έχει αναπτυχθεί στο παρόν έγγραφο τεκμηρίωσης, τα οποία με τη σειρά τους αποτελούνται από πολλά αρχεία διαφορετικών επεκτάσεων ανάλογα με την γλώσσα που κάθε φορά χρησιμοποιείται αλλά και την απαραίτητη δομή όπως αυτή καθορίζεται από την κάθε πλατφόρμα.

Γενικά η διαδικασία δημιουργίας εφαρμογών πληροφορικής χρειάζεται να μπορεί να ικανοποιήσει μια σειρά χαρακτηριστικών που κάνουν λειτουργική και αποδοτική την ανάπτυξη. Μερικές βασικές απαιτήσεις είναι :

- Δυνατότητα ανάπτυξης από πολλούς developers ταυτόχρονα σε διαφορετικά τμήματα αυτής.
- Δυνατότητα τμηματικής αποθήκευσης αλλαγών κώδικα.
- Δυνατότητα τεκμηρίωσης των παραπάνω αλλαγών.
- Δυνατότητα παράλληλης ανάπτυξης διαφορετικών δυνατοτήτων.
- Δυνατότητα επαναφοράς προηγούμενων εκδόσεων.
- Δυνατότητα κεντρικής αποθήκευσης και πρόσβασης.




Η ικανοποίηση των παραπάνω καθιστά τη διαδικασία ανάπτυξη λιγότερο επισφαλή, επιτρέπει τη συνεργασία και προσφέρει μια ευελιξία καθώς επιτρέπει την μη γραμμική ανάπτυξη αφού μπορούν οι μηχανικοί να δημιουργήσουν διακλαδώσεις για διαφορετικές εκδόσεις που δοκιμάζονται αυτόνομα και ενσωματώνονται στην κεντρική έκδοση προς τελική διάθεση. Η εικόνα 30 απεικονίζει ένα παράδειγμα ανάπτυξης όπου αναπτύσσονται παράλληλα διαφορετικές δυνατότητες μιας εφαρμογής, με σχόλια τεκμηρίωσης στα σημεία αποθήκευσης.

**Εικόνα 34. Απεικόνιση διακλαδώσεων παράλληλης ανάπτυξης**

Για την ικανοποίηση όλων των παραπάνω περιγραφόμενων συνθηκών, η εφαρμογή αναφοράς που χρησιμοποιείται από την κοινότητα των μηχανικών πληροφορικής είναι το GIT (25) που αναπτύχθηκε από τον Linus Torvalds, τον κύριο δημιουργό του συστήματος Linux. Είναι μια εφαρμογή με περιβάλλον γραμμής εντολών που επιτρέπει τη διαχείριση έργων με κύριες λειτουργίες (26):

- git clone Ανάκτηση έργου
- git commit Αποθήκευση αλλαγών
- git push Προώθηση αλλαγών στο κεντρικό αποθετήριο
- git fetch Λήψη αλλαγών από κεντρικό αποθετήριο

- git merge Ενσωμάτωση αλλαγών στο τοπικό αντίγραφο κώδικα
- git pull git fetch + git merge
- git branch Δημιουργία νέας διακλάδωσης για νέα δυνατότητα
- git checkout Αλλαγή τρέχοντος branch

Για κεντρικό repository (αποθετήριο) μπορεί να χρησιμοποιηθεί το ανοιχτό λογισμικό GitLab (27) το οποίο πρέπει να εγκατασταθεί σε υποδομή του developer ή έτοιμες λύσεις με πιο γνωστή το GitHub (28). Για τη παρούσα εργασία χρησιμοποιήθηκε η τελευταία λύση ως αποθετήριο για τα υποσυστήματα. Τα αντίστοιχα στοιχεία μπορούν να βρεθούν στις παρακάτω θέσεις :

- https://github.com/stellatanti/ai_android
- https://github.com/stellatanti/ai_be

Τέλος, αξίζει να αναφερθεί πως επειδή η διαχείριση των project με χρήση της γραμμής εντολών και των βασικών εντολών git είναι μη φιλική προς τον χρήστη, έχουν αναπτυχθεί προϊόντα τα οποία προσφέρουν γραφικό περιβάλλον για την εκτέλεση των αντίστοιχων ενεργειών. Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το SourceTree (29).

# 5 Συμπεράσματα και μελλοντικές επεκτάσεις

Η εργασία αυτή είχε ως αντικείμενο τη δημιουργία ενός end-to-end συστήματος μετασχηματισμού εικόνων του χρήστη από φορητή συσκευή Android σε αναπαράσταση Cartoon με χρήση τεχνητής νοημοσύνης. Ο κλάδος αυτό της πληροφορικής αποτελεί την αιχμή του δόρατος και όλο και περισσότερη πρόοδος παρουσιάζεται στον σύνθετο αυτό πεδίο. Η δυσκολία και σημαντικότητα της εργασίας αυτής συνοψίζεται σε δύο βασικές αλήθειες.

Πρώτον, η ανάπτυξη ενός τέτοιου συστήματος απαιτεί την ενασχόληση και ανάπτυξη κώδικα σε ένα πλήθος πλατφόρμων. Android, Linux, NodeJS, Tensor Flow ορίζουν το πλέγμα του οικοσυστήματος που πραγματεύτηκε το παρόν. Επίσης απαιτήθηκε η εργασία σε τρεις γλώσσες προγραμματισμού. JAVA για το Android, Typescript για NodeJS και Python για Tensor Flow. Παράλληλα ζητήματα επικοινωνίας μέσα από το διαδίκτυο αλλά και διαχείρισης αρχείων σε πολλαπλά λειτουργικά συστήματα έπρεπε να αντιμετωπιστούν.

Το δεύτερο στοιχείο που χαρακτηρίζει την εργασία αυτή, ταυτίζεται με τις προοπτικές μελλοντικής επέκτασης που προσφέρει. Επειδή ο σχεδιασμός είναι τέτοιος που κάθε υποσύστημα είναι ανεξάρτητο, είναι δυνατή η τροποποίηση του συστήματος με επέμβαση μόνο σε ένα υποσύστημα. Αν για παράδειγμα αλλάξουμε το μοντέλο μετασχηματισμού που εφαρμόζουμε, η εφαρμογή android αλλά και ο web server θα συνεχίσουν να λειτουργούν απρόσκοπτα.

Επιπρόσθετα, με μια μικρή παρέμβαση, θα μπορούσαμε να επιτρέπουμε την ύπαρξή πολλαπλών μοντέλων στο server. και με ένα επιπλέον drobox στο User Interface ο χρήστης να επιλέγει ανάμεσα από πολλά μοντέλα. Το γεγονός αυτό μπορεί να αναβαθμίσει την εργασία από σύστημα εφαρμογής ενός μοντέλου τεχνητής νοημοσύνης σε μια πλατφόρμα που μπορεί να υποδεχθεί δυναμικά πολλαπλά μοντέλα.

Ο κώδικας που περιγράφεται στο παρόν για τα τρία υποσύστημα μπορεί να βρεθεί στο αντίστοιχο GitHub repository στη διεύθυνση https://github.com/stellatanti.

# Βιβλιογραφία

1. Wang, Xinrui. White-box-Cartoonization Repository. *Github.* [Ηλεκτρονικό] 2020. https://github.com/SystemErrorWang/White-box-Cartoonization.

2. Dick, Stephanie. Artificial intelligence. *HDSR.* [Ηλεκτρονικό] 2 July 2019. https://hdsr.mitpress.mit.edu/pub/0aytgrau/release/2?readingCollection=672db545.

3. *Artificial Intelligence (AI) applications for COVID-19 pandemic.* Vaishya, Raju, et al. Diabetes & Metabolic Syndrome: Clinical Research & Reviews, 2020, Τόμ. 14.4.

4. Google. *Google Assistant.* [Ηλεκτρονικό] Google, 2021. https://assistant.google.com/.

5. Apple. *Siri.* [Ηλεκτρονικό] Apple, 2021. https://www.apple.com/siri/.

6. Amazon. Alexa. *Alexa.* [Ηλεκτρονικό] Amazon, 2021. https://www.alexa.com/.

7. Kröse, Ben, et al. An introduction to neural networks. *An introduction to neural networks.* s.l. : Kröse, Ben, et al., 1993.

8. Google. TensorFlow Tutorials. *TensorFlow.* [Ηλεκτρονικό] 2021. https://www.tensorflow.org/tutorials.

9. —. TensorFlow Platforms. [Ηλεκτρονικό] Google, 2021. https://www.tensorflow.org/learn.

10. —. Pix2Pix. *TensorFlow.* [Ηλεκτρονικό] Google, 2021. https://www.tensorflow.org/tutorials/generative/pix2pix.

11. *Image-to-Image Translation with Conditional Adversarial Networks.* Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. s.l. : arxiv, 2018. 1611.07004.

12. Con, PiVi &. Aging Booth. *Play Store.* [Ηλεκτρονικό] 2021. https://play.google.com/store/apps/details?id=com.piviandco.agingbooth&utm_source=www.apk4fun.com.

13. Co, PiVi &. FatBooth. *Play Store.* [Ηλεκτρονικό] 2021. https://play.google.com/store/apps/details?id=com.piviandco.fatbooth.

14. —. BaldBooth. *Play Store.* [Ηλεκτρονικό] 2021. https://play.google.com/store/apps/details?id=com.piviandco.baldbooth.

15. studios, Lirebird. ToonApp. *Play Store.* [Ηλεκτρονικό] 2021. https://play.google.com/store/apps/details?id=com.lyrebirdstudio.cartoon&hl=en_US&gl=US.

16. Google. *Android Studio.* [Ηλεκτρονικό] Google, 2021. https://developer.android.com/studio.

17. WikiPedia. JAVA. *WikiPedia.* [Ηλεκτρονικό] 2021. https://el.wikipedia.org/wiki/Java.

18. Google. Kotlin. *Adroid developer.* [Ηλεκτρονικό] Google, 2021. https://developer.android.com/kotlin?gclid=Cj0KCQiAhP2BBhDdARIsAJEzXlGonHso50_3gqAjp60WHHhrJJQgnWYgLqUJrw9n57VZeSOA6dzlvyMaAmb6EALw_wcB&gclsrc=aw.ds.

19. *Proxmox.* [Ηλεκτρονικό] 2021. https://www.proxmox.com/.

20. *NodeJs.* [Ηλεκτρονικό] 2021. https://nodejs.org/.

21. WikiPedia. *Javascript.* [Ηλεκτρονικό] 2021. https://el.wikipedia.org/wiki/JavaScript.

22. Microsoft. *Typescript.* [Ηλεκτρονικό] Microsoft, 2021. https://www.typescriptlang.org/.

23. HTTP Protocol. *WikiPedia.* [Ηλεκτρονικό] 2021. Hypertext_Transfer_Protocol.

24. HTTP Headers. *WikiPedia.* [Ηλεκτρονικό] 2021. https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#Request_fields.

25. Git. *WikiPedia.* [Ηλεκτρονικό] 2021. https://en.wikipedia.org/wiki/Git.

26. Community, Git. Git SCM. *Git SCM.* [Ηλεκτρονικό] 2021. https://git-scm.com/.

27. GitLab. *GitLab.* [Ηλεκτρονικό] 2021. https://about.gitlab.com/.

28. GitHub. *GitHub.* [Ηλεκτρονικό] 2021. https://github.com/.

29. Atlassian. Source Tree. *Source Tree.* [Ηλεκτρονικό] Atlassian, 2021. https://www.sourcetreeapp.com/.