

**Αναφορά Έργου**  
**ΣΥΣΤΗΜΑ Covid19-Stats**

**ΕΑΠ - ΠΛΗ 24**  
**3<sup>η</sup> ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ 2020-2021**

No	Ονοματεπώνυμο	A.M.
1	Μπέρκοβιτς- Ιωαννίδης Βασίλειος	Std135481
2	Δεϊρμεντζίδης Νικόλαος	std128011
3	Παναγιωτόπουλος Κυριάκος	std119237
4	Σιακαμπένης Βασίλειος	Std128314
ΣΧΟΛΙΑ ΠΡΟΣ ΤΟΝ ΚΑΘΗΓΗΤΗ		

**Υπεύθυνη Δήλωση Φοιτητή:** Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία αυτής της εργασίας, είναι πλήρως αναγνωρισμένη και αναφέρεται, είτε στο σημείο «Σχόλια προς καθηγητή», είτε μέσα στην εργασία. Επίσης, έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς, είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη Θεματική Ενότητα.

☒ Συμφωνώ και αποδέχομαι την ανωτέρω δήλωση

☐ Δε συμφωνώ και δεν αποδέχομαι την ανωτέρω δήλωση (στην περίπτωση αυτή, ο Κ-Σ έχει δικαίωμα να μην αξιολογήσει την εργασία του φοιτητή)

Ημερομηνία ανακοίνωσης εργασίας:, 26/01/2021

Ημερομηνία παράδοσης εργασίας: 10/03/2021

Καταληκτική ημερομηνία παραλαβής: 10/03/2021, 11:59 μ.μ.

Καταληκτική ημερομηνία παραλαβής σε περίπτωση ατομικής παράδοσης:

**ΔΕΝ ΘΑ ΔΟΘΟΥΝ ΠΑΡΑΤΑΣΕΙΣ**

ΣΤΟΙΧΕΙΑ ΠΟΥ ΣΥΜΠΛΗΡΩΝΕΙ Ο ΚΑΘΗΓΗΤΗΣ

<b>ΗΜΕΡΟΜΗΝΙΑ ΑΞΙΟΛΟΓΗΣΗΣ</b>	
<b>ΒΑΘΜΟΣ</b>	

**ΣΧΟΛΙΑ ΠΡΟΣ ΦΟΙΤΗΤΗ / ΦΟΙΤΗΤΡΙΑ**

**Αναλυτική Αξιολόγηση**

<u>Άσκηση</u>	<u>Περιγραφή</u>	<u>Ποσοστό</u>	<u>Βαθμός</u>
ΔΡΑΣΤΗΡΙΟΤΗΤΑ 1	Διαχείριση του Έργου με έμφαση στη χρήση του εργαλείου συνεργασίας	25	
ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2.Α	Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java (περιλαμβάνει χρήση API και επεξεργασία json)	20	
ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2.Β	Δημιουργία GUI Εφαρμογής	15	
ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2.Γ	Παρουσίαση Αποτελεσμάτων / Στατιστικών Στοιχείων	20	
ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2.Δ	Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής	15	
ΔΡΑΣΤΗΡΙΟΤΗΤΑ 3	Κριτικός Απολογισμός του Έργου	5	
	<b>Σύνολο</b>	<b>100</b>	

## ΠΕΡΙΕΧΟΜΕΝΑ

---

<b>1</b>	<b>ΕΙΣΑΓΩΓΗ .....</b>	<b>4</b>
<b>2</b>	<b>ΔΙΑΧΕΙΡΙΣΗ ΕΡΓΟΥ .....</b>	<b>7</b>
2.1	Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση .....	9
2.2	Υπολογισμός των προτεραιοτήτων .....	10
2.3	Χρονοδιάγραμμα του έργου.....	15
2.4	Το product backlog .....	17
2.5	Οργάνωση ομάδος και αναθέσεις αρμοδιοτήτων .....	24
2.6	Παρακολούθηση της προσπάθειας κατά τη διάρκεια του έργου.....	27
2.7	Χρήση εργαλείου trello .....	1
<b>3</b>	<b>ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2 - ΥΛΟΠΟΙΗΣΗ ΕΡΓΟΥ .....</b>	<b>6</b>
3.1	Ερώτημα Α– Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java .....	6
3.2	Ερώτημα Β – Δημιουργία GUI Εφαρμογής .....	27
3.3	Ερώτημα Γ – Παρουσίαση Χάρτη και Γραφημάτων .....	66
3.4	Ερώτημα Δ – Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής.....	71
<b>4</b>	<b>ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ .....</b>	<b>79</b>
<b>5</b>	<b>ΑΝΑΦΟΡΕΣ .....</b>	<b>84</b>

## 1 ΕΙΣΑΓΩΓΗ

Βασικός σκοπός της εφαρμογής είναι να μπορεί ο χρήστης, να βλέπει για τις χώρες που επιθυμεί, αναλυτικά σε διάφορες μορφές όπως σε αναλυτικό πίνακα δεδομένων ανά ημέρα, σε γράφημα ή σε χάρτη, τα δεδομένα που αφορούν την χώρα σε σχέση με την νόσο covid-19.

Οι λειτουργικές απαιτήσεις του συστήματος, είναι οι **R1, R2, R3** και **R4**, όπως αναφέρονται στο έγγραφο προδιαγραφών στην εκφώνηση της εργασίας.

Πέρα από τις λειτουργικές απαιτήσεις που αναφέρονται στην εκφώνηση της εργασίας, για να μπορεί η εφαρμογή να ανταποκρίνεται στον παραπάνω στόχο, προϋποθέτει ένα σύνολο

Μη λειτουργικών απαιτήσεων ως εξής:

1. Η εφαρμογή μπορεί να τρέξει σε σύστημα Microsoft Windows.
2. Για να μπορέσει να εκκινήσει η εφαρμογή Πρέπει να έχει δημιουργηθεί η βάση δεδομένων(Derby) και να είναι ενεργή.
3. Το username και ο κωδικός για το database είναι PLH24 και PLH24 αντίστοιχα
4. Πρέπει να υπάρχει σύνδεση στο ίντερνετ
5. Στον φάκελο του project πρέπει να υπάρχει το αρχείο countriesLatLong.txt που διορθώνει τις συντεταγμένες στον χάρτη για τις χώρες που είναι καταχωρημένες με τις πολιτείες τους ξεχωριστά.
6. Στον φάκελο του project πρέπει να υπάρχει ο κατάλογος html που περιέχει το αρχείο mappage.html.
7. Πρέπει να γίνει εισαγωγή της βιβλιοθήκης 3GE, όπως ορίζεται στην εκφώνηση.

Για τις

Λειτουργικές απαιτήσεις:

Η **R1**, για να ικανοποιηθεί, με την εκκίνηση της εφαρμογής, **εμφανίζεται μια γραφική διεπαφή χρήστη**, με τις βασικές επιλογές, χρησιμοποιώντας κουμπιά, όπου όταν ο χρήστης πατήσει κάποιο κουμπί, στην ουσία μεταφέρεται στην αντίστοιχη διεπαφή, έχοντας την δυνατότητα πλέον να αλληλοεπιδρά με το σύστημα με βάση την εκάστοτε απαίτηση.

Συγκεκριμένα,

- με την επιλογή του κουμπιού «Διαχείριση Δεδομένων Covid», το σύστημα εμφανίζει τη γραφική διεπαφή χρήστη, για την διαχείριση της βάσης δεδομένων,
- με την επιλογή του κουμπιού «Προβολή δεδομένων Covid ανά χώρα», το σύστημα εμφανίζει τη γραφική διεπαφή χρήστη, για την προβολή των δεδομένων σε διάφορες μορφές ανά χώρα,
- με την επιλογή του κουμπιού «Προβολή δεδομένων covid σε χάρτη», το σύστημα εμφανίζει την γραφική διεπαφή χρήστη, για την προβολή δεδομένων σε χάρτη,
- τέλος με την επιλογή του κουμπιού έξοδος, τερματίζεται η εφαρμογή.

Η **R2**, για να ικανοποιηθεί, δίνει την δυνατότητα στον χρήστη με το πάτημα κουμπιών **να διαχειρίζεται τα δεδομένα που έχει η βάση δεδομένων**. Κατά την εκκίνηση της γραφικής διεπαφής, ο χρήστης έχει τη δυνατότητα να βλέπει σε λίστα τις χώρες που είναι ήδη καταχωρημένες στην βάση δεδομένων, μπορεί

επίσης να βλέπει σε πίνακα για ποιες χώρες τι είδος δεδομένων και μέχρι ποια ημερομηνία έχουν καταχωρηθεί coviddata δεδομένα.

Μπορεί επίσης να επιλέγοντας το είδος των δεδομένων που επιθυμεί, να βλέπει σε λίστα τις χώρες, που έχει κατεβάσει το σύστημα, από το ίντερνέτ και να προχωράει σε εισαγωγή δεδομένων covid επιλέγοντας 1 χώρα κάθε φορά.

Συγκεκριμένα,

- επιλέγοντας πρώτα το είδος το δεδομένων που επιθυμεί ο χρήστης, πατώντας το κουμπί «Λήψη δεδομένων», το σύστημα κατεβάζει από το ίντερνέτ τα δεδομένα και εμφανίζει την λίστα με τις χώρες για τις οποίες βρέθηκαν δεδομένα, επίσης κατά την διαδικασία αυτή όλες οι χώρες που βρέθηκαν, αποθηκεύονται στην βάση δεδομένων αν δεν είναι ήδη αποθηκευμένες,
- επιλέγοντας 1 χώρα κάθε φορά και με την προϋπόθεση ότι η χώρα, είναι αποθηκευμένη στην βάση δεδομένων, ενεργοποιείται το κουμπί «Εισαγωγή»,
- πατώντας το κουμπί «Εισαγωγή», εισάγονται στην βάση δεδομένων, τα δεδομένων που μόλις ανακτήθηκαν από το ίντερνέτ για την συγκεκριμένη χώρα και εμφανίζεται στον πίνακα με τα δεδομένα, εγγραφή που αναφέρει το όνομα της χώρας, το είδος των δεδομένων και την ημερομηνία μέχρι την οποία υπάρχουν δεδομένα αποθηκευμένα στην βάση δεδομένων, αν υπήρχαν ήδη δεδομένα με προγενέστερη ημερομηνία, εισάγονται μόνο τα νέα δεδομένα και ενημερώνεται στον πίνακα η ημερομηνία,
- επιλέγοντας από την λίστα με τις αποθηκευμένες χώρες, από 1 και πάνω χώρες, και πατώντας το κουμπί «Διαγραφή Χωρών», εμφανίζεται μήνυμα επιβεβαίωσης στον χρήστη, και αν επιλέξει ναι, γίνεται διαγραφή των Χωρών από την βάση δεδομένων. Στην περίπτωση που έχουν καταχωρηθεί coviddata δεδομένα για κάποια από τις υπό διαγραφή χώρες, το σύστημα πρώτα διαγράφει τα δεδομένα αυτά και μετά τις χώρες,
- πατώντας το κουμπί «Διαγραφή όλων των δεδομένων», το σύστημα διαγράφει πρώτα όλα τα δεδομένα coviddata και μετά όλες τις χώρες από την βάση δεδομένων,
- επιλέγοντας 1 ή παραπάνω εγγραφές από τον πίνακα δεδομένων coviddata και πατώντας το κουμπί «Διαγραφή δεδομένων», το σύστημα διαγράφει όλα τα δεδομένα covid, από την βάση δεδομένων, που αναφέρονται στην χώρα και το είδος δεδομένων, της εγγραφής του πίνακα,
- τέλος πατώντας το κουμπί «Κεντρικό Μενού», το σύστημα εμφανίζει την γραφική διεπαφή χρήστη με τις βασικές επιλογές(R1).

Η **R3**, για να ικανοποιηθεί, δίνει την δυνατότητα στον χρήστη μέσα από την γραφική διεπαφή, προβολής δεδομένων ανά χώρα, όταν επιλέξει μια χώρα, **βλέπει τα δεδομένα της χώρα ανά είδος σε πίνακες**, να βλέπει τα δεδομένα **σε γράφημα** και τέλος να τα βλέπει **σε χάρτη**. Υπάρχει η δυνατότητα για όλες τις παραπάνω επιλογές, να γίνει περιορισμός δεδομένων, σε επιλεγμένο εύρος ημερομηνιών.

Συγκεκριμένα,

- επιλέγοντας χώρα από τον επιλογέα χώρας, το σύστημα ενεργοποιεί όλα τα κουμπιά και την πρώτη κατά σειρά καρτέλα που υπάρχουν δεδομένα για την χώρα αυτή. Στην καρτέλα υπάρχουν σε πίνακα εγγραφές από την πρώτη κατά ημερομηνία καταχώρηση, μέχρι την τελευταία, εμφανίζοντας στον χρήστη για κάθε ημερομηνία, το σωρευτικό πλήθος καταγραφών και το ημερήσιο,

- πατώντας το κουμπί ημερολογίου από, εμφανίζεται modal, σε μορφή ημερολογίου, ο χρήστης μπορεί να πλοηγηθεί σε ημερομηνίες από την 1<sup>η</sup> καταγραφή στην βάση δεδομένων, μέχρι και την τελευταία και να επιλέξει, κάποια ημερομηνία, ως ημερομηνία «από». Μετά από κάθε επιλογή, το σύστημα ελέγχει αν η ημερομηνία από είναι πριν την ημερομηνία έως, ώστε να ενεργοποιηθεί το κουμπί αναζήτηση,
- πατώντας το κουμπί ημερολογίου έως, εμφανίζεται modal, σε μορφή ημερολογίου, ο χρήστης μπορεί να πλοηγηθεί σε ημερομηνίες από την επιλεγμένη ημερομηνία «από», μέχρι και την τελευταία ημερομηνία καταγραφής στην βάση δεδομένων και να επιλέξει, κάποια ημερομηνία, ως ημερομηνία «έως». Μετά από κάθε επιλογή, το σύστημα ελέγχει αν η ημερομηνία από είναι πριν την ημερομηνία έως, ώστε να ενεργοποιηθεί το κουμπί αναζήτηση,
- πατώντας το κουμπί «Αναζήτηση», το σύστημα εμφανίζει τα δεδομένα στους πίνακες, στο επιλεγμένο μόνο εύρος ημερομηνιών. Σε περίπτωση επιθυμίας του χρήστη να προβάλει γράφημα ή χάρτη, το σύστημα θα τα προβάλει με το επιλεγμένο εύρος, αν έχει προηγηθεί το πάτημα του κουμπιού αυτού,
- Πατώντας το κουμπί «Προβολή σε διάγραμμα», το σύστημα πρώτα ελέγχει τις επιλογές διαγράμματος και εμφανίζει τα δεδομένα σε διάγραμμα σε μορφή modal. Ο χρήστης έχει τις επιλογές να επιλέξει, ποια από τα καταχωρημένα δεδομένα θα δει στο διάγραμμα, επίσης επιλέγει αν θέλει η ποσότητες να είναι στα δεδομένα ημέρας ή στα σωρευτικά δεδομένα,
- πατώντας το κουμπί «Προβολή σε χάρτη», εμφανίζεται χάρτης σε μορφή modal, ο οποίος εστιάζει στην επιλεγμένη χώρα. Εμφανίζεται επίσης πάνω στην επιλεγμένη χώρα, πινέζα, όπου αν κάνει κλικ ο χρήστης, εμφανίζεται πλαίσιο κειμένου, που αναφέρει τα δεδομένα στο επιλεγμένο εύρος ημερομηνιών,
- πατώντας το κουμπί «Διαγραφή δεδομένων», το σύστημα διαγράφει όλα τα καταχωρημένα δεδομένα, coviddata, από την βάση δεδομένων για την επιλεγμένη χώρα,
- τέλος πατώντας το κουμπί «Κεντρικό Μενού», το σύστημα εμφανίζει την γραφική διεπαφή χρήστη με τις βασικές επιλογές(R1).

Η **R4**, για να ικανοποιηθεί, δίνει την δυνατότητα στον χρήστη μέσα από την γραφική διεπαφή, **προβολής δεδομένων σε χάρτη**, να επιλέγει 1 χώρα βασική, όπου θα εστιάζει ο χάρτης, και όσες χώρες ακόμη επιθυμεί, από τις υπόλοιπες χώρες, και να προβάλει χάρτη, που θα **εμφανίζει πάνω σε όλες τις επιλεγμένες χώρες** πινέζα, όπου αν κάνει κλικ, θα μπορεί να βλέπει σε πλαίσιο κειμένου, τα δεδομένα της κάθε χώρας, για το επιλεγμένο εύρος ημερομηνιών.

Συγκεκριμένα,

- επιλέγοντας μία βασική χώρα, κατά την πρώτη επιλογή με την εκκίνηση της διεπαφής χρήστη, ενεργοποιείται το κουμπί προβολή χάρτη. Σε κάθε επόμενη επιλογή βασικής χώρας και αν προβάλλεται ήδη χάρτης, ο χάρτης εξαφανίζεται, ώστε να προβληθεί ο επόμενος χάρτης σύμφωνα με τις επιλογές του χρήστη. Σε κάθε επιλογή βασικής χώρας, η επιλεγμένη χώρα, εξαφανίζεται από την λίστα με τις υπόλοιπες χώρες, ώστε να μην μπορεί να γίνει επιλογή μίας χώρας και στις 2 λίστες,
- από την λίστα με τις υπόλοιπες χώρες, ο χρήστης έχει την δυνατότητα να επιλέξει από 0 έως και όλες τις διαθέσιμες χώρες που είναι στη λίστα,
- πατώντας το κουμπί ημερολογίου από, με την ίδια ακριβώς χρήση, όπως και στο R3, ο χρήστης επιλέγει ημερομηνία από, ώστε να περιοριστούν τα δεδομένα που θα προβάλλονται για κάθε χώρα,

- πατώντας το κουμπί ημερολογίου έως, με την ίδια ακριβώς χρήση, όπως και στο R3, ο χρήστης επιλέγει ημερομηνία έως, ώστε να περιοριστούν τα δεδομένα που θα προβάλλονται για κάθε χώρα,
- πατώντας το κουμπί «Προβολή χάρτη», το σύστημα εμφανίζει χάρτη μέσα στην γραφική διεπαφή προβολής δεδομένων σε χάρτη, όπου εστιάζει στην βασική χώρα και έχουν τοποθετηθεί πινέζες, σε κάθε επιλεγμένη χώρα. Με κλικ σε κάποια πινέζα, εμφανίζεται πλαίσιο κειμένου που αναφέρει για το επιλεγμένο εύρος ημερομηνιών, τα δεδομένα της χώρας,
- τέλος πατώντας το κουμπί «Κεντρικό Μενού», το σύστημα εμφανίζει την γραφική διεπαφή χρήστη με τις βασικές επιλογές(R1).

### Παραδοχές

- Λόγω του μεγάλου όγκου δεδομένων, αποφασίστηκε να δίνεται η δυνατότητα στον χρήστη να εισάγει για 1 χώρα κάθε φορά δεδομένα coviddata, στην βάση δεδομένων, ώστε να μην υπάρχουν μεγάλοι χρόνοι αναμονής κατά την εγγραφή δεδομένων.
- Για να μπορεί το σύστημα να αλλάζει τα περιεχόμενα του αρχείου html, για την προβολή του χάρτη, αποφασίστηκε το αρχείο να είναι στον φάκελο του project και όχι μέσα στον φάκελο src, δίνοντας έτσι την δυνατότητα να μπορεί να δημιουργηθεί jar εφαρμογή πλήρως λειτουργική, τοποθετώντας στον ίδιο φάκελο με την εφαρμογή τον φάκελο html που περιέχει το αρχείο.
- Για τις χώρες που είναι καταχωρημένες οι πολιτείες ξεχωριστά, επιλέχθηκε να γίνεται συγκεντρωτική καταχώρηση των δεδομένων, και δημιουργήθηκε txt αρχείο που περιέχει τις συντεταγμένες αυτών των χωρών, ώστε να το διαβάσει το σύστημα και να καταχωρεί τις σε αυτές τις σωστές συντεταγμένες.
- Κατά την επιλογή clean and build το NetBeans εμφανίζει κάποιο σφάλμα, και δεν μπορεί να ολοκληρώσει το build. Δυστυχώς δεν καταφέραμε να εντοπίσουμε το σφάλμα, οπότε για την δημιουργία του φάκελου dist, ώστε να μπορούμε να έχουμε την εφαρμογή σε jar αρχείο, θα πρέπει να γίνει πρώτα δεξί κλικ και compile στο πακέτο boundaryclasses του project και στην συνέχεια η επιλογή build λειτουργεί κανονικά.
- Κατά την προβολή δεδομένων ανά χώρα, λόγω του ότι η προβολή σε χάρτη, αναφέρεται σε μία και μόνο χώρα, αν ο χρήστης επιλέξει MS\_Zaandam ή Diamond\_Princess, τα οποία είναι κρουαζιερόπλοια που είχε γίνει μεγάλη διασπορά του ιού σε ταξίδι, η εφαρμογή θα προβάλει άρθρο από το Ίντερνετ, που αναφέρει το ιστορικό των κρουαζιερόπλοιων αυτών κατά την περίοδο που συνέβη η διασπορά του ιού. Στην προβολή δεδομένων σε χάρτη, λόγω του ότι υπάρχει δυνατότητα επιλογής πολλαπλών χωρών, απλά εμφανίζεται πινέζα στον χάρτη, στο σημείο {0,0} , όπου αν γίνει κλικ σε αυτήν, εμφανίζεται πλαίσιο διαλόγου που αναφέρει τα δεδομένα.
- Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι η 3GE όπως ορίζεται στην εκφώνηση, η eclipseLink (JPA 2.1), η Java DB Driver και η Beans Binding που εισάγεται αυτόματα από το NetBeans.
- Τελευταία στιγμή παρουσιάστηκε ένα πρόβλημα, πιο συγκεκριμένα όταν δοκιμάσαμε να δημιουργήσουμε από την αρχή το database, τα query επέστρεφαν μη συμβατές λίστες. Λόγω έλλειψης χρόνου, αφού πλησιάζει το dead line για την παράδοση της εργασίας, προστέθηκε κώδικας που θα παρουσιαστεί παρακάτω, ώστε να μπορεί να τρέχει η εφαρμογή, αφού δεν ήταν δυνατό να εντοπίσουμε την πηγή του προβλήματος.



Εργαλεία που χρησιμοποιήθηκαν (πέραν των απαιτούμενων από την εκφώνηση):

- Για την επικοινωνία μεταξύ των μελών και ανταλλαγή φακέλων και συναντήσεις το Microsoft Teams
- Για την δημιουργία του προγράμματος και αλλαγές (Version Control) χρησιμοποιήθηκε το Github

## 2 ΔΙΑΧΕΙΡΙΣΗ ΕΡΓΟΥ

### 2.1 Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση

Από την αρχή του έργου έγινε αξιολόγηση των απαιτήσεων ούτως ώστε να γίνει υπολογισμός της απαιτούμενης προσπάθειας (EFFORT) για κάθε απαίτηση με χρήση της μεθόδου Planning Poker για την εφαρμογή της οποίας χρησιμοποιήθηκε το online εργαλείο στην διεύθυνση <https://www.planningpoker.com/> στο οποίο έγινε χρήση των καρτών Fibonacci.

Αφού προηγήθηκε μελέτη των απαιτήσεων της εφαρμογής από την ομάδα ανάπτυξης (Scrum Team) πραγματοποιήθηκε meeting μέσω της γνωστής πλατφόρμας Teams της εταιρείας Microsoft κατά την διάρκεια του οποίου κάποια από τα ζητήματα που προβλημάτισαν την ομάδα ενδεικτικά είναι τα ακόλουθα:

- Τέθηκε ο προβληματισμός αν κάποιος από την ομάδα έχει πρότερη εμπειρία σε παρόμοιο Project.  
Διαπιστώσαμε ότι κανένας από την ομάδα δεν είχε πρότερη εμπειρία σε ανάπτυξη αντίστοιχου Project γεγονός το οποίο μας οδήγησε στο συμπέρασμα ότι η προσπάθεια που θα πρέπει να καταβάλουμε θα πρέπει να διέπτετε από σωστό προγραμματισμό εργασιών και συνέπεια ώστε να μπορέσουμε (στο μέτρο του εφικτού) να καλύψουμε την άνεση που πιθανώς να είχαμε αν υπήρχε εμπειρία.
- Προαπαιτούμενες γνώσεις προκειμένου να ξεκινήσουμε να εργαζόμαστε πάνω στο Project.  
Έγινε αμέσως αντιληπτό ότι έπρεπε να πειραματιστούμε με το JAVA JFrame Form μιας και κάποιοι δεν είχαν καθόλου γνώση του γραφικού περιβάλλοντος καθώς επίσης και με το πώς γίνεται το Bind ώστε να μπορέσουμε να κατανοήσουμε και να συνδέσουμε την Βάση Δεδομένων με το γραφικό περιβάλλον.
- Αν έχουμε γράψει παρόμοιο κώδικα άλλη φορά και την ποσότητα που απαιτείται.  
Διαπιστώσαμε (παρατηρώντας εργασίες προηγούμενων ετών) ότι ο κώδικας που απαιτείται είναι πολύς σε ποσότητα και άγνωστος σε πολλά σημεία για εμάς.
- Αρχικές εργασίες οι οποίες θα μας επιτρέψουν να αρχίσουμε να δοκιμάζουμε το Project  
Καταλήξαμε ότι πρέπει να γίνουν οι απαιτήσεις του 1<sup>ου</sup> Sprint για να μπορέσουμε να δοκιμάσουμε σε πρώτη φάση το Project και να το παρουσιάσουμε στον Ιδιοκτήτη του προϊόντος (Product Owner).
- Πόσο εύκολη είναι η ενοποίηση του Project.  
Παρατηρήσαμε ότι δεν υπάρχει πρότερη εμπειρία από όλα τα μέλη της ομάδας σε αντίστοιχα Project και το όλο εγχείρημα της ενοποίησης θα μας προκαλούσε διάφορα προβλήματα.

Κάθε μέλος της ομάδας ψήφισε με βάση την δική του εμπειρία δίνοντας έτσι την εκτίμησή του για τις «Ανθρωπώρες» που θα κατέβαλε αν την υλοποιούσε. Η απάντηση που έδινε η online πλατφόρμα ήταν ότι πιο κοντά (στην δοθείσα κλίμακα Fibonacci) στον μέσο όρο των ψήφων.

Διαπιστώθηκε ότι δεν είχαμε εργαστεί ξανά σε αντίστοιχα ομαδικά Projects και ότι η όλη διαδικασία αποτελούσε μία νέα πρόκληση για εμάς.

Αποφασίστηκε η εκτέλεση των εργασιών να γίνει σε 3 επαναλήψεις (Sprint) διάρκειας 14 ημερών οι 2 πρώτες και 17 ημερών η 3<sup>η</sup> και τελευταία. Έτσι για το συνολικό έργο απαιτήθηκε διάρκεια 45 ημερών.

Ακολουθεί το σύνολο των «Ανθρωπωρών» που συγκέντρωσε η κάθε απαίτηση καθώς και η αναλυτική παρουσίαση των καρτών που έριξε η ομάδα.

### Your Planning Poker® Game Summary

#### Effort

Story	Story Title	Score
1	[XXL]R1. Δημιουργία του GUI_R1	5
2	[XXL]R2. Δημιουργία κλάσης για την επικοινωνία με τον απομακρισμένο server	8
3	[L]R2. Δημιουργία Κλάσης presenter για το GUI_R2	13
4	[XL]R2. Δημιουργία Κλάσης για την διαχείριση της βάσης δεδομένων	34
5	[L]R2. Δημιουργία GUI για την προδιαγραφή 2	8
6	[XL]R3. Διαμόρφωση GUI , για λειτουργία επιλογής χώρας	34
7	[S]R3. Ενσωμάτωση ημερολογίου σε GUI	8
8	[L]R3. Δημιουργία Πλήκτρα προβολών και εμφάνιση χάρτη και Chart	55
9	[m]R3. Δημιουργία Πλήκτρου Διαγραφή δεδομένων	13
10	[XL]R4.1 Διαμόρφωση GUI για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χώρων	21
11	[m]R4.3 Λειτουργίες Χάρτη	21
12	[XS]Εργασία 1.Εισαγωγή Εισάγετε σχετική εικόνα. Στην Εικόνα θα πρέπει να φαίνεται το χρονοδιάγραμμα του έργου σε μορφή Gantt (που θα έχει κατασκευαστεί με χρήση του σχετικού εργαλείου). Παρουσιάστε με συντομία και με μορφή κειμένου τις βασικές δραστηριότητες της κάθε επανάληψης.	8
13	[XS]Εργασία 2.4 Το product backlog [Παρουσιάστε το χρόνο υλοποίησης ανά απαίτηση/ανά επανάληψη (sprint) σε σύγκριση με αυτά που είχαν προϋπολογιστεί, το διάγραμμα κατανάλωσης προσπάθειας (burnt down chart) ανά sprint, καθώς και την ταχύτητα υλοποίησης (velocity).	13
14	[XS]Εργασία 2.7 Χρήση εργαλείου trello	8
15	[L]Εργασία 3.1 Ερώτημα Α- Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java	21
16	[S]Εργασία 3.2 Ερώτημα Β - Δημιουργία GUI Εφαρμογής	5
17	[XS]Εργασία 3.3 Ερώτημα Γ - Παρουσίαση Χάρτη και Γραφημάτων	34
18	[m]Εργασία 3.4 Ερώτημα Δ - Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής	13
19	[XS]Εργασία 4. ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ	8
20	[XXS]5. ΑΝΑΦΟΡΕΣ- ΒΙΒΛΙΟΓΡΑΦΙΑ	13
21	[XXL]Δημιουργία του βασικού Frame με τις μεθόδους για την προβολή των GUI του συστήματος	2
22	[XXL]Δημιουργία ενός αρχικού διαγράμματος κλάσεων	8
23		5
24		

#### Story 1.

Θεωρήθηκε κατά γενική ομολογία εύκολο, με την ιδέα ότι τα components θα γίνουν drag n drop.

#### Story 2.

Έχοντας πειραματιστεί, πάνω στον κώδικα που μας δόθηκε, δεν θεωρήθηκε ότι θα μας δυσκολέψει

#### Story 3.

Εδώ ίσως είχαμε μια λάθος εκτίμηση, το συγκεκριμένο story είχε να κάνει με τις λειτουργίες της διαχείρισης δεδομένων, μη έχοντας εικόνα, θεωρήθηκε σχετικά μικρό.

#### Story 4.

Δεν είχαμε σαφή εικόνα, και θεωρήθηκε ότι θα πάρει αρκετές μέρες για την υλοποίηση

#### Story 5.

Με την ίδια λογική όπως στο story 1, θεωρήθηκε εύκολο

#### Story 6.

Λόγω των πολλών πληροφοριών του gui θεωρήθηκε πιο χρονοβόρο από τα άλλα 2.

#### Story 7.

Ήταν δοκιμασμένο από πριν, και ήθελε μόνο παραμετροποίηση στα δικά μας θέλω, οπότε θεωρήθηκε ότι ήταν απλή εργασία

#### Story 8.

Εδώ στην ουσία τίτλος ήταν λάθος, έγινε συζήτηση με βάση τις λειτουργίες του GUI της προδιαγραφής 3. Δόθηκε μεγάλη βαθμολογία, ίσως γιατί έπρεπε να σπάσει σε περισσότερες εργασίες. Έχει να κάνει με τον κώδικα του presenter του gui αυτού.

#### Story 9.

Δόθηκε σχετικά μικρή βαθμολογία, γιατί ήδη από την προδιαγραφή R2, θα είχαμε παρόμοια υλοποίηση.

#### Story 10.

Εδώ είχαμε να κάνουμε με την ενσωμάτωση του jfxPanel στο gui της προδιαγραφής R4, και δόθηκε μεγαλύτερη βαθμολογία σε σχέση με τα gui R1 και R2.

#### Story 11.

Εδώ δόθηκε μια σχετικά μεγάλη βαθμολογία, γιατί ήταν λίγο άγνωστο το πως θα υλοποιηθεί η απαίτηση R4 και κα επέκταση οι λειτουργικότητα του gui.

Τα υπόλοιπα Story, έχουν να κάνουν με την εγγραφή της εργασίας, οι βαθμολόγηση έγινε κατά εκτίμηση της ποσότητας που περιέχει κάθε ερώτημα για συγγραφή, παρά για την δυσκολία της υλοποίησης.

Εργασία 3.4 Ερώτημα Δ – Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής

#	Story	Score
11	R4.1 Διαμόρφωση GUI για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χωρών	xl
12	R4.3 Δημιουργία πλήκτρου "Προβολή σε χάρτη"	m
13	Δώστε μια σύντομη περιγραφή του συστήματός. Παρουσιάστε τους βασικούς στόχους του έργου, καθώς και τις παραδοχές σας. Αναφέρετε τα χαρακτηριστικά και τις λειτουργίες που πρέπει να επιτελεί το έργο ώστε να ανταποκρίνεται στις απαιτήσεις και τις ανάγκες του πελάτη. Αναφέρετε με ποιόν/ποιους τρόπο/τρόπους το έργο υλοποιείται αυτές τις απαιτήσεις.	xs
14	Εισάγετε σχετική εικόνα. Στην Εικόνα θα πρέπει να φαίνεται το	5

## 2.2 Υπολογισμός των προτεραιοτήτων

Σε αυτό το κομμάτι της διαχείρισης έργου πραγματοποιήθηκε ο υπολογισμός της προτεραιότητας κάθε απαίτησης με χρήση της μεθόδου Priority Poker η οποία αποτελεί παραλλαγή της Planning Poker και κατά την οποία εμείς σαν συμμετέχοντες δεν επιλέγουμε εκτιμήσεις προσπάθειας για κάθε απαίτηση αλλά επιλέγουμε προτεραιότητες. Για την εφαρμογή της παραπάνω μεθόδου χρησιμοποιήθηκε το online εργαλείο στην διεύθυνση <https://www.planningpoker.com/> στο οποίο έγινε χρήση των καρτών “T-Shirt Sizes”. Το τελικό αποτέλεσμα από την εφαρμογή της μεθόδου φαίνεται στις εικόνες που ακολουθούν και συγκεντρωτικά στους παρακάτω πίνακες ποιες απαιτήσεις πραγματοποιήθηκαν σε κάθε επανάληψη.

### Your Planning Poker® Game Summary

#### Priority

Story	Story Title	Score
1	R1. Δημιουργία του GUI_R1	xxl
2	Εργασία 2.5 Οργάνωση ομάδος και αναθέσεις αρμοδιοτήτων	xxl
3	R2. Δημιουργία κλάσης για την επικοινωνία με τον απομακρισμένο server	xxl
4	R2. Δημιουργία Κλάσης presenter για το GUI_R2	l
5	Δημιουργία Κλάσης για την διαχείριση της βάσης δεδομένων	xl
6	R2. Δημιουργία GUI για την προδιαγραφή 2	l
7	Διαμορφωση GUI , για λειτουργία επιλογής χώρας	xl
8	R3. Ενσωμάτωση ημερολογίου σε GUI	s
9	R3. Δημιουργία Πλήκτρα προβολών και εμφάνιση χάρτη και Chart	l
10	R3. Δημιουργία Πλήκτρου Διαγραφή δεδομένων	m
11	R4.1 Διαμόρφωση GUI για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χώρών	xl
12	R4.3 Δημιουργία πλήκτρου "Προβολή σε Χάρτη"	m
13	Δώστε μια σύντομη περιγραφή του συστήματος. Παρουσιάστε τους βασικούς στόχους του έργου, καθώς και τις παραδοχές σας. Αναφέρετε τα χαρακτηριστικά και τις λειτουργίες που πρέπει να επιτελεί το έργο ώστε να ανταποκρίνεται στις απαιτήσεις και τις ανάγκες του πελάτη. Αναφέρετε με ποιόν/ποιους τρόπο/τρόπους το έργο ικανοποιεί αυτές τις απαιτήσεις.	xs
14	Εισάγετε σχετική εικόνα. Στην Εικόνα θα πρέπει να φαίνεται το χρονοδιάγραμμα του έργου σε μορφή Gantt (που θα έχει κατασκευαστεί με χρήση του σχετικού εργαλείου). Παρουσιάστε με συντομία και με μορφή κειμένου τις βασικές δραστηριότητες της κάθε επανάληψης.	s
15	Θα πρέπει να παρουσιάσετε το product backlog σε μορφή φύλλου εργασίας (xls) που θα περιέχει τρία φύλλα εργασίας, ένα για κάθε επανάληψη, μαζί με τις συμπληρωματικές/βοηθητικές πληροφορίες που αποφασίσατε να συλλέξετε.	xs
16	[Παρουσιάστε το χρόνο υλοποίησης ανά απαίτηση/ανά επανάληψη (sprint) σε σύγκριση με αυτά που είχαν προϋπολογιστεί, το διάγραμμα κατανάλωσης προσπάθειας (burnt down chart) ανά sprint, καθώς και την ταχύτητα υλοποίησης (velocity).	xs
17	Εργασία 2.7 Χρήση εργαλείου trello	xxs
18	Εργασία 3.1 Ερώτημα Α- Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java	l
19	Εργασία 3.2 Ερώτημα Β - Δημιουργία GUI Εφαρμογής	s
20	Εργασία 3.3 Ερώτημα Γ - Παρουσίαση Χάρτη και Γραφημάτων	xs
21	Εργασία 3.4 Ερώτημα Δ - Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής	m
22	Εργασία 4. ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ	xs
23	5. ΑΝΑΦΟΡΕΣ- ΒΙΒΛΙΟΓΡΑΦΙΑ	xxs
24	Δημιουργία του βασικού Frame με τις μεθόδους για την προβολή των GUI του συστήματος	xxl
25	Δημιουργία ενός αρχικού διαγράμματος κλάσεων	xxl

Εδώ οι προτεραιότητες μπήκανε, με βάση τις εργασίες πρέπει να γίνουν πρώτα για να προχωρήσει το έργο και να μπορούν να γίνουν οι επόμενες εργασίες. Παραπάνω φαίνονται οι προτεραιότητες που θέσαμε. Στο διάγραμμα Gantt που ακολουθεί, σε επόμενο ερώτημα, έχουμε λεπτομερή αναπαράσταση, των εργασιών που θεωρήσαμε ότι είναι προ απαιτούμενες για κάποιες άλλες εργασίες.





Απαιτήσεις που πραγματοποιήθηκαν σε κάθε επανάληψη (Sprint)

Sprint 1 (01-14/02/2021)		
A/A	ΑΠΑΙΤΗΣΗ	PRIORITY
1	2.5 Οργάνωση ομάδας και αναθέσεις αρμοδιοτήτων	XXL
2	2.2 Υπολογισμός των προτεραιοτήτων (*)	XXL
3	2.1 Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση (*)	XXL
4	Δημιουργία ενός αρχικού διαγράμματος κλάσεων (*)	XXL
5	Δημιουργία του βασικού Frame με τις μεθόδους για την προβολή των GUI του συστήματος	XXL
6	R.1 Δημιουργία του GUI_R1	XXL
7	Δημιουργία κλάσεων Presenter για τα GUI (*)	L
8	R.2 Δημιουργία κλάσης για την επικοινωνία με τον απομακρυσμένο Server	XXL
9	R.2 Δημιουργία GUI για την προδιαγραφή R2	L
10	R.2 Δημιουργία κλάσης για την διαχείριση της Βάσης Δεδομένων	XL
11	R.2 Λειτουργίες Presenter για το Data Manage View (*)	XXL
12	Debug παραδοτέου 1 <sup>ου</sup> Sprint (*) (**)	XXL

Sprint 2 (15-28/02/2021)		
A/A	ΑΠΑΙΤΗΣΗ	PRIORITY
13	R.3 Διαμόρφωση GUI για λειτουργία επιλογής χώρας	XL
14	R.4.1 Διαμόρφωση GUI για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χωρών	XL

15	R.3 Λειτουργίες Presenter για το GUI R.3(Πλήκτρων προβολών και εμφάνιση χάρτη και Chart)	L
16	R.3 Λειτουργία πλήκτρου "Διαγραφή Δεδομένων" (Λειτουργικότητα)	M
17	R.3 Ενσωμάτωση ημερολογίου σε GUI	S
18	R.4.3 Λειτουργίες Presenter για το GUI R.4-Χάρτη (*)	M
19	Debug παραδοτέου 2 <sup>ου</sup> Sprint (*) (**)	XL

Sprint 3 (01-17/03/2021)		
A/A	ΑΠΑΙΤΗΣΗ	PRIORITY
20	Εργασία 1 Εισαγωγή	XS
21	Εργασία 2.5 Οργάνωση ομάδος και αναθέσεις αρμοδιοτήτων	XS
22	Εργασία 2.7 Χρήση εργαλείου Trello	XXS
23	Εργασία 3.1 Ερώτημα Α-Διάγραμμα κλάσεων και Υλοποίηση Κλάσεων σε Java	L
24	Εργασία 2.1 Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση	XS
25	Εργασία 2.2 Υπολογισμός των προτεραιοτήτων	XS
26	Εργασία 2.3 Χρονοδιάγραμμα του έργου	S
27	Εργασία 2.4 Το Product Backlog	XS
28	Εργασία 3.4 Ερώτημα Δ – Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής	M
29	Εργασία 3.2 Ερώτημα Β – Δημιουργία GUI Εφαρμογής (Συγγραφή)	S
30	Εργασία 3.3 Ερώτημα Γ – Παρουσίαση Χάρτη και Γραφημάτων	XS
31	5. ΑΝΑΦΟΡΕΣ- ΒΙΒΛΙΟΓΡΑΦΙΑ	XXS
32	Εργασία 4. ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ	XS
33	Εργασία 2.6 Παρακολούθηση της προσπάθειας κατά τη διάρκεια του έργου	XS
34	Debug παραδοτέου 3 <sup>ου</sup> Sprint (*) (**)	L

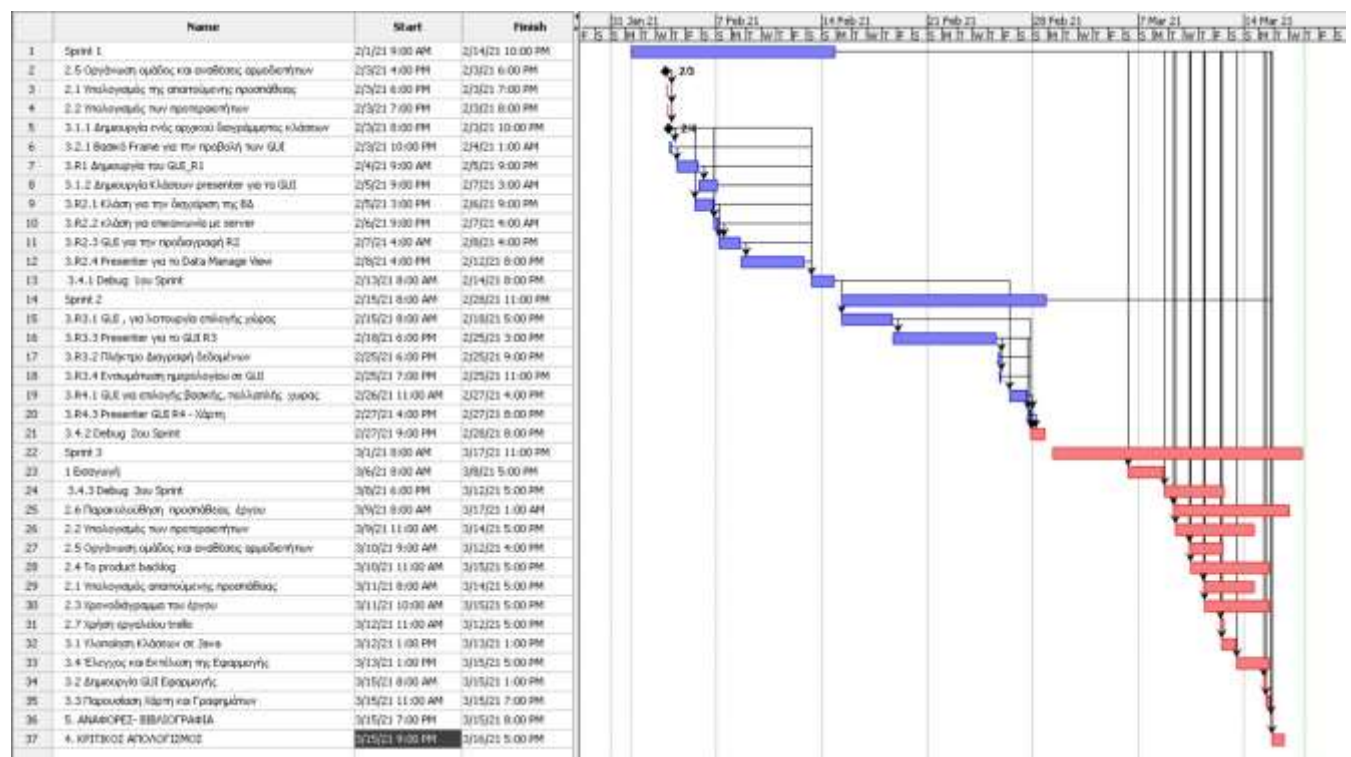
(\*)Απαιτήσεις οι οποίες είναι επιπλέον στις ήδη 25 και προέκυψαν ως ανάγκη κατά την διάρκεια της υλοποίησης.

(\*\*)Τα Debug των παραδοτέων Sprint γίνονταν κάθε Σάββατο.

## 2.3 Χρονοδιάγραμμα του έργου

Η κατασκευή του χρονοδιαγράμματος του έργου σε μορφή Gantt, έγινε με τη χρήση του εργαλείου ProjectLibre (<https://www.projectlibre.com>). Το παραχθέν αρχείο σε μορφή pdf, περιέχεται επισυναπτόμενο στον αντίστοιχο φάκελο της εργασίας μας.

Για την πραγμάτωση του πίνακα χρησιμοποιήθηκαν τα στοιχεία που προέκυψαν από τις ψηφοφορίες του PlanningPoker, σχετικά με τον καταμερισμό και τις προτεραιότητες των απαιτήσεων, ενώ οι χρόνοι και οι εξαρτήσεις μεταξύ τους καταγράφηκαν με το πέρας όλων των επιμέρους εργασιών, όπως φαίνεται παρακάτω.



Στο χρονοδιάγραμμα περιέχονται τα τρία sprint και οι επιμέρους απαιτήσεις με τις ημερομηνίες έναρξης και λήξης καθώς και η σχέση τους με τις προαπαιτούμενές τους. Στο σημείο αυτό να επισημάνουμε ότι οι σχέσεις μεταξύ των απαιτήσεων έχουν σημειωθεί κατά προσέγγιση καθώς εξελίσσονται χρονικά οι απαιτήσεις και ειδικότερα στο 3ο sprint ώστε το διάγραμμα να είναι ευανάγνωστο.

Σύμφωνα λοιπόν με τον παραπάνω πίνακα παρατηρούμε αρχικά ότι δεν υπάρχει επικάλυψη μεταξύ των τριών sprint λόγω του καλού συντονισμού της ομάδας με τη βοήθεια των εργαλείων "Trello" και "MS Teams".

1<sup>ο</sup> sprint, διάρκειας 2 εβδομάδων, από 01/02/2021 έως 14/02/2021



Πιο αναλυτικά, στον πρώτο κύκλο εργασιών η ομάδα εκμεταλλεύτηκε δώδεκα από τις δεκατέσσερις ημέρες καθώς οι δύο πρώτες ημέρες δαπανήθηκαν στην ατομική μελέτη των ζητούμενων από την εργασία.

**2° sprint**, διάρκειας 2 εβδομάδων, από 15/02/2021 έως 28/02/2021

Στο δεύτερο κύκλο παρατηρείται συνολική χρονική κάλυψη χωρίς κενά, καθότι ο κύκλος αυτός αφορά αποκλειστικά την συγγραφή του κώδικα υπό την άρτια εποπτεία και καθοδήγηση του scrum master.

**3° sprint**, διάρκειας 2 εβδομάδων και 3 ημερών, από 01/03/2021 έως 17/03/2021

Τέλος στον τρίτο κύκλο φαίνεται μια πενθήμερη καθυστέρηση στην ανάληψη απαιτήσεων λόγω της επίλυσης των τελευταίων λεπτομερειών επί του κώδικα, του επανασυντονισμού της ομάδας σχετικά με το διαμοιρασμό αρμοδιοτήτων που αφορούν την τελική εικόνα και συγγραφή της εργασίας, αλλά και των εξωγενών υποχρεώσεων των μελών της ομάδας.

## 2.4 Το product backlog

Το product backlog του έργου ανήκει στη φάση της σχεδίασής του και παρουσιάζει τη θεωρητική και όχι την πραγματική πορεία του. Καταρτίστηκε ακολουθώντας τη μεθοδολογία Scrum και προέκυψε βάσει του υπολογισμού ανθρωποωρών και προτεραιοτήτων.

Οι αποκλίσεις λοιπόν του χρονοδιαγράμματος κατά την υλοποίηση, σε σχέση με τις εκτιμώμενες ανθρωποώρες που παρουσιάζονται παρακάτω, περιγράφονται αναλυτικά στην παράγραφο 2.6 .

Οι πληροφορίες για την κατασκευή αρχείου excel ,το οποίο και παρατίθεται στον φάκελο της εργασίας, αντλήθηκαν απευθείας από το εργαλείο «trello».

Στον πίνακα που ακολουθεί φαίνονται οι απαιτήσεις και για τα τρία sprint όπως ορίστηκαν αρχικά από την ομάδα, καθώς και η εκτιμώμενη προτεραιότητα και απαιτούμενη προσπάθεια για την υλοποίησή τους.

ΑΑ	Κωδικός	Απαιτήσεις	Εκτιμώμενη Προτεραιότητα	Εκτιμώμενη Απαιτούμενη Προσπάθεια (ανθρωποώρες)	Ανάθεση σε Sprint
1	Task 2.1	Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση	XXL	3	1ο
2	Task 2.2	Υπολογισμός των προτεραιοτήτων	XXL	3	1ο
3	Task 2.5	Οργάνωση ομάδας και αναθέσεις αρμοδιοτήτων	XXL	5	1ο
4	Task 3.1.1	Δημιουργία ενός αρχικού διαγράμματος κλάσεων	XXL	5	1ο
5	Task 3.2.1	Δημιουργία του βασικού Frame με τις μεθόδους για την προβολή των GUI του συστήματος	XXL	8	1ο
6	Task 3.1.2	Δημιουργία Κλάσεων presenter για τα GUI	L	13	1ο
7	Task 3.R1	Δημιουργία του GUI_R1	XXL	5	1ο
8	Task 3.R2.1	Δημιουργία Κλάσης για την διαχείριση της βάσης δεδομένων	XL	13	1ο
9	Task 3.R2.2	Δημιουργία κλάσης για την επικοινωνία με τον απομακρισμένο server	XXL	8	1ο
10	Task 3.R2.3	Δημιουργία GUI για την προδιαγραφή R2	L	8	1ο
11	Task 3.R2.4	Λειτουργίες Presenter για το Data Manage View	XXL	34	1ο
12	Task 3.4.1	Debug παραδοτέου 1ου Sprint	XXL	13	1ο

13	Task 3.R3.1	Διαμόρφωση GUI , για λειτουργία επιλογής χώρας	XL	34	2ο
14	Task 3.R3.2	Λειτουργία Πλήκτρου Διαγραφή δεδομένων (λειτουργικότητα)	M	13	2ο
15	Task 3.R3.3	Λειτουργίες Presenter για το GUI R3 (Πλήκτρων προβολών και εμφάνιση χάρτη και Chart)	L	55	2ο
16	Task 3.R3.4	Ενσωμάτωση ημερολογίου σε GUI	S	8	2ο
17	Task 3.R4.1	Διαμόρφωση GUI για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χωρών	XL	21	2ο
18	Task 3.R4.3	Λειτουργίες Presenter για το GUI R4 - Χάρτη	M	21	2ο
19	Task 3.4.2	Debug παραδοτέου 2ου Sprint	XL	13	2ο
20	Task 1	Εισαγωγή	XS	8	3ο
21	Task 2.1	Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση(Συγγραφή)	XS	8	3ο
22	Task 2.2	Υπολογισμός των προτεραιοτήτων(Συγγραφή)	XS	8	3ο
23	Task 2.3	Χρονοδιάγραμμα του έργου	S	13	3ο
24	Task 2.4	Το product backlog	XS	8	3ο
25	Task 2.5	Οργάνωση ομάδος και αναθέσεις αρμοδιοτήτων(Συγγραφή)	XS	8	3ο
26	Task 2.6	Παρακολούθηση της προσπάθειας κατά τη διάρκεια του έργου	XS	21	3ο
27	Task 2.7	Χρήση εργαλείου trello	XXS	5	3ο
28	Task 3.1	Ερώτημα Α– Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java	L	34	3ο
29	Task 3.2	Ερώτημα Β – Δημιουργία GUI Εφαρμογής	S	13	3ο
30	Task 3.3	Ερώτημα Γ – Παρουσίαση Χάρτη και Γραφημάτων	XS	8	3ο
31	Task 3.4	Ερώτημα Δ – Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής	M	21	3ο
32	Task 4	ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ	XS	13	3ο
33	Task 5	ΑΝΑΦΟΡΕΣ- ΒΙΒΛΙΟΓΡΑΦΙΑ	XXS	2	3ο
34	Task 3.4.3	Debug παραδοτέου 3ου Sprint	L	13	3ο

Πιο αναλυτικά για κάθε sprint παρουσιάζονται και τα κριτήρια αποδοχής για το κάθε user story.

Έτσι για το **1<sup>ο</sup> Sprint** έχουμε τα εξής στοιχεία:

ΑΑ	Κωδικός	Απαιτήσεις	Κριτήρια αποδοχής
1	Task 2.1	Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση	Προέκυψε μετά από ψηφοφορία με τη χρήση του εργαλείου planning poker και αφού ορίστηκαν αναλυτικά οι απαιτήσεις
2	Task 2.2	Υπολογισμός των προτεραιοτήτων	Προέκυψε μετά από ψηφοφορία με τη χρήση του εργαλείου planning poker και αφού ορίστηκαν αναλυτικά οι απαιτήσεις
3	Task 2.5	Οργάνωση ομάδος και αναθέσεις αρμοδιοτήτων	Έγινε αξιολόγηση των δεξιοτήτων και της πρότερης εμπειρίας των μελών της ομάδας σε σχέση με τα ζητούμενα της εργασίας
4	Task 3.1.1	Δημιουργία ενός αρχικού διαγράμματος κλάσεων	Ανάλυση απαιτήσεων σε συλλογικό επίπεδο και κατασκευή αρχικού διαγράμματος κλάσεων στο netbeans για καλύτερη εποπτεία της δομής του project
5	Task 3.2.1	Δημιουργία του βασικού Frame με τις μεθόδους για την προβολή των GUI του συστήματος	Δημιουργία ενός βασικού Frame το οποίο θα εισάγει στον Container κάθε φορά το Panel(GUI) που έχει επιλέξει ο χρήστης
6	Task 3.1.2	Δημιουργία Κλάσεων presenter για τα GUI	Ανάλυση απαιτήσεων των GUI presenter και δημιουργία αντίστοιχων κλάσεων
7	Task 3.R1	Δημιουργία του GUI_R1	Το GUI περιλαμβάνει αρχικά 2 κουμπιά α. Χρήση πλήκτρων β. Χρήση μενού επιλογών  που παραπέμπουν αντίστοιχα στις 4 παρακάτω επιλογές: 1. Διαχείριση δεδομένων Covid19 2. Προβολή δεδομένων Covid19 ανά χώρα 3. Προβολή δεδομένων Covid19 σε χάρτη 4. Έξοδος
8	Task 3.R2.1	Δημιουργία Κλάσης για την διαχείριση της βάσης δεδομένων	Εισαγωγή Χωρών α) Γίνεται έλεγχος ύπαρξης στη βάση δεδομένων των χωρών στις οποίες αναφέρονται τα δεδομένα. Οι χώρες που δεν υπάρχουν αποθηκεύονται στη βάση δεδομένων. β) Εισάγονται τα δεδομένα τα οποία δεν υπάρχουν στη βάση δεδομένων. Διαγράφονται από τη βάση δεδομένων τα σχετικά δεδομένα. Διαγράφονται από τη βάση δεδομένων οι χώρες. Για τη διαγραφή μιας χώρας απαιτείται να μην υπάρχουν σχετικά δεδομένα.

9	Task 3.R2.2	Δημιουργία κλάσης για την επικοινωνία με τον απομακρισμένο server	Δημιουργία κλάσης που θα επικοινωνεί με τον απομακρυσμένο server όπως δίνεται από την εκφώνηση
10	Task 3.R2.3	Δημιουργία GUI για την προδιαγραφή R2	<p>Πρέπει ο χρήστης να μπορεί να επιλέγει το είδος των δεδομένων που επιθυμεί να αντληθούν από το api.</p> <p>Στην οθόνη υπάρχουν διαθέσιμα τα πλήκτρα «Εισαγωγή χωρών» και «Εισαγωγή δεδομένων».</p> <p>Μετά την εισαγωγή των δεδομένων</p> <p>Στην οθόνη διαχείρισης των δεδομένων εμφανίζονται επιπλέον τα πλήκτρα «Διαγραφή δεδομένων» και «Διαγραφή χωρών».</p> <p>Η ολοκλήρωση οποιασδήποτε διαγραφής δεδομένων γίνεται αφού προηγηθεί ερώτημα επαλήθευσης με επιβεβαίωση από το χρήστη.</p>
11	Task 3.R2.4	Λειτουργίες Presenter για το Data Manage View	Για τη βασική χώρα, αλλά και για κάθε μία χώρα από τις λοιπές δημιουργείται στο χάρτη ένα σχετικό σημάδι (mark) η επιλογή του οποίου εμφανίζει τα δεδομένα της σχετικής χώρας . Ο χάρτης εστιάζει στη βασική χώρα.
12	Task 3.4.1	Debug παραδοτέου 1ου Sprint	Γίνεται γενική επισκόπηση του έργου μέχρι την τρέχουσα στιγμή και πριν την έναρξη του επόμενου κύκλου για τυχόν αστοχίες και λάθη.

Για το 2<sup>ο</sup> Sprint

ΑΑ	Κωδικός	Απαιτήσεις	Κριτήρια αποδοχής
13	Task 3.R3.1	Διαμορφωση GUI , για λειτουργία επιλογής χώρας	Στην οθόνη προβολής των δεδομένων Covid εμφανίζεται πεδίο επιλογής χώρας. Με την επιλογή μιας χώρας εμφανίζονται τα αποθηκευμένα για τη χώρα δεδομένα σε grid σε τρεις διαφορετικές σελίδες.
14	Task 3.R3.2	Λειτουργία Πλήκτρου Διαγραφή δεδομένων (λειτουργικότητα)	Επιλέγοντας τη διαγραφή δεδομένων διαγράφονται όλα τα δεδομένα για την επιλεγείσα χώρα (αφού προηγηθεί ερώτημα επαλήθευσης με επιβεβαίωση από το χρήστη).
15	Task 3.R3.3	Λειτουργίες Presenter για το GUI R3 (Πλήκτρων προβολών και εμφάνιση χάρτη και Chart)	Με την «Προβολή σε διάγραμμα» εμφανίζονται σε διάγραμμα τρεις γραμμές, μία για κάθε κατηγορία δεδομένων (θάνατοι, επιβεβαιωμένα κρούσματα, ασθενείς που έχουν ανακάμψει) με το πλήθος των αντίστοιχων ατόμων. Η εφαρμογή θα πρέπει να δίνει τη δυνατότητα με επιλογή του χρήστη: α) εμφάνισης μιας μόνο από τις τρεις καμπύλες και β) εμφάνισης των σωρευτικών δεδομένων. Με την επιλογή «Προβολή σε χάρτη» εμφανίζεται χάρτης google με σημάδι (mark) στις συντεταγμένες της επιλεγείσας χώρας. Η επιλογή mark εμφανίζει τα δεδομένα της σχετικής χώρας (συνολικό πλήθος θανάτων, κρουσμάτων και ασθενών που έχουν ανακάμψει). Ο χάρτης εστιάζει στη χώρα.
16	Task 3.R3.4	Ενσωμάτωση ημερολογίου σε GUI	Για τον περιορισμό των δεδομένων και την διευκόλυνση του χρήστη δημιουργούνται τα πεδία «Από ημερομηνία» και «Έως ημερομηνία».
17	Task 3.R4.1	Διαμόρφωση GUI για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χωρών	Με την προβολή των δεδομένων Covid σε χάρτη google εμφανίζεται πεδίο επιλογής μιας βασικής χώρας και πεδίο πολλαπλής επιλογής λοιπών χωρών.
18	Task 3.R4.3	Λειτουργίες Presenter για το GUI R4 - Χάρτη	Η προβολή στο χάρτη γίνεται επιλέγοντας το πλήκτρο «Προβολή σε χάρτη».
19	Task 3.4.2	Debug παραδοτέου 2ου Sprint	Γίνεται γενική επισκόπηση του έργου μέχρι την τρέχουσα στιγμή και πριν την έναρξη του επόμενου κύκλου για τυχόν αστοχίες και λάθη.

**Τέλος για το 3<sup>ο</sup> Sprint**

ΑΑ	Κωδικός	Απαιτήσεις	Κριτήρια αποδοχής
20	Task 1	Εισαγωγή	Σύντομη περιγραφή του συστήματος σχετικά με τους βασικούς στόχους του έργου, καθώς και τις παραδοχές μας. Αναφέρονται τα χαρακτηριστικά και οι λειτουργίες που πρέπει να επιτελεί το έργο ώστε να ανταποκρίνεται στις απαιτήσεις και τις ανάγκες του πελάτη και με ποιον τρόπο τις ικανοποιεί.
21	Task 2.1	Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση(Συγγραφή)	Υπολογισμός απαιτούμενης προσπάθειας για το έργο σε ανθρωπόωρες με τη χρήση της μεθόδου planning poker. Τεκμηρίωση παραδοχών, υλικού στο οποίο βασιστήκαμε για την εφαρμογή της μεθόδου, καθώς και τον τρόπο εργασίας.
22	Task 2.2	Υπολογισμός των προτεραιοτήτων(Συγγραφή)	Υπολογισμός προτεραιότητας της κάθε απαίτησης χρησιμοποιώντας τη μέθοδο Priority Poker. Παρουσίαση τελικού αποτελέσματος από την εφαρμογή της μεθόδου, καθώς και παρατηρήσεις σχετικές με τον τρόπο εργασίας.
23	Task 2.3	Χρονοδιάγραμμα του έργου	Χρονοδιάγραμμα του έργου σε μορφή Gantt (με χρήση εργαλείου excel). Παρουσίαση εν συντομία και με μορφή κειμένου των βασικών δραστηριοτήτων της κάθε επανάληψης.
24	Task 2.4	Το product backlog	Παρουσίαση του product backlog σε μορφή φύλλου εργασίας (xls) που θα περιέχει τρία φύλλα εργασίας, ένα για κάθε επανάληψη, μαζί με τις συμπληρωματικές/βοηθητικές πληροφορίες που συλλέξαμε.
25	Task 2.5	Οργάνωση ομάδας και αναθέσεις αρμοδιοτήτων(Συγγραφή)	Ανάλυση της οργάνωσης της ομάδας του έργου, καθώς και των ρόλων των μελών της, όπως επίσης και του τρόπου επικοινωνίας των μελών, τη διαχείριση διαφορών μεταξύ των και των πρακτικών του SCRUM που χρησιμοποιήθηκαν
26	Task 2.6	Παρακολούθηση της προσπάθειας κατά τη διάρκεια του έργου	Παρουσίαση του χρόνου υλοποίησης ανά απαίτηση/ανά επανάληψη (sprint) σε σύγκριση με αυτά που είχαν προϋπολογιστεί, το διάγραμμα κατανάλωσης προσπάθειας (burnt down chart) ανά sprint, καθώς και την ταχύτητα υλοποίησης (velocity).
27	Task 2.7	Χρήση εργαλείου trello	1. Οργάνωση Agile Board, ποιες λίστες χρησιμοποιήθηκαν με ενδεικτικό Screenshot. 2. Ορισμός κατηγοριών δραστηριοτήτων 3. Καταγραφή χρόνου που χρειάστηκε η κάθε δραστηριότητα 4. Συνεργασία ομάδας στο Agile Board 5. Λειτουργίες του Trello και χρηστικότητα αυτών. 6. Συνδέσεις με τρίτες εφαρμογές
28	Task 3.1	Ερώτημα Α– Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java	Εισαγωγή του κατάλληλα μορφοποιημένου και τεκμηριωμένου κώδικα java που έχει γραφτεί από την ομάδα μόνο, για τις σχετικές κλάσεις. Δεν περιλαμβάνονται κομμάτια κώδικα που παράγονται στη διαδικασία

29	Task 3.2	Ερώτημα Β – Δημιουργία GUI Εφαρμογής	Εισαγωγή του κατάλληλα μορφοποιημένου και τεκμηριωμένου κώδικα java που έχει γραφτεί από την ομάδα μόνο, για τις σχετικές κλάσεις. Δεν περιλαμβάνονται κομμάτια κώδικα που παράγονται στη διαδικασία
30	Task 3.3	Ερώτημα Γ – Παρουσίαση Χάρτη και Γραφημάτων	Εισαγωγή του κατάλληλα μορφοποιημένου και τεκμηριωμένου κώδικα java που έχει γραφτεί από την ομάδα μόνο, για τις σχετικές κλάσεις. Δεν περιλαμβάνονται κομμάτια κώδικα που παράγονται στη διαδικασία
31	Task 3.4	Ερώτημα Δ – Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής	Εκτέλεση εφαρμογής και εισαγωγή σε μορφή πίνακα εικόνων (screendumps), όπου θα φαίνεται το αποτέλεσμα της εκτέλεσης.
32	Task 4	ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ	1)τα προβλήματα , 2)τις αποκλίσεις , 3)το χρόνο που απαιτήθηκε σε σχέση με το χρόνο που προϋπολογίσατε, 4)το εργαλείο συνεργασίας trello και με ποιο τρόπο σας βοήθησε στην καθημερινή σας εργασία 5)το πιο πολύτιμο μέλος της ομάδας σας και τον τρόπο που το επιλέξατε, 6)τους κινδύνους που πραγματώθηκαν, καθώς και πώς αποκριθήκατε σε αυτούς, 7)τι θα αλλάζατε σε ένα επόμενο αντίστοιχο έργο, καθώς και 8)ποια ήταν τα θετικά σημεία που αποκομίσατε από αυτή την εργασία
33	Task 5	ΑΝΑΦΟΡΕΣ- ΒΙΒΛΙΟΓΡΑΦΙΑ	Παράθεση βιβλιογραφικών αναφορών (βιβλία, άρθρα, σύνδεσμοι στο διαδίκτυο) που χρησιμοποιήθηκαν για την υλοποίηση της εργασίας
34	Task 3.4.3	Debug παραδοτέου 3ου Sprint	Γενική επισκόπηση συνολικά όλου του έργου από όλα τα μέλη της ομάδας.



## 2.5 Οργάνωση ομάδας και αναθέσεις αρμοδιοτήτων

[Αναφέρετε την οργάνωση της ομάδας του έργου, καθώς και τους ρόλους των μελών της ομάδας. Πιο συγκεκριμένα:

- Ποιος ήταν ο ρόλος του κάθε μέλους της ομάδας;
  - Ποιος ήταν ο υπεύθυνος για την υλοποίηση της κάθε απαίτησης;
  - Αναφέρετε τον τρόπο επικοινωνίας των μελών της ομάδας, τη διαχείριση διαφορών μεταξύ των μελών, κ.λπ. και ποιες από τις πρακτικές του SCRUM χρησιμοποιήσατε;
- Εάν δεν έχετε δώσει απάντηση γράψτε με κεφαλαία γράμματα, ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ. Εάν εν γνώση σας δίνετε ελλιπή απάντηση γράψτε με κεφαλαία γράμματα, ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ]

Κατά την εκπόνηση της εργασίας ακολουθήθηκε η μεθοδολογία SCRUM με μεγάλη έμφαση στην αρχική διερεύνηση και σχεδιασμό της εφαρμογής.

Αποφύγαμε τις παράλληλες εργασίες καθώς αυτό μπορεί να οδηγούσε σε απορρόφηση ενός(η περισσοτέρων μελών) σε μία λύση που δεν θα γινόταν αποδεκτή. Αποφασίστηκε μία μεθοδολογία-αρχική σχεδίαση και ακολουθήθηκε πιστά. Όλες οι εργασίες «έσπασαν» σε κομμάτια καθώς ήταν ξεκάθαρη η αρχική υλοποίηση. Όλα τα μέλη ασχολήθηκαν με πολλά τα κομμάτια του έργου δουλεύοντας παράλληλα με βάση τις εργασίες που συμφωνήθηκαν.

**Ιδιοκτήτης(product owner)** του προϊόντος ήταν ο καθηγητής μας Κός Κούτσικας Χρήστος.

Μας καθοδήγησε ως προς την οργάνωση, έλυσε απορίες και παρακολούθησε την πρόοδο μας μέσω Trello.

**Ο διαχειριστής έργου (Scrum Master)** ο Μπέρκοβιτς Βασίλειος.

Εκτός από την ενεργή συμμετοχή, οργάνωση και καθοδήγηση των υπόλοιπων, ανέλαβε και την εκπαίδευση μας. Οι απαιτήσεις ήταν πολλές και χωρίς τον Βασίλη δεν θα τα είχαμε καταφέρει.

Ουσιαστικά η συμμετοχή του ήταν καθοριστική στην εκτέλεση της εργασίας. Ο Βασίλης επίλυε άμεσα κάθε απορία μας είτε σε ομαδικό είτε σε προσωπικό επίπεδο. Συμμετείχε σε όλα τα στάδια του έργου και ως καθοδηγητής αλλά και γράφοντας-διορθώνοντας το μεγαλύτερο κομμάτι κώδικα.

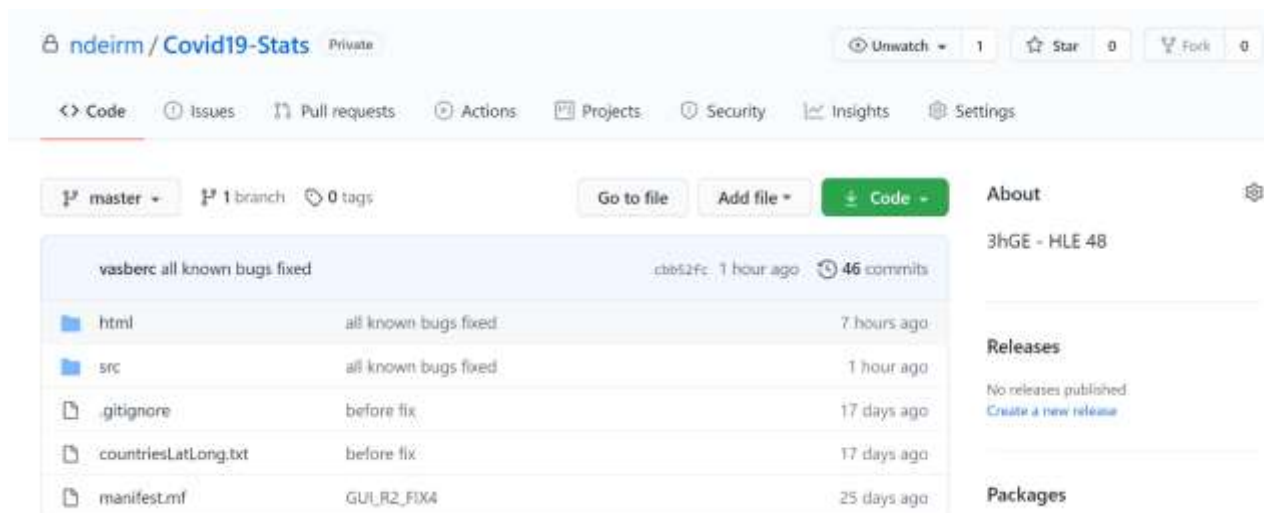
**Η ομάδα ανάπτυξης (Scrum team)** αποτελούμενη από Σιακαμπένη Βασίλειο, Παναγιωτίδη Κυριάκο, Δεϊρμεντζίδη Νικόλαο.

Χωρίς ιδιαίτερη εμπειρία σε ανάπτυξη εφαρμογών προχωρήσαμε με το priority Poker και διαχωρισμό – εκτίμηση των εργασιών.

Κατά την διάρκεια των συναντήσεων συμφωνήθηκαν οι εργασίες που το κάθε μέλος θα μπορούσε να αναλάβει. Λόγω έλλειψης ξεκάθαρων ειδικοτήτων προσαρμόσαμε τις εργασίες σε εύκολες και δύσκολες και ανάλογα με το επίπεδο του κάθε μέλους στο συγκεκριμένο τομέα διαμοιράστηκαν κατάλληλα.

Ο στόχος ήταν εκτός από την γρήγορη εκπόνηση της εργασίας να δοθεί και χρόνος για εκπαίδευση.

Το τελικό κομμάτι κώδικα ελεγχόταν από τον Scrum Master ο οποίος ήταν ο μόνος που το ανέβαζε το Github για αποφυγή διενέξεων στα commits.



Η επικοινωνία μας ήταν καθημερινή γινόταν μέσω Microsoft Teams. Ουσιαστικά πραγματοποιήθηκαν πάνω από 10 προγραμματισμένες, 2ωρες συναντήσεις και αρκετές εμβόλιμες. Το Microsoft Teams ήταν και η πλατφόρμα που ανταλλάσσαμε αρχεία. Ενώ εκεί έγινε και η συγγραφή της εργασίας μέσω κοινού φακέλου.



Οι αρχικές συναντήσεις αφορούσαν τον σχεδιασμό και την εκπαίδευση και μετά προχωρήσαμε στην υλοποίηση.

Παρακάτω οι εργασίες στις οποίες συμμετείχε κάθε μέλος.

ΑΑ	Περιγραφή εργασίας	Εργάστηκαν για την υλοποίηση
1	Ανάλυση έργου , οργάνωση ομάδας, προγραμματισμός προτεραιοτήτων , υπολογισμός απαιτούμενης προσπάθειας, Εκπαίδευση Trello - Teams	Μπέρκοβιτς Βασίλειος , Δεϊρμεντζίδης Νικόλαος, Παναγιωτοπουλος Κυριακος, Σιακαμπένης Βασίλειος
2	Δημιουργία αρχικού - τελικού Διαγράμματος κλάσεων	Μπέρκοβιτς Βασίλειος ,
3	Άνοιγμα λογαριασμού στο Github , αρχική εκπαίδευση, και συγχρονισμός	Μπέρκοβιτς Βασίλειος, Δεϊρμεντζίδης Νικόλαος,

4	Σύνδεση και κλήση covid2019-api, Δημιουργία κλάσεων presenter για GUI - κλάση ServerConnection	Μπέρκοβιτς Βασίλειος , Παναγιωτοπουλος Κυριακος, Δεϊρμεντζίδης Νικόλαος
5	Δημιουργία κλάσεων model για την διαχείριση βάσης δεδομένων και υλοποίηση τους. Δοκιμές bidding με βασικές οθόνες	Μπέρκοβιτς Βασίλειος , Παναγιωτοπουλος Κυριακος, Δεϊρμεντζίδης Νικόλαος
6	Δημιουργία Βάσης Δεδομένων , δημιουργία κλάσης Dbmanager συγχρονισμός δοκιμές Entity manager	Μπέρκοβιτς Βασίλειος,
7	GUI/Δημιουργία- σχεδίαση οθονών και κλάσεων Views του συστήματος	Μπέρκοβιτς Βασίλειος , Σιακαμπένης Βασίλειος
8	Διαμόρφωση GUI για επιλογές χωρών, διασύνδεση με presenter, λειτουργία πλήκτρων προβολής- διαγραφής, ενσωμάτωση ημερολογίου, χάρτη	Μπέρκοβιτς Βασίλειος , Δεϊρμεντζίδης Νικόλαος, Παναγιωτοπουλος Κυριακος, Σιακαμπένης Βασίλειος
9	Συγγραφή εργασίας	Μπέρκοβιτς Βασίλειος , Δεϊρμεντζίδης Νικόλαος, Παναγιωτοπουλος Κυριακος, Σιακαμπένης Βασίλειος
10	Εμφάνιση χάρτη - διαγράμματος , δοκιμές	Μπέρκοβιτς Βασίλειος , Δεϊρμεντζίδης Νικόλαος Σιακαμπένης Βασίλειος
11	Debug εφαρμογής σε 1ο, 2ο & 3το Spring	Μπέρκοβιτς Βασίλειος , Παναγιωτοπουλος Κυριακος, Δεϊρμεντζίδης Νικόλαος

## 2.6 Παρακολούθηση της προσπάθειας κατά τη διάρκεια του έργου

Η παρουσίαση των χρόνων υλοποίησης έχει γίνει σε ξεχωριστό αρχείο excel και παρουσιάζουμε σε screenshot τους πίνακες που δημιουργήθηκαν. Επίσης από τα δεδομένα του αρχείου έχουν δημιουργηθεί και τα διαγράμματα κατανάλωσης προσπάθειας.

Εδώ αξίζει να αναφερθεί ότι λόγω της απειρίας της ομάδας σε ανάλογα project, η εκτιμώμενη προσπάθεια δεν έχει υπολογισθεί σωστά. Η αρχική παραδοχή και χρησιμοποιώντας την ακολουθία Fibonacci, ήταν ότι το μέγιστο (89), θα ήταν ένα ολόκληρο Sprint.

Λόγω της παραπάνω απόκλισης, παρατηρούμε ότι σε κάθε Sprint έχουμε διαφορετικό velocity, που ενδεχομένως δεν παρουσιάζει την πραγματική ταχύτητα υλοποίησης.

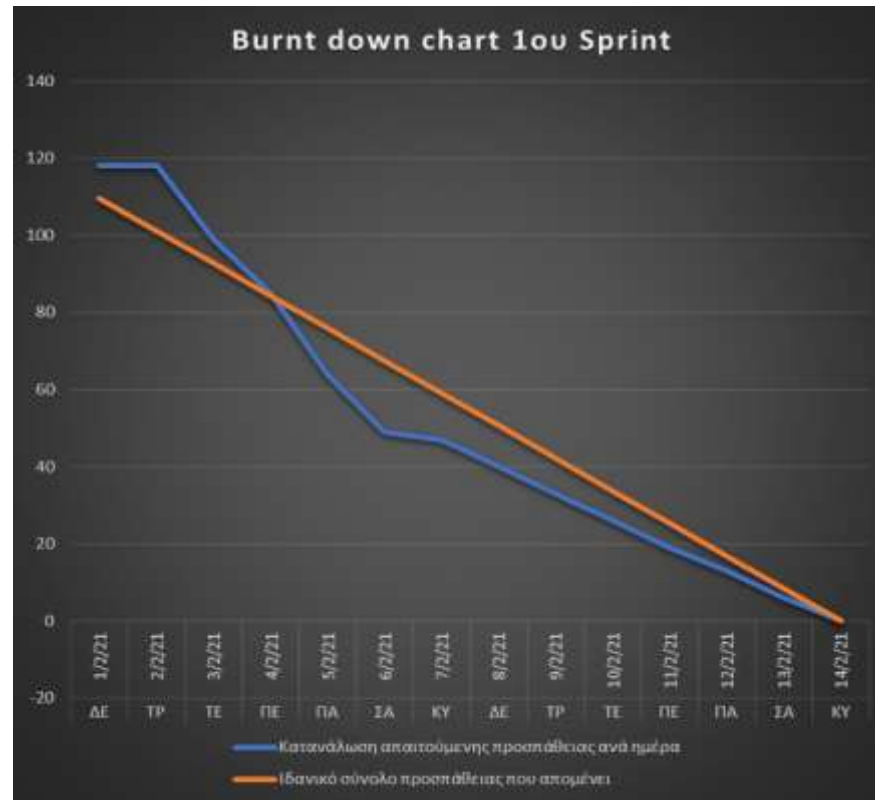
Ο υπολογισμός της ημερήσιας κατανάλωσης προσπάθειας ανά απαίτηση, έχει γίνει προσεγγιστικά, σε σχέση με το ποσοστό υλοποίησης της απαίτησης (π.χ. σε μια απαίτηση με εκτιμώμενη προσπάθεια 5, αν η εκτίμηση προόδου μέσα σε μία μέρα είναι στο 20%, τότε εκείνη τη μέρα η ημερήσια ποσότητα είναι 1).

Να αναφέρουμε επίσης ότι στον προγραμματισμό του 1<sup>ου</sup> sprint είχαμε συμπεριλάβει τις κάρτες, Διαμόρφωση GUI, για λειτουργία επιλογής χώρας, Διαμόρφωση GUI για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χωρών και Λειτουργίες Presenter για το GUI R3 (Πλήκτρων προβολών και εμφάνιση χάρτη και Chart). Όμως έγινε γρήγορα αντιληπτό ότι δεν ήταν εφικτό να υλοποιηθούν κατά το 1<sup>ο</sup> sprint, λόγω χρόνου και τις αφήσαμε για το 2<sup>ο</sup> sprint. Λόγω του ότι αυτή η ενέργεια έγινε συνειδητά και πριν τελειώσει το 1<sup>ο</sup> sprint, αποφασίστηκε να μην τις συμπεριλάβουμε στην παρουσίαση του 1<sup>ου</sup> sprint.

Για τον υπολογισμό του velocity ανά Sprint απλά προσθέτουμε την υπολογισμένη απαιτούμενη προσπάθεια των καρτών που περιλαμβάνει το κάθε Sprint. Στην ουσία και λόγω των παραπάνω παραδοχών, παρατηρείται ότι το velocity είναι ίσο με την συνολική υπολογισμένη απαιτούμενη προσπάθεια και δεν αντιστοιχεί σε πραγματικές ανθρωποώρες.

1° Sprint 1-14/02/2021

Α/Α	Απαίτηση	Εκτιμώμενη απαιτούμενη προσπάθεια	Κατανάλωση απαιτούμενης προσπάθειας ανά ημέρα													
			ΔΕ	ΤΡ	ΤΕ	ΠΕ	ΠΑ	ΣΑ	ΚΥ	ΔΕ	ΤΡ	ΤΕ	ΠΕ	ΠΑ	ΣΑ	ΚΥ
			1/2/21	2/2/21	3/2/21	4/2/21	5/2/21	6/2/21	7/2/21	8/2/21	9/2/21	10/2/21	11/2/21	12/2/21	13/2/21	14/2/21
1	Οργάνωση ομάδας και αναθέσεις αρμοδιοτήτων	5			5											
2	Υπολογισμός των προτεραιοτήτων	3			3											
3	Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση	3			3											
4	Δημιουργία ενός αρχικού διαγράμματος κλάσεων	5			5											
5	Δημιουργία του βασικού Frame με τις μεθόδους για την προβολή των GUI του συστήματος	8			3	5										
6	Δημιουργία του GUI_R1	5				3	2									
7	Δημιουργία Κλάσεων presenter για τα GUI	13				6	7									
8	Δημιουργία κλάσης για την επικοινωνία με τον απομακρυσμένο server	8					5	3								
9	Δημιουργία Κλάσης για την διαχείριση της βάσης δεδομένων	13					7	6								
10	Δημιουργία GUI για την προδιαγραφή R2	8						6	2							
11	Λειτουργίες Presenter για το Data Manage View	34								7	7	7	7	6		
12	Debug παραδοτέου 1ου Sprint	13													7	6
Πραγματικό σύνολο προσπάθειας που απομένει		118	118	118	99	85	64	49	47	40	33	26	19	13	6	0
Ιδανικό σύνολο προσπάθειας που απομένει		118	109,57	101,14	92,71	84,29	75,86	67,43	59,00	50,57	42,14	33,71	25,29	16,86	8,43	0,00

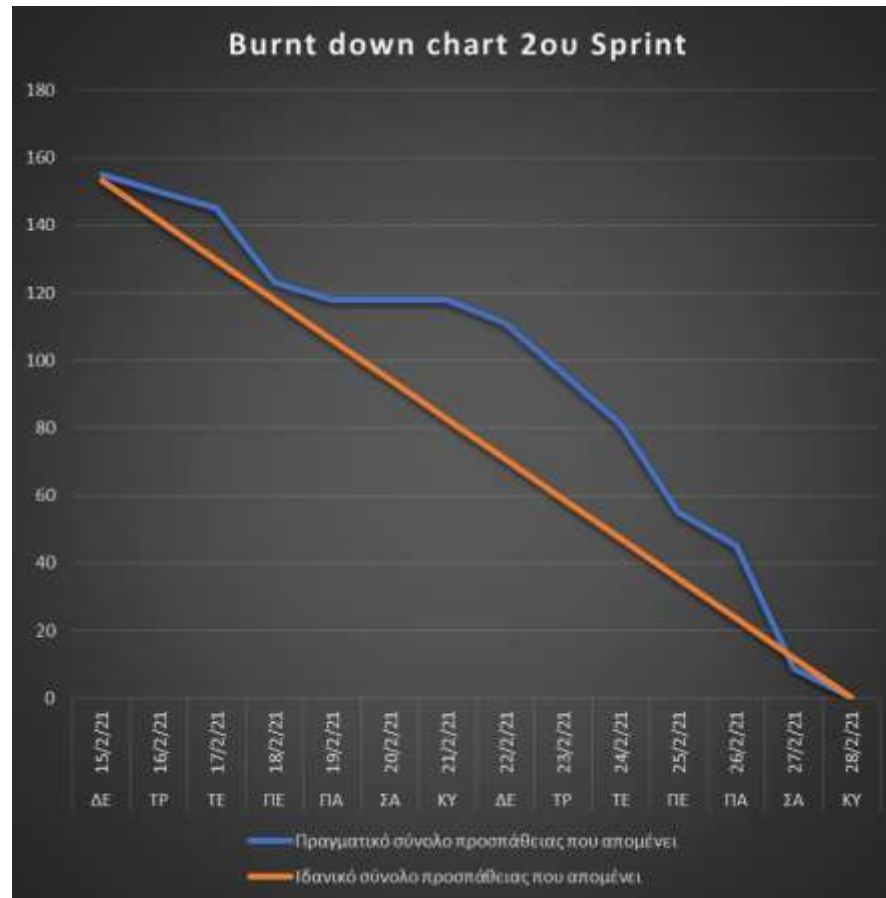


Velocity 1<sup>ου</sup> Sprint

**Velocity = 5+3+3+5+8+5+13+8+8+13+34+13=118**

2° Sprint 15-28/02/2021

Α/Α	Απαίτηση	Εκτιμώμενη απαιτούμενη προσπάθεια	Κατανάλωση απαιτούμενης προσπάθειας ανά ημέρα													
			ΔΕ	ΤΡ	ΤΕ	ΠΕ	ΠΑ	ΣΑ	ΚΥ	ΔΕ	ΤΡ	ΤΕ	ΠΕ	ΠΑ	ΣΑ	ΚΥ
			15/2/21	16/2/21	17/2/21	18/2/21	19/2/21	20/2/21	21/2/21	22/2/21	23/2/21	24/2/21	25/2/21	26/2/21	27/2/21	28/2/21
1	Διαμόρφωση GUI R3, για λειτουργία επιλογής χώρας	34	10	5	5	14										
2	Λειτουργίες Presenter για το GUI R3 (Πλήκτρων προβολών και εμφάνιση χάρτη και Chart)	55				8	5			7	15	15	5			
3	Λειτουργία Πλήκτρου Διαγραφή δεδομένων	13											13			
4	Ενσωμάτωση ημερολογίου σε GUI	8											8			
5	Διαμόρφωση GUI R4 για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χωρών	21												10	11	
6	Λειτουργίες Presenter για το GUI R4 - Χάρτη	21													21	
7	Debug παραδοτέου 2ου Sprint	13													4	9
Πραγματικό σύνολο προσπάθειας που απομένει		165	155	150	145	123	118	118	118	111	96	81	55	45	9	0
Ιδανικό σύνολο προσπάθειας που απομένει		165	153,21	141,43	129,64	117,86	106,07	94,29	82,50	70,71	58,93	47,14	35,36	23,57	11,79	0,00



Στο 2<sup>ο</sup> Sprint παρατηρούμε ότι η προσπάθεια της ομάδας κινήθηκε πάνω από την ιδανική γραμμή. Κατά ένα μεγάλο ποσοστό αυτό οφείλεται στο ότι η απαίτηση 6 είχε αρχικά υπολογισθεί με εκτιμώμενη προσπάθεια 21, ανέβασε την συνολική προσπάθεια, ενώ κατά την υλοποίηση, χρησιμοποιήθηκε σε πολύ μεγάλο ποσοστό κώδικας από την απαίτηση 3. Άλλοι λόγοι έχουν να κάνουν με τις προσωπικές υποχρεώσεις των μελών της ομάδας, όπου δεν ήταν εφικτό να διαθέτουν κάθε μέρα τον ίδιο χρόνο για την περάτωση της εργασίας.

Velocity 2<sup>ου</sup> Sprint

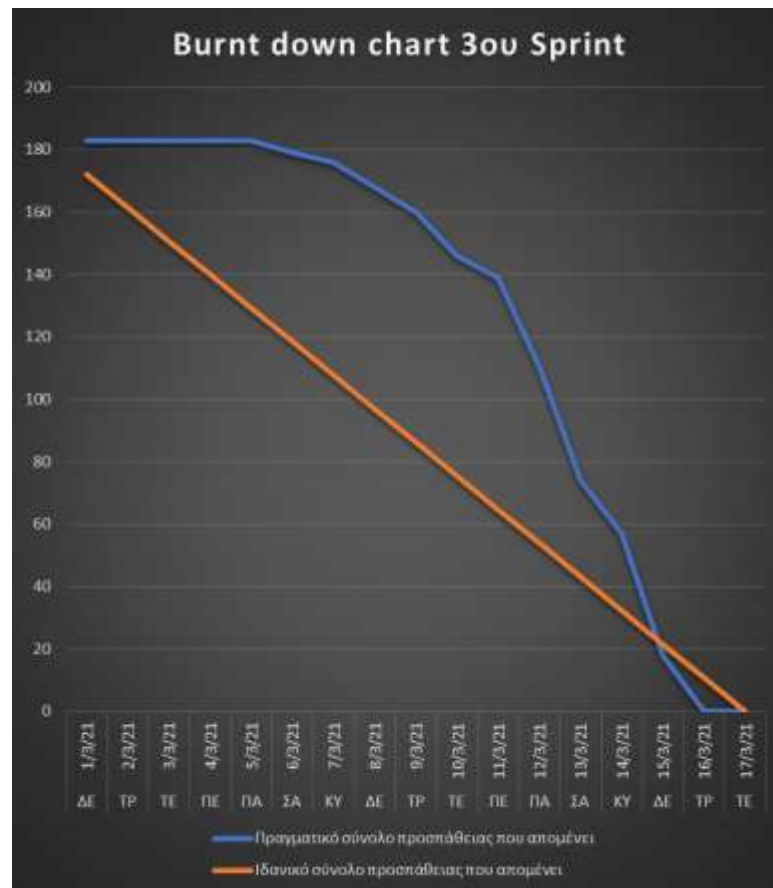
**Velocity = 34+21+55+13+8+21+13=165**



3<sup>ο</sup> Sprint 1-17/03/2021

Α/Α	Απαίτηση	Εκτιμώμενη απαιτούμενη προσπάθεια	Κατανάλωση απαιτούμενης προσπάθειας ανά ημέρα																
			ΔΕ	ΤΡ	ΤΕ	ΠΕ	ΠΑ	ΣΑ	ΚΥ	ΔΕ	ΤΡ	ΤΕ	ΠΕ	ΠΑ	ΣΑ	ΚΥ	ΔΕ	ΤΡ	ΤΕ
			1/3/21	2/3/21	3/3/21	4/3/21	5/3/21	6/3/21	7/3/21	8/3/21	9/3/21	10/3/21	11/3/21	12/3/21	13/3/21	14/3/21	15/3/21	16/3/21	17/3/21
1	Εργασία 1.Εισαγωγή	8						4	3	1									
2	Debug παραδοτέου 3ου Sprint	13								7		3		3					
3	Εργασία 2.6 Παρακολούθηση της προσπάθειας κατά τη διάρκεια του έργου	21									5	9						7	
4	Εργασία 2.2 Υπολογισμός των προτεραιοτήτων	8									3		2	2		1			
5	Εργασία 2.5 Οργάνωση ομάδας και αναθέσεις αρμοδιοτήτων	8										1		7					
6	Εργασία 2.4 Το product backlog	8										1	1	2		2	2		
7	Εργασία 2.1 Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση	8											2	2		4			
8	Εργασία 2.3 Χρονοδιάγραμμα του έργου	13											2	3		3	5		
9	Εργασία 2.7 Χρήση εργαλείου trello	5												5					

10	Εργασία 3.1 Ερώ- τημα Α– Διάγραμμα Κλάσεων και Υλοποί- ηση Κλάσεων σε Java	34												5	29				
11	Εργασία 3.4 Ερώ- τημα Δ – Συνολικός Έλεγχος και Εκτέ- λεση της Εφαρμογής	21													7	7	7		
12	Εργασία 3.2 Ερώ- τημα Β – Δημιουργία GUI Εφαρμογής	13															13		
13	Εργασία 3.3 Ερώ- τημα Γ – Παρουσί- αση Χάρτη και Γρα- φημάτων	8															8		
14	5. ΑΝΑΦΟΡΕΣ- ΒΙ- ΒΛΙΟΓΡΑΦΙΑ	2															2		
15	Εργασία 4. ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ	13															2	11	
Πραγματικό σύνολο προ- σπάθειας που απομένει		183	183	183	183	183	183	179	176	168	160	146	139	110	74	57	18	0	0
Ιδανικό σύνολο προσπά- θειας που απομένει		183	172,24	161,47	150,71	139,94	129,18	118,41	107,65	96,88	86,12	75,35	64,59	53,82	43,06	32,29	21,53	10,76	0,00



Παρατηρούμε ότι και στο 3<sup>ο</sup> Sprint δεν ήμασταν κοντά στην ιδανική καμπύλη, αυτό οφείλεται, αν παρατηρήσουμε και τον πίνακα πιο πάνω, στο ότι λόγω προσωπικών υποχρεώσεων των μελών της ομάδας, κατά την εκκίνηση του Sprint, μέχρι και την Παρασκευή(5 ημέρες), δεν έγιναν καθόλου εργασίες.

Velocity 3<sup>ου</sup> Sprint

**Velocity = 8+13+21+8+8+8+8+13+5+34+21+13+8+2+13=183**

## 2.7 Χρήση εργαλείου trello

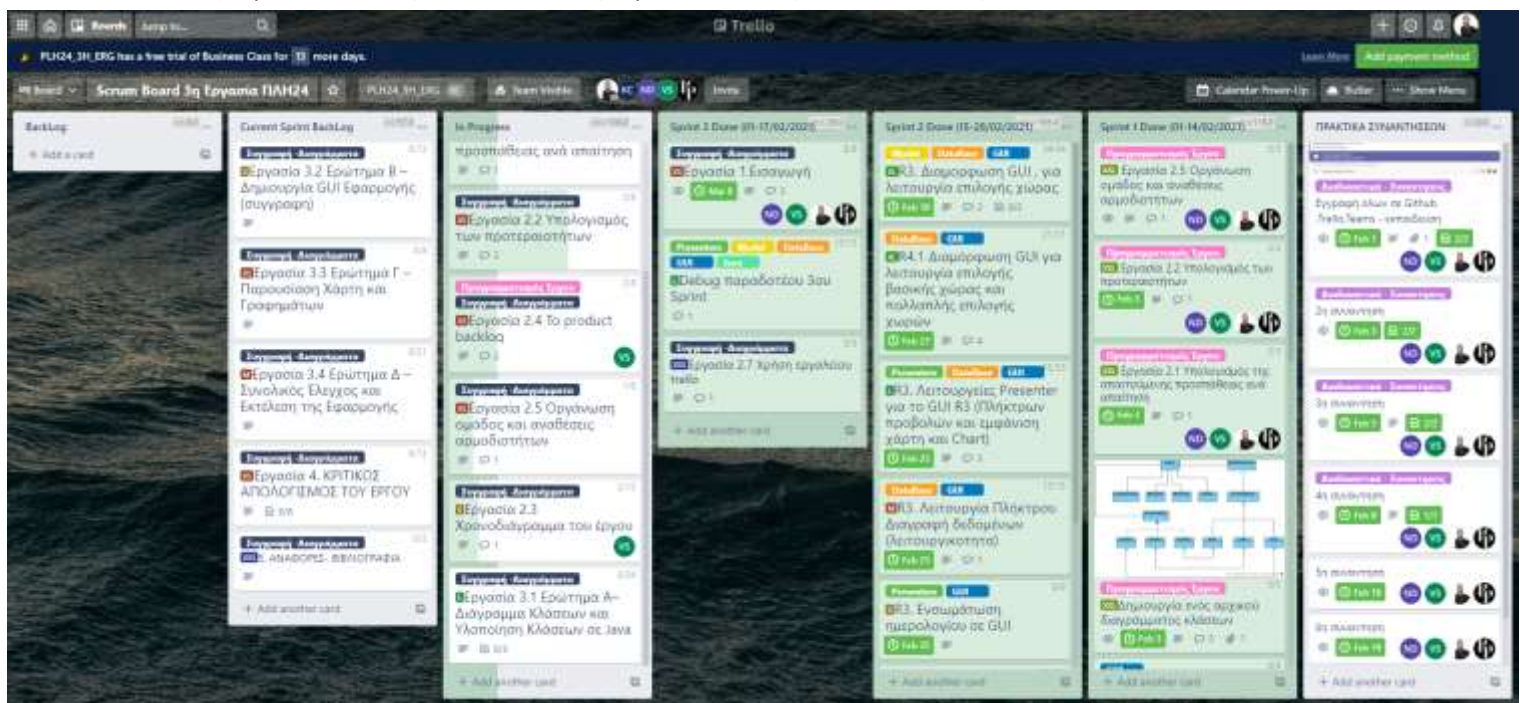
Παρουσιάστε τον τρόπο χρήσης του εργαλείου:

1. Πώς οργανώσατε το Agile Board, ποιες λίστες χρησιμοποιήσατε. Δώστε ενδεικτικό Screenshot.
2. Πώς ορίσατε τις κατηγορίες δραστηριοτήτων
3. Πώς καταγράψατε τον χρόνο που χρειάστηκε η κάθε δραστηριότητα
4. Πώς συνεργάστηκε η ομάδα στο Agile Board
5. Πως εκμεταλλευτήκατε τις λειτουργίες του Trello. Ποιες σας φάνηκες χρήσιμες, ποιες εύκολες στη χρήση και ποιες όχι.
6. Αναλύστε ποιες συνδέσεις με τρίτες εφαρμογές χρησιμοποιήσατε και πώς. Πχ με το github.

Εάν δεν έχετε δώσει απάντηση γράψτε με κεφαλαία γράμματα, ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ. Εάν εν γνώση σας δίνετε ελλιπή απάντηση γράψτε με κεφαλαία γράμματα, ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ]

1)

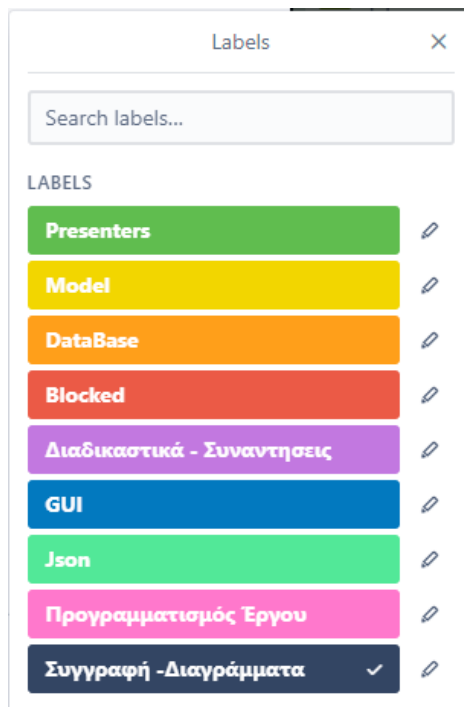
Η οργάνωση του πίνακα έγινε με βάση τις λίστες, Backlog, Current Sprint Backlog, in Progress, Sprint 1 Done (01-14/02/2021), Sprint 2 Done (15-28/02/2021), Sprint 3 Done (01-17/02/2021) και ΠΡΑΚΤΙΚΑ ΣΥΝΑΝΤΗΣΕΩΝ.



- Η λίστα Backlog, στην αρχή συμπληρώθηκε με όλες τις κάρτες που έπρεπε να υλοποιηθούν.
- Η λίστα Current Sprint Backlog, στην εκκίνηση του εκάστοτε Sprint, συμπληρωνόταν μετακινώντας τις κάρτες που θα υλοποιούσε η ομάδα, από την Backlog σε αυτή.
- Στη λίστα in Progress, κάθε μέλος της ομάδας που ξεκινούσε μια εργασία από το Current Sprint Backlog, μετακινούσε και την αντίστοιχη κάρτα σε αυτή.
- Οι λίστες Sprint 1 Done (01-14/02/2021), Sprint 2 Done (15-28/02/2021) και Sprint 3 Done (01-17/02/2021), λειτούργησαν με τον ίδιο τρόπο, ανάλογα το Sprint που έτρεχε, κατά την ολοκλήρωση των εργασιών μιας κάρτας, η κάρτα αυτή μεταφερόταν στην αντίστοιχη done λίστα.
- Η λίστα ΠΡΑΚΤΙΚΑ ΣΥΝΑΝΤΗΣΕΩΝ, χρησιμοποιήθηκε στην αρχή του έργου, για την οργάνωση της ομάδας. Φτιάχναμε μια κάρτα άσχετη με το backlog, για κάθε συνάντηση που προγραμματιζόταν, ώστε να μπορεί κάθε μέλος της ομάδας να αφήνει ένα σχόλιο εκεί, για κάποιο θέμα που έπρεπε να συζητηθεί.

2)

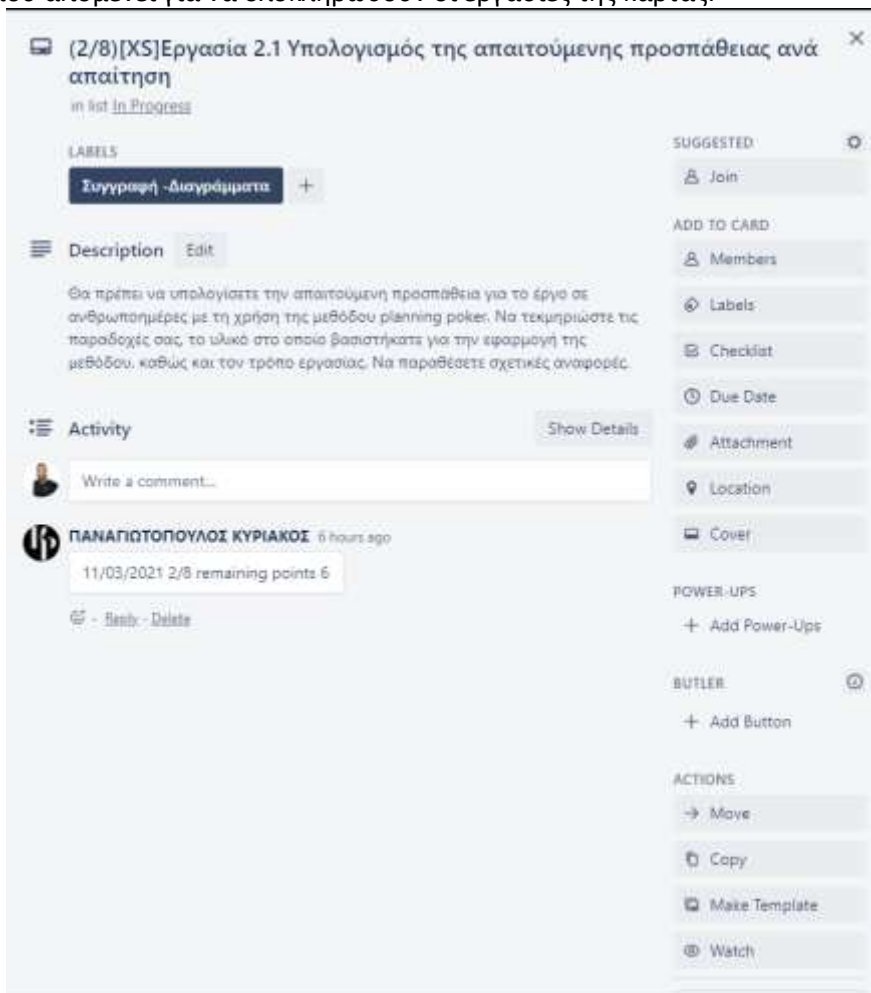
Οι κατηγορίες δραστηριοτήτων χωρίστηκαν με βάση το αντικείμενο της κάθε κάρτας, οι κατηγορίες που προκύψαν είναι: Προγραμματισμός Έργου, Διαδικαστικά – Συναντήσεις, GUI, DataBase, Model, Presenters, Json, Συγγραφή -Διαγράμματα και Blocked. Για κάθε κατηγορία, ορίσαμε και ένα label με διαφορετικό χρώμα, ώστε να μπορεί να μπαίνει σε κάρτες.



- Προγραμματισμός Έργου, στην κατηγορία αυτή ανήκουν οι κάρτες- εργασίες που έπρεπε να γίνουν, στην πρώτη συνάντηση για να μπορέσει να ξεκινήσει το έργο. Παράδειγμα αυτών των εργασιών αποτελεί η κάρτα προσδιορισμού προτεραιοτήτων, όπου έπρεπε να γίνει αρχικά για μπορέσουμε με βάση τις ορισμένες προτεραιότητες να προσθέσουμε κάρτες για το 1° Sprint.
- Διαδικαστικά – Συναντήσεις, η κατηγορία αυτή αναφέρετε στις κάρτες που μπήκαν στην λίστα ΠΡΑΚΤΙΚΑ ΣΥΝΑΝΤΗΣΕΩΝ.
- GUI, στην κατηγορία αυτή ανήκουν όλες οι κάρτες που έχουν να κάνουν με τον σχεδιασμό και την υλοποίηση των κλάσεων Views και σε κάρτες που είχαν να κάνουν με τους Presenters, λόγω του ότι η υλοποίηση τις λειτουργίας κάθε κουμπιού ή event έγινε στις κλάσεις αυτών.
- DataBase, σε αυτή την κατηγορία ανήκουν οι κάρτες όπου έχουν σχέση με τις λειτουργίες εγγραφής ανάγνωσης και διαγραφής από την βάση δεδομένων. Στην κατηγορία αυτή προστέθηκαν και κάποιες κλάσεις από τα Views, διότι για την προβολή κάποιων πινάκων χρησιμοποιήθηκε custom κώδικας στα queries.
- Model, στην κατηγορία αυτή μπήκαν οι κάρτες που είχαν να κάνουν με την δημιουργία των κλάσεων του πακέτου Model.
- Presenters, στην κατηγορία αυτή μπήκαν οι κάρτες που είχαν να κάνουν με την υλοποίηση των Presneters
- Json, στην κατηγορία αυτή μπήκαν οι κάρτες που είχαν να κάνουν με την υλοποίηση της κλάσης που διαβάζει τα δεδομένα σε μορφή json.
- Συγγραφή -Διαγράμματα, στην κατηγορία αυτή μπήκαν οι κάρτες που έχουν να κάνουν με την συγγραφή της εργασίας στο word και την υλοποίηση των ζητούμενων διαγραμμάτων.
- Blocked, η κατηγορία αυτή είχε οριστεί για την διαχείριση της κατά κάποιο τρόπο ασύγχρονης εργασίας της ομάδας. Όταν κατά την υλοποίησης κάποιας εργασίας που ήταν in progress πρόκυπτε κάποιο πρόβλημα και δεν ήταν δυνατό να συνεχιστούν οι εργασίες, θα έμπαινε το label αυτό την κάρτα, για να μπορούν να το δουν τα υπόλοιπα μέλη, μαζί με κάποιο σχόλιο που θα ανέφερε τους λόγους που έχουν μπλοκάρει οι εργασίες.

3)

Για την καταγραφή του χρόνου που χρειάστηκε η κάθε δραστηριότητα χρησιμοποιήθηκε η δυνατότητα να γράφουμε comment στις κάρτες. Κάθε μέρα που τελείωναν οι εργασίες σε μια κάρτα, στα σχόλια γραφόταν η ημερομηνία και η ποσότητα προσπάθειας που απομένει για να ολοκληρωθούν οι εργασίες της κάρτας.



4)  
Για την συνεργασία πάνω στο agile board, κάθε μέλος μπορούσε να πάρει την κάρτα που επιθυμούσε να υλοποιήσει τις εργασίες της. Στις συναντήσεις που γινόντουσαν περίπου 3 φορές την εβδομάδα, προφορικά δήλωνε το κάθε μέλος με ποια κάρτα θα ασχοληθεί τις επόμενες μέρες. Χρησιμοποιήθηκε το addon του web Browser Agile SCRUM for Trello boards, που έδινε την δυνατότητα να έχουμε μια εικόνα για την πρόοδο των εργασιών. Επίσης στα πλαίσια της συνεργασίας της ομάδας, χρησιμοποιήθηκαν και διάφορα comment πάνω στις κάρτες, με προτάσεις για την υλοποίηση των εργασιών.



**(55/55)[L]R3. Λειτουργίες Presenter για το GUI R3 (Πλήκτρων προβολών και εμφάνιση χάρτη και Chart)**  
in list [Sprint 2 Done](#) (15-28/02/2021)

**LABELS**  
Presenters DataBase GUI +

**DUE DATE**  
Feb 25 at 2:55 PM **COMPLETE**

**Description** Edit

Στην οθόνη εμφανίζονται τα πλήκτρα «Προβολή σε διάγραμμα» και «Προβολή σε χάρτη».  
Με την επιλογή «Προβολή σε διάγραμμα» εμφανίζονται σε διάγραμμα τρεις γραμμές, μία για κάθε κατηγορία δεδομένων (θάνατοι, επιβεβαιωμένα κρούσματα, ασθενείς που έχουν ανακάμψει) με το πλήθος των αντίστοιχων ατόμων. Στην εικόνα 3 φαίνεται σχετικό παράδειγμα. Η εφαρμογή θα πρέπει να δίνει τη δυνατότητα με επιλογή του χρήστη: α) εμφάνισης μιας μόνο από τις τρεις καμπύλες και β) εμφάνισης των σωρευτικών δεδομένων.  
Με την επιλογή «Προβολή σε χάρτη» εμφανίζεται χάρτης google με σημάδι (mark) στις συντεταγμένες της επιλεγείσας χώρας. Η επιλογή mark εμφανίζει τα δεδομένα της σχετικής χώρας (συνολικό πλήθος θανάτων, κρουσμάτων και ασθενών που έχουν ανακάμψει). Ο χάρτης εστιάζει στη χώρα.

**Activity** Show Details

Write a comment...

**ΜΠΕΡΚΟΒΙΤΣ-ΙΩΑΝΝΙΔΗΣ ΒΑΣΙΛΕΙΟΣ** Mar 6 at 2:27 PM  
18/02/2021 8/55 remaining points 47  
19/02/2021 13/55 remaining points 42  
22/02/2021 20/55 remaining points 35  
23/02/2021 35/55 remaining points 20  
24/02/2021 50/55 remaining points 5  
25/02/2021 55/55 remaining points 0

**ΜΠΕΡΚΟΒΙΤΣ-ΙΩΑΝΝΙΔΗΣ ΒΑΣΙΛΕΙΟΣ** Feb 15 at 1:39 PM  
Δεν προλάβουμε στο 1ο sprint θα γίνει στο 2ο

**Nikos Deirmentzidis** Feb 3 at 12:59 AM (edited)  
To διάγραμμα πιθανόν να πρέπει να εμφανιστεί σε άλλο frame

**SUGGESTED**  
Join

**ADD TO CARD**  
Members  
Labels  
Checklist  
Due Date  
Attachment  
Location  
Cover

**POWER-UPS**  
Add Power-Ups

**BUTLER**  
Add Button

**ACTIONS**  
Move  
Copy  
Make Template  
Watch  
Archive  
Share

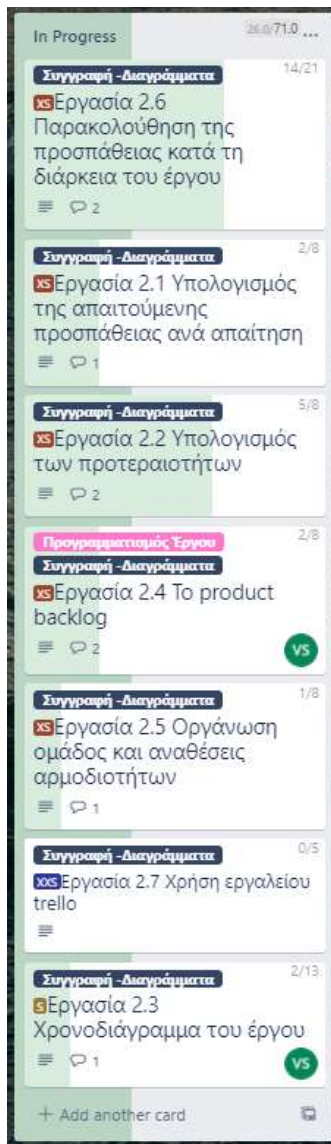
5)  
Οι δυνατότητες του trello χρησιμοποιήθηκαν για τον πλήρη προγραμματισμό του έργου, για κάθε εργασία που έγινε υπήρχε μια κάρτα, οι λίστες που φτιάξαμε μας βοήθησαν πολύ στο να υπάρχει μια εικόνα για την πρόοδο του έργου και όπως αναφέρθηκε και πιο πάνω, οι δυνατότητες σχολιασμού και ετικετών, επίσης χρησιμοποιήθηκαν για την καταγραφή της προόδου και των κατηγοριών των εργασιών αντίστοιχα. Δεν μας δυσκόλεψε η χρήση του trello, όμως λόγω των διαφορετικών υποχρεώσεων, επαγγελματικών και μη, των μελών της ομάδας, σπαταλήθηκε πολύ χρόνος μέχρι να υπάρξει ένας κοινός τρόπος χρησιμοποίησής του.

Στην ουσία δεν χρησιμοποιήσαμε πολλές από τις δυνατότητες του trello πέραν αυτών που προαναφερθήκαν, οπότε δεν υπάρχει κάποια λειτουργία που μπορούμε να πούμε ότι μας δυσκόλεψε.

6)

Η μόνη έξτρα λειτουργία που χρησιμοποιήθηκε είναι το addon Agile SCRUM for Trello boards, που δίνει την δυνατότητα να έχουμε οπτική παρουσίαση της προόδου των εργασιών κάθε κάρτας και επίσης την δυνατότητα να προσθέσουμε μια ετικέτα στον τίτλο της κάρτας, που στην δική μας περίπτωση περιείχε τον βαθμό προτεραιότητας της κάρτας.

Για την πρόοδο έπρεπε για αρχή να βάλουμε στον τίτλο μέσα σε παρένθεση την υλοποιημένη προσπάθεια και την συνολική, σε μορφή (0/8), όπου 0 η υλοποιημένη προσπάθεια και 8 η συνολική. Κάθε φορά που προχωρούσαν οι εργασίες, ανανεώναμε την υλοποιημένη προσπάθεια μέχρι το τέλος της εργασίας που θα είχε την μορφή (8/8). Με ενεργοποιημένο το addon είχαμε μια οπτικοποίηση της προόδου της κάρτας, αλλά και όλης της λίστας, όπως φαίνεται στην εικόνα που ακολουθεί. Για την ετικέτα της προτεραιότητας απλά στον τίτλο έπρεπε να βάλουμε μέσα σε square brackets, το σύμβολο προτεραιότητας, πχ [XXL]. Η εικόνα που ακολουθεί, μας δείχνει την οπτική παρουσίαση που μας δίνει το addon αυτό.

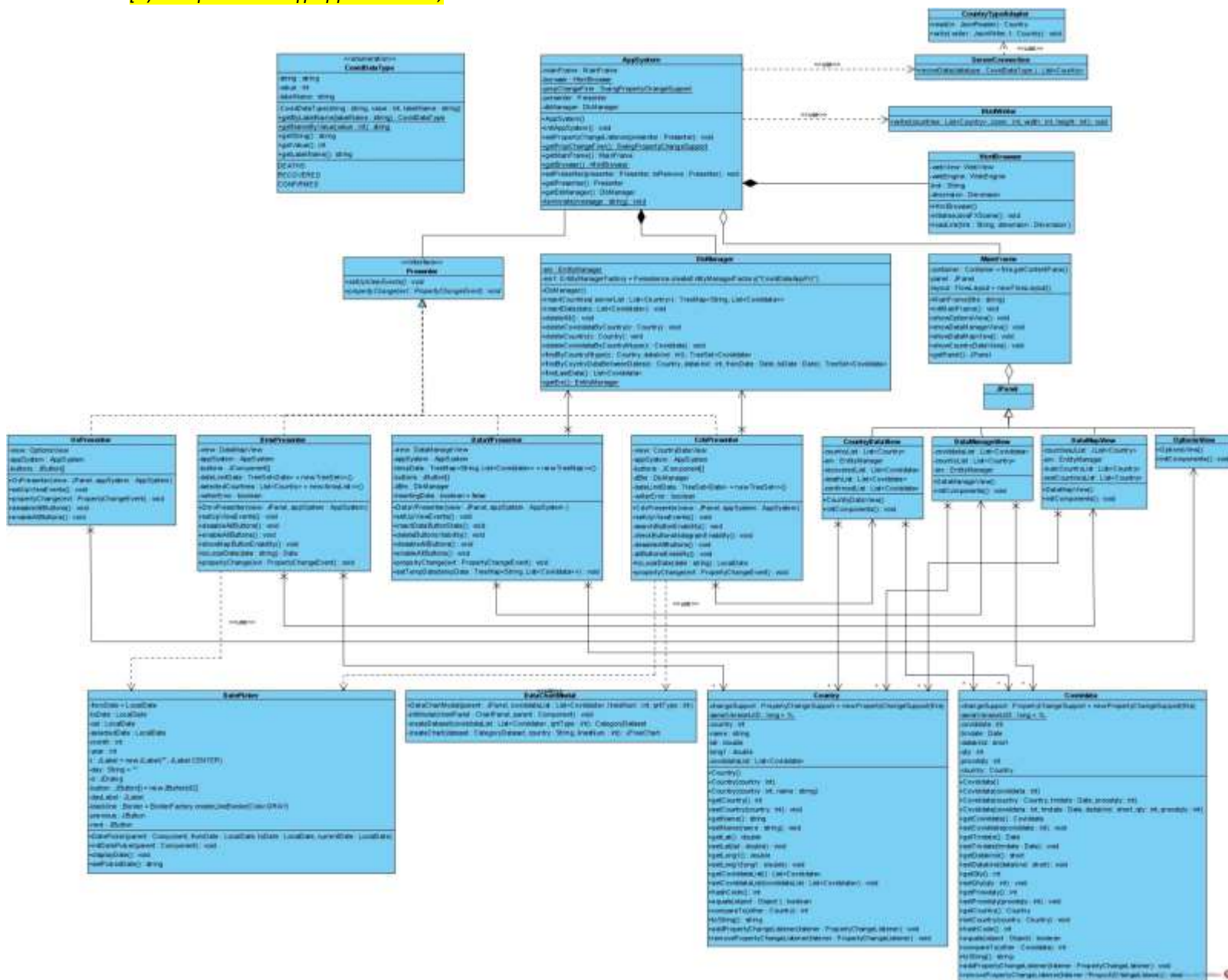




## 3 ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2 - ΥΛΟΠΟΙΗΣΗ ΕΡΓΟΥ

### 3.1 Ερώτημα Α- Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java

[α] Εισάγεται το διάγραμμα κλάσεων,



Στο διάγραμμα κλάσεων και συγκεκριμένα στις κλάσεις των views, για λόγους απλότητας δεν έχουν συμπεριληφθεί τα διάφορα components γραφικών, όπως κουμπιά κτλ. Λόγω του μεγάλου μεγέθους του διαγράμματος, θα παραδοθεί και το αρχείο GE3.vpp που περιέχει τα 2 διαγράμματα που δημιουργήθηκαν κατά την πρόοδο του έργου.

(β) Εισάγεται των κώδικα για τις κλάσεις του συστήματος και

Πακέτο **boundaryclasses**, περιέχει τις κλάσεις που επικοινωνούν με εξωτερικά συστήματα. Η κλάση **DbManager** επικοινωνεί με την βάση δεδομένων και εκτελεί όλες τις ενέργειες εγγραφής, διαγραφής και ανάγνωσης δεδομένων. Η κλάση **HtmlWriter**, γράφει το αρχείο html στον φάκελο του project ή αν η εφαρμογή έχει γίνει build και τρέχει από το jar αρχείο, γράφει το html στον ίδιο φάκελο με το jar. Η κλάση **ServerConnection**, εκτελεί την σύνδεση με το api, και κατεβάζει τα δεδομένα από τον απομακρυσμένο server στο σύστημα. Μέσα στο αρχείο της κλάσης **ServerConnection**, υπάρχει η βοηθητική κλάση **CountryTypeAdapter**, η οποία ορίζει τον τρόπο που διαβάζουμε τα αντικείμενα από το Json string.

```
package boundaryclasses;

import coviddataapp.AppSystem;
import java.sql.SQLNonTransientConnectionException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.TreeMap;
import java.util.TreeSet;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.PersistenceException;
import javax.persistence.TemporalType;
import model.Country;
import model.Coviddata;

/**
 *Χρήσιμη εντολή SQL για το τέλος των δοκιμών ALTER TABLE COVIDDATA ALTER COLUMN
COVIDDATA RESTART WITH 1
 *
 */
public class DbManager {
    private static EntityManager em;//Για να τον χρησιμοποιούμε σε κάθε query στην
εφαρμογή
    private final EntityManagerFactory emf =
        Persistence.createEntityManagerFactory("CovidDataAppPU");

    public DbManager() {
        String oldVal = "",
            newVal = "";

        try {
            em = emf.createEntityManager(); //αρχικοποιώ τη μεταβλητή em
            newVal = "Η σύνδεση με την βάση δεδομένων ολοκληρώθηκε";
            AppSystem.getPropChangeFirer()
                .firePropertyChange("Db login", oldVal, newVal);
        }
        catch (PersistenceException e ) {
            //Μήνυμα αν δεν έχει γίνει σύνδεση στην βάση δεδομένων
            if(e.getCause().getCause() instanceof
                SQLNonTransientConnectionException){
                AppSystem.terminate("<html>Η βάση δεδομένων δεν είναι συνδεδεμένη, η
εφαρμογή θα τερματιστεί.</html>"); //Τερματίζει την εφαρμογή
            }
        }
    }
}
```

```

    }
}
catch (Exception e) {
    //Μήνυμα για οποιαδήποτε άλλη εξαίρεση κατά την αρχικοποίηση του
    DbManager
    e.printStackTrace();
    AppSystem.terminate("<html>Συνέβη ένα αναπάντεχο σφάλμα η εφαρμογής
        θα τερματιστεί</html>");
}
}

public TreeMap<String, List<Coviddata>> insertCountries(List<Country> server-
List){
    //Θα μπουν όλα τα Coviddata και θα επιστραφούν για ενέργειες του χρή-
    στη
    TreeMap<String, List<Coviddata>> data = new TreeMap<>();
    Boolean inserted = false;
    String labelString;
    //Θα μπουν όλες οι χώρες που έχει η βάση δεδομένων για σύγκριση
    List<Country> dbList =
        em.createNamedQuery("Country.findAll", Country.class)
            .getResultList();
    em.getTransaction().begin(); //ξεκινάω μια καινούργια συναλλαγή
    for(Country country: serverList){
        //Εισαγωγή δεδομένων coviddata στο TreeMap
        data.put(country.getName(), country.getCoviddataList());
        //Θέτω null το coviddataList για εισαγωγής της χώρας στην βάση
        δεδομένων
        country.setCoviddataList(null);
        //Θα εισαχθεί αν δεν υπάρχει ήδη στην βάση δεδομένων
        if(!dbList.contains(country)) {
            em.persist(country); // δημιουργώ το query εισαγωγής
            inserted = true;
        }
    }
    em.getTransaction().commit(); //κάνω το commit όταν έχω τελειώσει με τις
    επαναλήψεις
    //Δημιουργώ PropertyChange event για ενέργειες του Presenter
    if(inserted)
        labelString = "Οι χώρες αποθηκεύτηκαν στη βάση δεδομένων επιτυχώς";
    else labelString = "Δεν υπάρχουν νέες χώρες για αποθήκευση";
    AppSystem.getPropChangeFirer()
        .firePropertyChange("Countries insert", "", labelString);
    return data;
}

public void insertData(List<Coviddata> data ){
    //Χρησιμοποιώ TreeSet για να βελτιώσω τους χρόνους
    TreeSet<Coviddata> dbList = new TreeSet<>();
    Boolean inserted = false;
    String labelString;
    dbList.addAll(em.createNamedQuery("Coviddata.findAll", Coviddata.class)
        .getResultList());
    em.getTransaction().begin();

```

```

for(Coviddata d : data){
    //Θα εισαχθεί στην βάση δεδομένων μόνο αν δεν υπάρχει ήδη
    if(!dbList.contains(d)) {
        em.persist(d); // δημιουργώ το query εισαγωγής
        inserted = true;
    }

}

//Κάνω commit 1 φορά για όλες τις εγγραφές
em.getTransaction().commit();
//Δημιουργία PropertyChange event
if(inserted)
    labelString = "Τα δεδομένα αποθηκεύτηκαν στη βάση επιτυχώς";
else labelString = "Δεν υπάρχουν καινούρια δεδομένα για αποθήκευση";
AppSystem.getPropChangeFirer()
    .firePropertyChange("Covid data insert", "", labelString);

}
//Διαγράφει όλα τα δεδομένα
public void deleteAll() {
    em.getTransaction().begin();
    //Διαγράφω από την βάση τα δεδομένα
    em.createNamedQuery("Coviddata.deleteAll").executeUpdate();
    //Διαγράφω από την βάση την χώρα
    em.createNamedQuery("Country.deleteAll").executeUpdate();
    em.getTransaction().commit();
}
//Διαγράφει όλα τα δεδομένα Coviddata για την χώρα που δίνεται σαν όρισμα
public void deleteCoviddataByCountry (Country c) {
    em.getTransaction().begin();
    //Named query με παράμετρο
    em.createNamedQuery("Coviddata.deleteByCountry")
        .setParameter("countryname", c.getName()).executeUpdate();
    em.getTransaction().commit();
}
//Διαγράφει από την βάση δεδομένων την χώρα που περνάει σαν όρισμα
public void deleteCountry(Country c) {
    em.getTransaction().begin();
    em.remove(c);
    em.getTransaction().commit();
}
//Διαγράφει όλα τα ομοειδή δεδομένα με βάση την χώρα και τον τύπο που έχει το
Coviddata που περνάει σαν όρισμα
public void deleteCoviddataByCountryNtype (Coviddata c) {
    em.getTransaction().begin();
    em.createNamedQuery("Coviddata.deleteByCountryNdata")
        .setParameter("countryname", c.getCountry().getName())
        .setParameter("datakind", c.getDatakind()).executeUpdate();
    em.getTransaction().commit();
}
//Βρίσκει για μια χώρα όλα τα δεδομένα με βάση το είδος
public TreeSet<Coviddata> findByCountryNtype (Country c, int datakind) {
    TreeSet<Coviddata> dataList = new TreeSet<>();
    dataList.addAll(
        em.createNamedQuery("Coviddata.findByCountryNdata", Coviddata.class)

```

```

        .setParameter("countryname", c.getName())
        .setParameter("datakind", datakind).getResultList()
    );
    return dataList;
}
//Βρίσκει τα δεδομένα μιας χώρας ανάμεσα σε 2 ημερομηνίες
public TreeSet<Coviddata> findByCountryDataBetweenDates(Country c, int
datakind, Date fromDate, Date toDate) {
    TreeSet<Coviddata> dataList = new TreeSet<>();
    dataList.addAll(
em.createNamedQuery("Coviddata.findByCountryDataBetweenDates", Coviddata.class)
.setParameter("countryname", c.getName())
.setParameter("datakind", datakind)
.setParameter("fromDate", fromDate, TemporalType.DATE)
.setParameter("toDate", toDate, TemporalType.DATE)
.getResultList()
    );
    return dataList;
}
//Βρίσκει για κάθε χώρα που έχει Coviddata στην βάση δεδομένων
//την τελευταία εγγραφή Coviddata με βάση την ημερομηνία
public List<Coviddata> findLastData(){
    List<Country> list =
em.createNamedQuery("Country.findByData", Country.class).getResultList();
    List<Coviddata> lastData = new ArrayList<>();
    for(Country c: list) {
        TreeSet<Coviddata> set = findByCountryNtype(c,1);
        if(!set.isEmpty())
            lastData.add(set.last());
        set = findByCountryNtype(c,2);
        if(!set.isEmpty())
            lastData.add(set.last());
        set = findByCountryNtype(c,3);
        if(!set.isEmpty())
            lastData.add(set.last());
    }
    return lastData;
}
//static μέθοδος για επιστροφή του Entity Manager
public static EntityManager getEm() {
    return em;
}
}

```

```

package boundaryclasses;

import coviddataapp.AppSystem;
import java.io.BufferedWriter;
import java.io.FileOutputStream;

```

```
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.time.Instant;
import java.time.LocalDate;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.TreeSet;
import model.Country;
import model.Coviddata;

public class HtmlWriter {
    //Static μέθοδος για την εγγραφή του html αρχείου
    public static void writer(List<Country> countries, int zoom, int width, int
height) {
        String[] locations = new String[countries.size()];
        int deaths = 0,
            confirmed = 0,
            recovered = 0;
        boolean hasDeaths = false,
            hasConfirmed = false,
            hasRecovered = false;
        TreeSet<Coviddata> coviddata = new TreeSet<>();
        coviddata.addAll(countries.get(0).getCoviddataList());
        //Την βασική χώρα την γράφουμε ξεχωριστά
        for(Coviddata c: coviddata) {
            //Εάν το Coviddata είναι deaths
            if(c.getDatakind() == 1) {
                //πρόσθεσε την ποσότητα
                deaths += c.getQty();
                hasDeaths = true;
            }
            //Εάν το Coviddata είναι recovered
            else if (c.getDatakind() == 2) {
                //πρόσθεσε την ποσότητα
                recovered += c.getQty();
                hasRecovered = true;
            }
            else {//Αλλιώς είναι confirmed
                //πρόσθεσε την ποσότητα
                confirmed += c.getQty();
                hasConfirmed = true;
            }
        }
        //Γράψε στην πρώτη θέση του πίνακα και στο πρώτο πεδίο το όνομα της
        χώρας
        locations[0] = "["+countries.get(0).getName();
        //Εάν υπάρχουν δεδομένα για τις επιλεγμένες ημερομηνίες
        if(!coviddata.isEmpty()) {
            //https://stackoverflow.com/a/48429952/15090628
            locations[0] +=
            ", <br>Δεδομένα από: "+LocalDate.from(Instant.ofEpochMilli(
                coviddata.first().getTrndate().getTime()
            ).atZone(ZoneId.systemDefault()))
            .format(DateTimeFormatter.ofPattern("dd/MM/yyyy"))+
        }
```

```
" έως: "+LocalDate.from(Instant.ofEpochMilli(
    coviddata.last().getTrndate().getTime()
).atZone(ZoneId.systemDefault()))
    .format(DateTimeFormatter.ofPattern("dd/MM/yyyy"));
locations[0] +=", <br>Θάνατοι=";
//Εάν έχει θανάτους
if(hasDeaths)
    //Γράψε τους στο πρώτο πεδίο
    locations[0] += deaths;
else locations[0] += "Δεν υπάρχουν δεδομένα";
locations[0] +=", <br>Κρούσματα=";
//Εάν έχει επιβεβαιωμένα κρούσματα
if(hasConfirmed)
    //Γράψε τα στο πρώτο πεδίο
    locations[0] += confirmed;
else locations[0] += "Δεν υπάρχουν δεδομένα";
locations[0] +=", <br>Ανέκαμψαν=";
//Εάν έχει ανακάμψαντες
if(hasRecovered)
    //Γράψε τους στο πρώτο πεδίο
    locations[0] += recovered;
else locations[0] += "Δεν υπάρχουν δεδομένα";
}
else locations[0] += "<br>Δεν υπάρχουν δεδομένα για τις επιλεγμένες
    ημερομηνίες."
    + "<br>Παρακαλώ επιλέξτε μια προγενέστερη ημερομηνία
    \<br>Από\"";
    + "<br>ή μεταβείτε στην διαχείριση δεδομένων και
    ενημερώστε"
    + "<br>τα δεδομένα της χώρας.";
//Στο δεύτερο και τρίτο πεδίο γράψε lat και long
locations[0] +=", "+countries.get(0).getLat()+"", "
    +countries.get(0).getLong1()+""]";

//Διαπέρασε τις υπόλοιπες χώρες και κάνει τα ίδια
for(int i=1; i<countries.size(); i++) {
    coviddata.clear();
    coviddata.addAll(countries.get(i).getCoviddataList());
    deaths = 0;
    confirmed = 0;
    recovered = 0;
    hasDeaths = false;
    hasConfirmed = false;
    hasRecovered = false;
    for(Coviddata c: countries.get(i).getCoviddataList()) {
        //Εάν το Coviddata είναι deaths
        if(c.getDatakind() == 1) {
            //πρόσθεσε την ποσότητα
            deaths += c.getQty();
            hasDeaths = true;
        }
        //Εάν το Coviddata είναι recovered
        else if (c.getDatakind() == 2) {
```

```

        //πρόσθεσε την ποσότητα
        recovered += c.getQty();
        hasRecovered = true;
    }
    else { //Αλλιώς είναι confirmed
        //πρόσθεσε την ποσότητα
        confirmed += c.getQty();
        hasConfirmed = true;
    }
}

//Γράψε στην πρώτη θέση του πίνακα και στο πρώτο πεδίο το όνομα
της χώρας
locations[i] += "\n          [" + countries.get(i).getName() +
//Εάν υπάρχουν δεδομένα για τις επιλεγμένες ημερομηνίες
if(!coviddata.isEmpty()) {
    locations[i] += ", <br>Δεδομένα από: " + LocalDate.from(
        Instant.ofEpochMilli(coviddata.first().getTrndate().getTime())
    ).atZone(ZoneId.systemDefault())
        .format(DateTimeFormatter.ofPattern("dd/MM/yyyy")) +
        " έως: " + LocalDate.from(
        Instant.ofEpochMilli(coviddata.last().getTrndate().getTime())
    ).atZone(ZoneId.systemDefault())
        .format(DateTimeFormatter.ofPattern("dd/MM/yyyy"));
    locations[i] += ", <br>Θάνατοι=";
    //Εάν έχει θανάτους
    if(hasDeaths)
        //Γράψε τους στο πρώτο πεδίο
        locations[i] += deaths;
    else locations[i] += "Δεν υπάρχουν δεδομένα";
    locations[i] += ", <br>Κρούσματα=";
    //Εάν έχει επιβεβαιωμένα κρούσματα
    if(hasConfirmed)
        //Γράψε τα στο πρώτο πεδίο
        locations[i] += confirmed;
    else locations[i] += "Δεν υπάρχουν δεδομένα";
    locations[i] += ", <br>Ανέκαμψαν=";
    //Εάν έχει ανακάμψαντες
    if(hasRecovered)
        //Γράψε τους στο πρώτο πεδίο
        locations[i] += recovered;
    else locations[i] += "Δεν υπάρχουν δεδομένα";
}
else locations[i] += "<br>Δεν υπάρχουν δεδομένα για τις επιλεγμένες
    ημερομηνίες."
    + "<br>Παρακαλώ επιλέξτε μια προγενέστερη ημερομηνία
    \ "Από\ "
    + "<br>ή μεταβείτε στην διαχείριση δεδομένων και
    ενημερώστε"
    + "<br>τα δεδομένα της χώρας.";
//Στο δεύτερο και τρίτο πεδίο γράψε lat και long
locations[i] += "\n, " + countries.get(i).getLat() + ", "
    + countries.get(i).getLong1() + "]\n";
}
Boolean ioError = false;
//https://stackoverflow.com/a/1001568/15090628

```



```
//https://docs.oracle.com/javase/tutorial/essential/exceptions/tryRe-
sourceClose.html
try (BufferedWriter bw = new BufferedWriter(
    new OutputStreamWriter(
        new FileOutputStream("html\\mappage.html"),
        "UTF-8"
    )
);
) {
    //Γράψε στο αρχείο
    bw.write(
        "<!DOCTYPE html>\n" +
        "<html> \n" +
        "<head> \n" +
        "  <meta http-equiv=\"content-type\" content=\"text/html; charset=UTF-8\" /> \n" +
        "  <title>Google Maps Multiple Markers</title> \n" +
        "  <script src=\"http://maps.google.com/maps/api/js?sensor=false\" \n" +
        "    type=\"text/javascript\"></script>\n" +
        "</head> \n" +
        "<body style=\"margin: 0; padding: 0;\">\n" +
        "  <div id=\"map\" style=\"width: "+width+"px; height: "+height+"px;\"></div>\n" +
        "\n" +
        "  <script type=\"text/javascript\">\n" +
        "    var locations = [\n");
    bw.flush();
    //Βάλε τον πίνακα με τις χώρες
    for(int i=0; i<locations.length; i++) {
        bw.write(locations[i]);
    }
    bw.flush();
    bw.write(
        "\n    ];\n" +
        "\n" +
        "    var map = new google.maps.Map(
        document.getElementById('map'), {\n" +
        "      zoom: "+zoom+",\n" +
        "      center: new google.maps.LatLng("
        +countries.get(0).getLat()
        +", "+countries.get(0).getLongl()+"),\n" +
        "      mapTypeId: google.maps.MapTypeId.ROADMAP\n" +
        "    });\n" +
        "\n" +
        "    var infowindow = new google.maps.InfoWindow();\n" +
        "\n" +
        "    var marker, i;\n" +
        "\n" +
        "    for (i = 0; i < locations.length; i++) { \n" +
        "      marker = new google.maps.Marker({\n" +
        "        position: new google.maps.LatLng(
        locations[i][1], locations[i][2]),\n" +
        "        map: map\n" +
        "      });\n" +

```

```

        "\n" +
        "        google.maps.event.addListener(
        marker, 'click', (function(marker, i) {\n" +
        "        return function() {\n" +
        "        infowindow.setContent(locations[i][0]);\n" +
        "        infowindow.open(map, marker);\n" +
        "        }\n" +
        "        })(marker, i));\n" +
        "        }\n" +
        "    </script>\n" +
        "</body>\n" +
        "</html>"

    );

}
catch (IOException ex) {
    ioError = true;
    System.out.println("ioError");
}
catch (Exception ex) {
    AppSystem.terminate("Συνέβη κάποιο μη αναστρέψιμο σφάλμα κατά την
    εγγραφή του χάρτη,<br>η εφαρμογή θα τερματιστεί");
    System.out.println("Error");
}
finally {
    //Συμβάν firePropertyChange, ενημερώνει τον presenter ότι η
    διαδικασία τελείωσε
    AppSystem.getPropChangeFirer()
        .firePropertyChange("Html writer, oldVal->true,
        newVal->hasIOError",
        true,
        (boolean)ioError);
}
}
}

```

```

package boundaryclasses;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import com.google.gson.TypeAdapter;
import com.google.gson.stream.JsonReader;
import com.google.gson.stream.JsonWriter;
import coviddataapp.AppSystem;
import java.io.IOException;

```

```
import coviddataapp.CovidDataType;
import java.io.File;
import java.sql.Date;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import model.Country;
import model.Coviddata;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class ServerConnection{
    //static μέθοδος για συλλογή δεδομένων
    public static List<Country> reciveData(CovidDataType datatype) {
        //Λίστα που θα μπουν όλα τα δεδομένα
        List<Country> countries = new ArrayList<>();
        //Το URL που θα πάρουμε όλα τα δεδομένα, αλλάζει με βάση το
        CovidDataType που περνάμε σαν όρισμα
        String urlToCall =
            "https://covid2019-api.herokuapp.com/timeseries/"+datatype.getString();
        OkHttpClient client=new OkHttpClient();
        Request request = new Request.Builder().url(urlToCall).build();

        try (Response response = client.newCall(request).execute()) {
            if (response.isSuccessful() && response.body() != null) {
                String responseString=response.body().string();

                JsonParser parser = new JsonParser();
                GsonBuilder gsonBuilder = new GsonBuilder();
                //Βάζω τον custom adapter που έχω φτιάξει
                gsonBuilder.registerTypeAdapter(
                    Country.class, new CountryTypeAdapter());
                Gson gson = gsonBuilder.create();
                //Μετατροπή του responseString σε jsonTree
                JsonElement jsonTree = parser.parse(responseString);
                //Κάνω το jsonTree JsonObject για να χρησιμοποιήσω την get()
                JsonObject jsonObject = jsonTree.getAsJsonObject();
                //Με την get() παίρνω τις χώρες σαν JsonElement που είναι
                εμφωλευμένες στο datatype
                JsonElement jcountries = jsonObject.get(datatype.getString());
                //Διατρέχω τα εμφωλευμένα Element και τα διαβάζω σύμφωνα με τον
                adapter που όρισα
                for(JsonElement jCountry : jcountries.getAsJsonArray()){
                    //Παίρνω το στοιχείο σαν Country
                    Country country = gson.fromJson(jCountry, Country.class);
                    //Εάν υπάρχει ήδη η χώρα με το ίδιο όνομα μέσα στη λίστα
                    if(countries.contains(country)){
                        Country updatedCountry =
                            countries.get(countries.indexOf(country));
                        countries.remove(updatedCountry); //Την αφαιρώ
                        //Ψάχνω τις ίδιες ημερομηνίες για να προσθέσω τις
                        ποσότητες
                        for(Coviddata d: updatedCountry.getCoviddataList())
```

```
        for(Coviddata dNew: country.getCoviddataList()){
            if(d.getTrndate().equals(dNew.getTrndate())){
                d.setProodqty(d.getProodqty()
                    +dNew.getProodqty());
                //κάνω break από την τελευταία for για να
                //ενημερώσω την επόμενη ημερομηνία
                break;
            }
        }
        //Τα θέτω null για να τα φτιάξω μετά από το αρχείο
        if(updatedCountry.getLat() != null) {
            updatedCountry.setLat(null);
            updatedCountry.setLongl(null);
        }
        //Με το τέλος των 2 for βάζω στην λίστα την
        updatedCountry
        countries.add(updatedCountry);
    }
    //Αν δεν υπάρχει η χώρα την βάζω όπως είναι
    else countries.add(country);
}
//Διαπερνάω πάλι τις χώρες για να φτιάξω την ημερήσια ποσότητα
//και να θέσω id
for(Country country: countries){
    //Θέτω id με βάση το όνομα
    country.setCountry(country.getName().hashCode());
    int prevQnt = 0; //Σωρευτική ποσότητα προηγούμενης μέρας
    αρχικοποιείται σε 0 αφού δεν υπάρχει προηγούμενη μέρα
    //Διαπερνάω το coviddatalist της χώρας
    for(Coviddata d: country.getCoviddataList()){
        //Θέτω το datatype σύμφωνα με το όρισμα που περάσαμε
        d.setDatakind((short)datatype.getValue());
        //Θέτω το dayQnt με βάση το prevQnt
        d.setQty(d.getProodqty()-prevQnt);
        //Κάνω το prevQnt να ισούται με το Proodqty της τωρινής
        //μέρας
        prevQnt = d.getProodqty();
    }
    //Εάν η χώρα έχει null στα lat και long, θα τα διαβάσουμε
    //από το αρχείο
    if(country.getLat() == null && country.getLongl() == null)
    {
        File countriesLatLong =
        new File("countriesLatLong.txt");
        Scanner scanner = new Scanner(countriesLatLong);
        //Διαπερνάει όλες τις γραμμές
        while (scanner.hasNextLine()) {
            //Κάνουμε τα ονόματα σε lower case για να μην
            //υπάρχει τυχόν διαφορά
            String line = scanner.nextLine().toLowerCase();
            String name = country.getName().toLowerCase();
            //αρχικοποίηση των μεταβλητών
            double lat = 0;
            double longl = 0;
            //Εάν η γραμμή έχει το όνομα της χώρας
```

```

        if(line.contains(name)) {
            //Διαβάζει το lat
            lat = Double.parseDouble(
                line.substring(line.indexOf(
                    "lat"
                )+3, line.indexOf("long")
            ));
            //Διαβάζει το long
            long1 = Double.parseDouble(
                line.substring(line.indexOf(
                    "long")+4, line.indexOf("eol")
            ));
            //Τα θέτει στην χώρα
            country.setLat(lat);
            country.setLong1(long1);
        }
    }
    scanner.close();
}

//firePropertyChange event
AppSystem.getPropChangeFirer().firePropertyChange("server
conection", "", "Η σύνδεση με τον σέρβερ ολοκληρώθηκε, εισαγωγή
χωρών στη βάση δεδομένων");
}
catch (IOException e){
    //Σε περίπτωση IOException, δίνω μήνυμα να γίνει ξανά η προσπάθεια
    AppSystem.getPropChangeFirer().firePropertyChange("server
connection", "", "<html><span style=\"color:red\">Η σύνδεση με τον
σέρβερ απέτυχε, παρακαλώ δοκιμάστε ξανά</span></html>");
    countries = null; //Επιστρέφει null για να μην εκτελεστεί η μέθοδος
    του DbManager
}
catch (Exception e){
    //Σε άλλο Exception τερματίζω το πρόγραμμα
    AppSystem.terminate("<html>Συνέβη ένα αναπάντεχο σφάλμα η εφαρμογή
θα τερματιστεί</html>");
}
return countries;
}
}

```

```

//Κλάση type adapter για την ανάγνωση json elements
class CountryTypeAdapter extends TypeAdapter<Country> {
    @Override//Για την ανάγνωση
    public Country read(JsonReader in) throws IOException {
        final Country country = new Country();
        in.beginObject(); //Για κάθε στοιχείο στο Json String που είναι μέσα σε
        {}, θα εκτελεστεί η παρακάτω ανάγνωση
        List<Coviddata> datas = new ArrayList<>();
        while (in.hasNext()) { //θα εκτελεστεί όσο υπάρχουν ακόμη πεδία

```

```
//η nextName() επιστρέφει το όνομα του πεδίου
String fieldName = in.nextName();
try{//Όταν το πεδίο είναι το Province/State
    if ("Province/State".equals(fieldName)) {
        //Αγνοώ το state το βάζω
        in.skipValue();
    }//Για το πεδίο Country/Region
    else if ("Country/Region".equals(fieldName)) {
        //Θέτει σαν όνομα χώρας το όνομα
        country.setName(in.nextString());
    }//Για το πεδίο Lat, απλά το περνάει στο πεδίο της χώρας
    else if ("Lat".equals(fieldName)) {
        country.setLat(in.nextDouble());
    }//Για το πεδίο Long, απλά το περνάει στο πεδίο της χώρας
    else if ("Long".equals(fieldName)) {
        country.setLong1(in.nextDouble());
    }
    //Κάθε άλλο πεδίο στο Json String αναφέρεται σε μια ημερομηνία
    else {
        // Παίρνει από το όνομα του πεδίου την ημερομηνία
        //Βρες τον δείκτη για το πρώτο /
        int firstindex = fieldName.indexOf("/");
        //Βρες τον δείκτη για το δεύτερο /
        int lastindex = fieldName.lastIndexOf("/");
        //Ο μήνας είναι από την αρχή μέχρι το 1ο /
        int month =
            Integer.parseInt(fieldName.substring(0, firstindex));
        //Η ημέρα ανάμεσα στα 2 /
        int day =
            Integer.parseInt(fieldName.substring(
                firstindex+1, lastindex));
        //Το έτος από το 2 / μέχρι το τέλος
        int year = Integer.parseInt(
            fieldName.substring(lastindex+1))+2000;
        //Μεταβλητή Date που είναι συμβατή με την βάση δεδομένων
        Date date =
            java.sql.Date.valueOf(LocalDate.of(year, month, day));
        //Δημιουργώ ένα αντικείμενο Coviddata
        Coviddata data =
            new Coviddata(country, date, in.nextInt());
        //Το προσθέτω στη λίστα που θα μπει στο αντικείμενο χώρας.
        datas.add(data);
    }
}
//Αν το αρχείο έχει κάποια μη συμβατή τιμή αγνόησέ την
catch (IllegalArgumentException e){
    in.skipValue();
}
//Σε κάθε άλλο σφάλμα το πετάει στην μέθοδο reciveData() της
ServerCnnection class
catch (Exception e){
    throw e;
}
}
in.endObject();//Συνάντησε }
```

```
//Δίνουμε στην χώρα τη λίστα με τα coviddata
country.setCoviddataList(datas);

return country;//επιστρέφει τη χώρα
}
//Για την εγγραφή σε Json String, δεν χρειάζεται στα πλαίσια της εργασίας
@Override
public void write(JsonWriter writer, Country t) throws IOException {

}

}
```

**Πακέτο coviddataapp**, περιέχει τις κλάσεις **CovidDataApp**, **AppSystem** και **CovidDataType**. Η κλάση CovidDataApp είναι η κλάση με την μέθοδο main, που απλά δημιουργεί ένα αντικείμενο AppSystem ώστε να εκκινήσει η εφαρμογή. Η κλάση AppSystem είναι ο πυρήνας τους συστήματος, αρχικοποιεί το βασικό παράθυρο, αρχικοποιεί τον HtmlBrowser, που είναι μια jfx εφαρμογή μέσα σε ένα jfx panel και τον DbManager, ώστε να γίνει η σύνδεση με την βάση δεδομένων. Η κλάση CovidDataType, είναι enumeration κλάση σχεδιασμένη για τις ανάγκες της εφαρμογής.

```
package coviddataapp;

import boundaryclasses.DbManager;
import java.awt.Frame;
import javafx.application.Platform;
import javax.swing.JOptionPane;
import javax.swing.event.SwingPropertyChangeSupport;
import presenters.OvPresenter;
import presenters.Presenter;
import views.HtmlBrowser;
import views.MainFrame;

public class AppSystem {

    private final MainFrame mainFrame;
    private static HtmlBrowser browser;
    private static SwingPropertyChangeSupport propChangeFirer;
    private Presenter presenter;
    private DbManager dbManager;

    public AppSystem() {
        //Αρχικοποίηση του mainFrame
        mainFrame = new MainFrame("Covid Stats Application");
        //Αρχικοποίηση του jfx browser
        //Αν κλείσει ο browser να μην τερματιστεί το thread του jfx app
        Platform.setImplicitExit(false);
        browser = new HtmlBrowser();
        //Θέτει το σύστημα σαν propertyChangeFirer
        propChangeFirer = new SwingPropertyChangeSupport(this);
        //Με την κατασκευή του συστήματος θέτει στον προεπιλεγμένο Presenter
        presenter = new OvPresenter(mainFrame.getPanel(), this);
        // Αρχικοποίηση του συστήματος
        initAppSystem();
    }
}
```

```
}

private void initAppSystem() {
    //Ορίζω Listener για PropertyChanged events
    setPropertyChangedListener(presenter);
    //αρχικοποιώ το DbManager
    dbManager = new DbManager();
}

//Θέτει τον Presenter που θα ακούει τον propertyChangeFirer
public void setPropertyChangedListener(Presenter presenter) {
    propChangeFirer.addPropertyChangedListener(presenter);
}

//static μέθοδος για να μπορώ να πυροδοτώ συμβάντα
public static SwingPropertyChangeSupport getPropChangeFirer() {
    return propChangeFirer;
}

public MainFrame getMainFrame() {
    return mainFrame;
}

public static HtmlBrowser getBrowser() {
    return browser;
}

//Θέτει τον Presenter του τρέχοντος View
public void setPresenter(Presenter presenter, Presenter toRemove) {
    if (toRemove != null)
        //Αφαιρεί τον προηγούμενο Presenter από τη λίστα με τους
        //PropertyChangeListeners
        propChangeFirer.removePropertyChangedListener(toRemove);
    this.presenter = presenter;
    setPropertyChangedListener(presenter);
}

}

public Presenter getPresenter() {
    return presenter;
}

}

public DbManager getDbManager() {
    return dbManager;
}

//Τερματίζει την εφαρμογή, οι εργασίες της main, έχουν τελειώσει,
//άρα αν κάνουμε dispose το παράθυρο θα κλείσει η εφαρμογή
public static void terminate(String message){
    //Πάρε όλα τα frame
    Frame[] frames = Frame.getFrames();
    //Εάν ο χρήστης έχει κλείσει την εφαρμογή με το x
    //και το σύστημα προσπαθεί να κάνει αρχικοποίηση του entity manager,
    //χωρίς συνδεδεμένη την βάση δεδομένων, να μην εμφανιστεί το μήνυμα
    if(frames[0].isVisible()) {
        JOptionPane.showMessageDialog(frames[0], message,
            "Απροσδόκητο σφάλμα", JOptionPane.ERROR_MESSAGE);
    }
    //Διαπέρασέ τα
```



```
        for(Frame f: frames)
            //κλείσε τα
            f.dispose();
        //Κλείσε το jfx app
        Platform.exit();
    }
}
```

```
/*
 * https://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html#program-
 * matic Look and feel
 */
package coviddataapp;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.UIManager;
import javax.swing.UIManager.LookAndFeelInfo;
import javax.swing.UnsupportedLookAndFeelException;

/**
 *
 *
 *
 *
 */
public class CovidDataApp {

    public static void main(String[] args) {
        try {//Θέτει lookAndFell στην εφαρμογή
            LookAndFeelInfo[] lookAndFeells = UIManager.get-
InstalledLookAndFeels();
            UIManager.setLookAndFeel(lookAndFeells[1].getClassName());
        } catch (ClassNotFoundException | InstantiationException |
IllegalAccessException | UnsupportedLookAndFeelException ex)
        {
            Logger.getLogger(CovidDataApp.class.getName())
                .log(Level.SEVERE, null, ex);
        }
        //Αλλαγή text των κουμπιών στα optionpane
        UIManager.put("OptionPane.yesButtonText", "Ναι");
        UIManager.put("OptionPane.noButtonText", "Όχι");
        new AppSystem();//Εκκινεί το σύστημα
    }
}
```

```
package coviddataapp;
//Enum κλάση με πεδία
public enum CovidDataType {
    DEATHS("deaths", 1, "Θάνατοι"),
    RECOVERED("recovered", 2, "Ασθενείς που έχουν ανακάμψει"),
    CONFIRMED("confirmed", 3, "Επιβεβαιωμένα κρούσματα");

    private final String string;
    private final int value;
    private final String labelName;

    //ο constructor θέτει τα πεδία των enum που ορίσαμε πιο πάνω
    private CovidDataType(String string, int value, String labelName) {
        this.string = string;
        this.value = value;
        this.labelName = labelName;
    }
    //Επιστρέφει ένα CovidDataType enum ανάλογα το labelName
    public static CovidDataType getByLabelName (String labelName) {
        CovidDataType type = null;
        for(CovidDataType t: CovidDataType.values()){
            if(t.getLabelName() == labelName) {
                type = t;
                break;
            }
        }
        return type;
    }
    //Επιστρέφει ένα CovidDataType enum ανάλογα το labelName
    public static String getNameByValue (int value) {
        CovidDataType type = null;
        for(CovidDataType t: CovidDataType.values()){
            if(t.value == value) {
                type = t;
                break;
            }
        }
        return type.labelName;
    }
    public String getString() {
        return string;
    }
    public int getValue() {
        return value;
    }
    public String getLabelName() {
        return labelName;
    }
}
```

(γ) Εισάγετε τον κώδικα java για τις rojo κλάσεις. Θα πρέπει να εισάγεται ΜΟΝΟ τον κώδικα που έχετε γράψει εσείς και όχι αυτόν που παράγεται αυτόματα προσδιορίζοντας ευκρινώς το όνομα της κλάσης και της μεθόδου. Φροντίστε ο κώδικας να είναι μορφοποιημένος κατάλληλα και να είναι ευανάγνωστος. Θα πρέπει να υπάρχει τεκμηρίωση με μορφή σχολίων. Εάν δεν έχετε δώσει απάντηση γράψτε με κεφαλαία γράμματα, ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ. Εάν εν γνώση σας δίνετε ελλιπή απάντηση γράψτε με κεφαλαία γράμματα, ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.

Οι rojo κλάσεις έχουν δημιουργηθεί από το netbeans με την επιλογή δημιουργίας entity class from database. Παρακάτω θα παραθέσουμε μόνο τον κώδικα που προσθέσαμε εμείς για τις ανάγκες του συστήματος.

### Κλάση Country

Έχουμε εισάγει 2 πρόσθετα namedqueries και έχουμε αλλάξει τις μεθόδους equals, compareTo και toString όπως φαίνεται παρακάτω:

```
//για διαγραφή
, @NamedQuery(name = "Country.deleteByName", query = "DELETE FROM Country c WHERE
c.name = :name")
//για διαγραφή όλων
, @NamedQuery(name = "Country.deleteAll", query = "DELETE FROM Country c")

@Override
public boolean equals(Object object) {

    if (!(object instanceof Country)) {
        return false;
    }
    Country other = (Country) object;
    //Όταν έχουν διαφορετικό όνομα χώρας δεν είναι equals
    if ((this.name == null && other.name != null) ||
        (this.name != null && !this.name.equals(other.name))) {
        return false;
    }
    return true;
}

@Override
//Συγκρίνει τα αντικείμενα με βάση το όνομα
public int compareTo(Country other) {
    return this.getName().compareTo(other.getName());
}
//Ωστε να εμφανίζεται το όνομα στα JComboBox
@Override
public String toString() {
    return name;
}
```

**Κλάση Coviddata**, εδώ έχουμε δημιουργήσει έναν νέο constructor, που τον χρησιμοποιούμε κατά την ανάγνωση του Json string, έχουμε εισάγει πάλι κάποια namedqueries και έχουμε αλλάξει τις μεθόδους equals και compareTo:

```
//Τον δημιουργήσαμε και τον χρησιμοποιούμε κατά την ανάγνωση του Json string
public Coviddata(Country country, Date trndate, int proodqty) {
    this.country = country;
    this.trndate = trndate;
    this.proodqty = proodqty;
}

//Βρίσκει δεδομένα με το όνομα της χώρας, ανάμεσα σε 2 ημερομηνίες
, @NamedQuery(name = "Coviddata.findBYCountryDataBetweenDates",
    query = "SELECT c FROM Coviddata c WHERE c.country.name = :coutryname
        and c.datakind = :datakind
        and c.trndate >= :fromDate and c.trndate <= :toDate")
//Βρίσκει δεδομένα με το όνομα της χώρας
, @NamedQuery(name = "Coviddata.findByCountry",
    query = "SELECT c FROM Coviddata c WHERE c.country.name = :coutryname")
//Βρίσκει τα δεδομένα με βάση τον τύπο
, @NamedQuery(name = "Coviddata.findByCountryNdata",
    query = "SELECT c FROM Coviddata c WHERE c.country.name = :coutryname
        and c.datakind = :datakind")
//διαγραφή με την χώρα και τον τύπο
, @NamedQuery(name = "Coviddata.deleteByCountryNdata",
    query = "DELETE FROM Coviddata c WHERE c.country.name = :coutryname
        and c.datakind = :datakind")
//διαγραφή με την χώρα
, @NamedQuery(name = "Coviddata.deleteByCountry",
    query = "DELETE FROM Coviddata c WHERE c.country.name = :coutryname")
//διαγραφή όλων
, @NamedQuery(name = "Coviddata.deleteAll",
    query = "DELETE FROM Coviddata c")

@Override
public boolean equals(Object object) {

    if (!(object instanceof Coviddata)) {
        return false;
    }
    Coviddata other = (Coviddata) object;
    //Είναι ίδια αντικείμενα αν αναφέρονται στην ίδια χώρα και στην ίδια
    ημερομηνία και στο ίδιο είδος
    if ((this.country.equals(other.country)) && (datakind==other.datakind) &&
(trndate.equals(other.trndate))) {
        return true;
    }
    return false;
}

@Override
public int compareTo(Coviddata other) {
    if (country.equals(other.country))//Αν αναφέρονται στην ίδια χώρα
        //σύγκριση με βάση την ημερομηνία
        if (trndate.equals(other.trndate))
            //αν είναι ίδια η ημερομηνία σύγκρινε με βάση το datakind
            return datakind-other.datakind;
```

```
        else
            return trndate.compareTo(other.trndate);
        //Αλλιώς σύγκριση με βάση το όνομα χώρας
        return country.getName().compareTo(other.country.getName());
    }
```

## Ερώτημα Β – Δημιουργία GUI Εφαρμογής

Εδώ κάπου θα πρέπει να αναφερθεί ότι έχουμε χρησιμοποιήσει το μοντέλο MVP (Model, View, Presenter), που είναι παραλλαγή του μοντέλου MVC, με την διαφορά ότι το Model και το View επικοινωνούν μόνο μέσω του Controller και ποτέ απευθείας μεταξύ τους. Με βάση το μοντέλο αυτό, στα Views ο κώδικας που βάλουμε είναι ελάχιστος, έχουμε προσθέσει για όλα τα components που χρειαζόταν κάποιος χειρισμός getters και η διαχείριση τους γίνεται στην κλάση του εκάστοτε Presenter. Οπότε για κάθε view, εκτός την παρουσίαση του κώδικα που βάλουμε εμείς, θα παρουσιάζεται και ο κώδικας τους presenter που αντιστοιχεί στο view αυτό.

Έχει δημιουργηθεί ένα interface Presenter, όπου έχει τις 2 βασικές μεθόδους, που πρέπει να υλοποιούν όλοι οι Presenters, και για κάθε View, έχουμε 1 κλάση Presenter, που υλοποιεί το interface αυτό.

Επίσης να αναφέρουμε ότι η εφαρμογή είναι εφαρμογή ενός παραθύρου, όπου κάθε φορά μέσα στο βασικό παράθυρο προβάλλεται το view που έχει επιλέξει ο χρήστης. Για να υλοποιηθεί αυτό, έχουμε δημιουργήσει μια κλάση mainframe, με τις ανάλογες μεθόδους, για την προβολή του view που επιλέγει ο χρήστης.

### Interface Presenter

```
package presenters;

import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

/**
 *
 */
public interface Presenter extends PropertyChangeListener {

    public void setUpViewEvents();
    @Override
    public void propertyChange(PropertyChangeEvent evt);

}
```

### Κλάση mainFrame

```
package views;

import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javafx.application.Platform;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class MainFrame extends JFrame{

    private final Container container = this.getContentPane();
    private JPanel panel;
    private final FlowLayout layout = new FlowLayout();
```

```
public MainFrame(String title) {
    super(title);
    initMainFrame();
}
//Αρχικοποίηση του Frame
private void initMainFrame() {
    container.setLayout(layout);
    //Αλλαγή εικονιδίου στο παράθυρο
    setIconImage(
        new ImageIcon(getClass()
            .getResource("/img/covidIcon.png"))
            .getImage());
    //Listener για τερματισμό της εφαρμογής
    addWindowListener(new WindowAdapter () {
        @Override
        //Οι εργασίες της main, έχουν τελειώσει,
        //άρα αν κάνουμε dispose το παράθυρο θα κλείσει η εφαρμογή
        public void windowClosing(WindowEvent event){
            //Πάρε όλα τα frame
            Frame[] frames = Frame.getFrames();
            //Διαπέρασέ τα
            for(Frame f: frames)
                //κλείσε τα
                f.dispose();
            //Κλείσε το jfx app thread
            Platform.exit();
        }
    });
    setResizable(false);
    //Προεπιλεγμένο View κατά την εκκίνηση
    showOptionsView();
    setVisible(true);
}
//Δείχνει το View με τα Options
public void showOptionsView() {
    container.removeAll(); //Αφαιρεί από το ContentPane πιθανό περιεχόμενο
    container.repaint(); //Ζωγραφίζει ξανά το ContentPane
    panel = new OptionsView(); //Δημιουργεί το View
    container.add(panel); //Το τοποθετεί στο ContentPane
    container.repaint(); //Ζωγραφίζει ξανά το ContentPane
    container.validate(); //Κάνει valid τα συστατικά του ContentPane
    pack(); //Θέτει το μέγεθος του παραθύρου με βάση τα περιεχόμενά του
}
//Τις ενέργειες για το DataManagerView
public void showDataManagerView() {
    container.removeAll();
    panel = new DataManagerView();
    container.add(panel);
    container.repaint();
    container.validate();
    pack();
}
//Τις ενέργειες για το DataMapView
public void showDataMapView() {
```



```

        container.removeAll();
        panel = new DataMapView();
        container.add(panel);
        container.repaint();
        container.validate();
        pack();
    }
    //Τις ενέργειες για το CountryDataView
    public void showCountryDataView() {
        container.removeAll();
        panel = new CountryDataView();
        container.add(panel);
        container.repaint();
        container.validate();
        pack();
    }
    //Επιστρέφει το View για διαχείριση
    public JPanel getPanel() {
        return panel;
    }
}

```

**Κλάση OptionsView**, εδώ έχουμε μόνο κουμπιά και labels, δεν έχει κάποια λειτουργία, το μόνο που έχουμε εισάγει σαν κώδικα, για την παρουσίαση των labels και των κουμπιών από το properties το κείμενο, το οποίο έχει διαμορφωθεί με html. Θα παρουσιάσουμε ενδεικτικά μόνο για αυτό το View τον προαναφερόμενο κώδικα, λόγω του ότι δεν έχει κάτι άλλο να παρουσιάσουμε. Εδώ να υπενθυμίσουμε ότι για τα components που θέλουμε να ελέγξουμε για κάθε view, με δεξί κλικ και την επιλογή inset code, έχουμε εισάγει τους κατάλληλους getters, όπου για λόγους συντομίας δεν θα τους παρουσιάσουμε εδώ.

Για το dataButton

```

<html>
    <p style="text-align:center">Διαχείριση δεδομένων</p>
    <p style="text-align:center">Covid19</p>
</html>

```

Για το label δίπλα στο dataButton

```

<html>
    <h3>Επιλογές διαχείρισης βάσης δεδομένων</h3>
    <p>-Εισαγωγή δεδομένων από το ίντερνετ</p>
    <p>-Προβολή των αποθηκευμένων δεδομένων στην βάση δεδομένων,</p>
    <p>-Διαγραφή δεδομένων</p>
</html>

```

Για το countryButton

```

<html>Προβολή δεδομένων Covid19 ανά χώρα</html>

```

Για το label δίπλα στο countryButton

```

<html>
    <h3>Επιλογές προβολής δεδομένων</h3>
    <p>-Προβολή δεδομένων ανά χώρα σε πίνακα</p>
    <p>-Προβολή δεδομένων ανά χώρα σε γράφημα</p>
    <p>-Προβολή δεδομένων ανά χώρα σε χάρτη</p>
    <p>-Διαγραφή δεδομένων</p>
</html>

```

Για το mapButton

```
<html>Προβολή δεδομένων Covid19 σε χάρτη</html>
```

Για το label δίπλα στο mapButton

```
<html>
    <h3>Επιλογές προβολής δεδομένων σε χάρτη</h3>
    <p>-Προβολή δεδομένων ανά χώρα σε χάρτη</p>
    <p>-Δυνατότητα πολλαπλής επιλογής χωρών και προβολή σε χάρτη</p>
</html>
```

### Κλάση OvPresenter,

είναι ο Presenter του OptionsView, για κάθε κουμπί, θέτει actionListener που καλεί τις ανάλογες μεθόδους του συστήματος και του mainframe, ώστε να θέτει τον τρέχον Presenter του συστήματος και να δείχνει το View που επιλέγει ο χρήστης αντίστοιχα.

```
package presenters;

import boundaryclasses.DbManager;
import coviddataapp.AppSystem;
import java.awt.Frame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.beans.PropertyChangeEvent;
import javafx.application.Platform;
import javax.swing.JButton;
import javax.swing.JPanel;
import views.OptionsView;

/**
 *
 *
 */
public class OvPresenter implements Presenter{//OptionsView Presenter

    private final OptionsView view;
    private final AppSystem appSystem;
    private JButton[] buttons;

    public OvPresenter(final JPanel view, final AppSystem appSystem) {
        //Κάνει cast το JPanel στην κλάση του View
        this.view = (OptionsView) view;
        //Θέτει το σύστημα για να μπορεί να κάνει ενέργειες όταν πατάει ο
        //χρήστης κουμπιά
        this.appSystem = appSystem;
        setUpViewEvents();
    }

    @Override
    public final void setUpViewEvents() {
        buttons = new JButton[]{
            view.getDataButton(), //0 = showDataView
            view.getCountryButton(), //1 = showCountryDataView
            view.getMapButton(), //2 = showDataMapView
            view.getExitButton() // 3 = System.exit(0)
        };
    }
}
```

```
};
if(DbManager.getEm() ==null)
    disableAllButtons();
else {
    view.getInfoLabel().setVisible(false);
    view.getIconLabel().setVisible(false);
}
//Κουμπί DataManager
buttons[0].addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //Καλεί τον mainFrame να αλλάξει View
        appSystem.getMainFrame().showDataMapView();
        //Δημιουργεί τον Presenter για το νέο View
        Presenter p =
        new DataVPresenter( appSystem.getMainFrame().getPanel(),
        appSystem);
        //Στέλνει στο σύστημα τον νέο Presenter και τον Παλίο για ε-
νέργειες
        appSystem.setPresenter(p, appSystem.getPresenter());
    }
});
//Τίδεις λειτουργίες για το CountryButton
buttons[1].addActionListener((ActionEvent e) -> {
    appSystem.getMainFrame().showCountryDataView();
    Presenter p =
    new CdvPresenter( appSystem.getMainFrame().getPanel(),
    appSystem);
    appSystem.setPresenter(p, appSystem.getPresenter());
});
//Τίδεις λειτουργίες για το MapButton
buttons[2].addActionListener((ActionEvent e) -> {
    appSystem.getMainFrame().showDataMapView();
    Presenter p =
    new DmvPresenter( appSystem.getMainFrame().getPanel(),
    appSystem);
    appSystem.setPresenter(p, appSystem.getPresenter());
});
//Κλείσιμο της εφαρμογής
buttons[3].addActionListener((ActionEvent e) -> {
    //Πάρε όλα τα frame
    Frame[] frames = Frame.getFrames();
    //Διαπέρασέ τα
    for(Frame f: frames)
        //κλείσε τα
        f.dispose();
    //Κλείσε το jfx app thread
    Platform.exit();
});
}

@Override
```

```
//Διαχειρίζεται το GUI σε σχέση με τις αλλαγές στο σύστημα
public void propertyChange(PropertyChangeEvent evt) {
    //Το όνομα που ορίσαμε στο PropertyChangeEvent
    String propName = evt.getPropertyName();
    //Η νέα τιμή που περάσαμε
    Object newVal = evt.getNewValue();
    //Η παλιά τιμή που περάσαμε
    Object oldVal = evt.getOldValue();

    if("Db login".equalsIgnoreCase(propName)){
        view.getInfoLabel().setText(String.valueOf(newVal));
        view.getIconLabel().setVisible(false);
        view.repaint();
        if(!String.valueOf(newVal).equalsIgnoreCase(""))
            enableAllButtons();
    }

    private void disableAllButtons() {
        for(JButton button: buttons)
            button.setEnabled(false);
    }

    private void enableAllButtons() {
        for(JButton button: buttons)
            button.setEnabled(true);
    }
}
```

### Κλάση DataManagerView

Με την επιλογή customize code, έχουμε αλλάξει τον κώδικα σε κάποια components όπως παρακάτω:

#### EntityManager

Ο Entity manager που έχουμε χρησιμοποιήσει είναι μοναδικός για όλο το σύστημα και έχουμε εισάγει static μέθοδο στην DbManager κλάση για να τον κάνουμε get

```
em = DbManager.getEm();
```

Καθαρίζουμε τον em, στην έναρξη κάθε view, γιατί είχε παρατηρηθεί, λόγω κάποιων πράξεων πάνω στα αντικείμενα του model, ότι το entity manager persistence context, ππολλέςφορές διέφερε από τα δεδομένα που είναι αποθηκευμένα στην βάση δεδομένων. Στην επόμενη εντολή έχει επιλεγεί post-create, ώστε να εκτελείται μετά την δημιουργία του entityManager.

```
em.clear();
```

#### serverList

Είναι η λίστα που έχει συνδεθεί με το JList που δείχνει τα αποτελέσματα αναζήτησης από το ιντερνετ, έχουμε αλλάξει τον κώδικα, να δημιουργεί μια κενή λίστα, και να μην παίρνει αποτελέσματα από κάποιο query

```
serverList = java.beans.Beans.isDesignTime() ?
java.util.Collections.emptyList() :
org.jdesktop.observablecollections.ObservableCollections.observableList(
    new ArrayList<>()
);
```

#### coviddataList

Η λίστα αυτή είναι συνδεδεμένη με το JTable που δείχνει τι έχουμε αποθηκεύσει στην βάση δεδομένων, όμως, επειδή κατά την πρώτη εκτέλεση της εφαρμογής είναι κενή η βάση δεδομένων, η λίστα έχει αρχικοποιηθεί με τον ίδιο τρόπο, σαν κενή και όπως θα δούμε παρακάτω, ο Presenter, όταν αρχικοποιείται την γεμίζει ανάλογα τα περιεχόμενα της βάσης δεδομένων.

```
coviddataList = java.beans.Beans.isDesignTime() ?  
    java.util.Collections.emptyList() :  
    org.jdesktop.observablecollections.ObservableCollections.observableList(  
        new ArrayList<>()  
    );
```

Εδώ να αναφέρουμε ότι το loadingLabel με επιλογή after-all-set από το customize code έχει γίνει μη ορατό και μετά την ορατότητά του την διαχειρίζεται το presenter, ενώ για την ορατότητα των delete buttons, έχει επιλεγεί να δρα αποκλειστικά ο presenter, γιατί δεν μπορούμε να γνωρίζουμε από πριν, αν πρέπει να εμφανίζονται.

Τα υπόλοιπα components, έχουν γίνει με την χρήση του design view και ο κώδικας έχει παραχθεί αυτόματα.

**Κλάση DataVPresenter**, presenter υπεύθυνος τις λειτουργίες του **DataManageView**, θέτει όλα τα actionListeners και λαμβάνει propertyChange Events από το σύστημα, ώστε να ανανεώνει το VIEW.

```
package presenters;  
  
import boundaryclasses.DbManager;  
import boundaryclasses.ServerConnection;  
import coviddataapp.AppSystem;  
import coviddataapp.CovidDataType;  
import java.awt.Color;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.beans.PropertyChangeEvent;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
import java.util.TreeMap;  
import javax.swing.JButton;  
import javax.swing.JLabel;  
import javax.swing.JOptionPane;  
import javax.swing.JPanel;  
import javax.swing.SwingWorker;  
import javax.swing.event.ListSelectionEvent;  
import model.Country;  
import model.Coviddata;  
import views.DataManageView;  
  
/**  
 *  
 */  
public class DataVPresenter implements Presenter {  
    //DataManageView Presenter  
  
    private final DataManageView view;  
    private final AppSystem appSystem;  
    private TreeMap<String, List<Coviddata>> tempData = new TreeMap<>();  
    private JButton[] buttons;
```

```
private final DbManager dBm;
private boolean insertingData = false;

public DataVPresenter(JPanel view, AppSystem appSystem) {
    this.dBm = appSystem.getDbManager();
    //Κάνει cast το JPanel στην κλάση του View
    this.view = (DataManageView) view;
    //Θέτει το σύστημα για να μπορεί να κάνει ενέργειες όταν πατάει ο
    χρήστης κουμπιά
    this.appSystem = appSystem;
    setUpViewEvents();
}

@Override
public final void setUpViewEvents() { //Αρχικοποίηση Presenter και των
Event
    //Με την αρχικοποίηση του View βάζω στην λίστα του πίνακα Coviddata
    //μια συνοπτική παρουσίαση, για κάθε χώρα, τι είδος δεδομένα έχουμε
    αποθηκεύσει
    //στην βάση δεδομένων και μέχρι ποια ημερομηνία
    List<Coviddata> coviddataList = view.getCoviddataList();
    //Καλεί την μέθοδο που βρίσκει την τελευταία ημερομηνία για κάθε χώρα
    και είδος
    coviddataList.addAll(dBm.findLastData());
    //Εάν η λίστα δεν είναι άδεια
    if(!coviddataList.isEmpty())
        //Ταξινομώ την
        coviddataList.sort((Coviddata c1, Coviddata c2) -> {
            //Πρώτα ανά χώρα και μετά ανά dataKind
            if(c1.getCountry().equals(c2.getCountry()))
                return c1.getDataKind() - c2.getDataKind();
            else
                return c1.getCountry().compareTo(c2.getCountry());
        });

    buttons = new JButton[]{
        view.getBackToOptionsButton(), //0 = BackToOptionsButton
        view.getInsertCountryButton(), //1 = InsertCountryButton
        view.getInsertDataButton(), // 2 = InsertDataButton
        view.getDeleteCountryButton(), //3 = getDeleteCountryButton
        view.getDeleteDataButton(), // 4 = getDeleteDataButton
        view.getDeleteAll() // 5 = getDeleteAll()
    };
    //Με την εκκίνηση του GUI φτιάχνω το visibility των deleteButtons
    deleteButtonsVisibility();

    //Κουμπί deleteAll
    //Ανώνυμος ActionListener
    buttons[5].addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            JLabel label = view.getInfoText();
            //Ρωτάμε τον χρήστη αν θέλει να προχωρήσει
            int x =
                JOptionPane.showConfirmDialog (view,
```

```

        "Είστε σίγουροι ότι θέλετε να διαγράψετε όλα τα δεδομένα;",
        "Προσοχή",
        JOptionPane.YES_NO_OPTION);
    if(x==0) {
        disableAllButtons(); //Λόγω νήματος απενεργοποιούμε όλα τα
κουμπιά
        //Χρησιμοποιεί ένα νήμα διαθέσιμο από τα 10 που έχει το
SwingWorker Threadpool
        new SwingWorker<Void, Void> (){//ορίζω το νήμα
            List<Country> obsCountryList =
            view.getCountryList();
            List<Coviddata> obsCovidatalist =
            view.getCoviddataList();
            List<Country> serverList = view.getServerList();
            @Override
            //λειτουργίες που εκτελούνται στο SwingWorker thread
            protected Void doInBackground() {
                //Διαγραφή δεδομένων
                dBm.deleteAll();
                return null;
            }
            @Override
            //Λειτουργίες που εκτελούνται στο event dispatch
thread μόλις ολοκληρωθεί το SwingWorker thread
            public void done() {
                //Ενημέρωση των observable list του GUI
                //Ενημέρωση του πίνακα με τις χώρες
                obsCountryList.clear();
                obsCovidatalist.clear();
                serverList.clear();
                //εμφανίζει μήνυμα στο view
                label.setText("η διαγραφή δεδομένων,
                            ολοκληρώθηκε με επιτυχία");
                //επαναφέρω τα κουμπιά
                enableAllButtons();
                deleteButtonsVisibility();
                //Μήνυμα προς τον χρήστη
                view.getDataLabel().setText(
                "<html><span style=\"color:red\">
                Δεν υπάρχουν δεδομένα για εισαγωγή,
                παρακαλώ πατήστε λήψη δεδομένων</span><html>");
                //Απόκρυψη του loading label
                view.getLoadingLabel().setVisible(false);
            }
        }.execute();
        //Εκτελείται παράλληλα με το SwingWorker thread
        //εμφανίζει μήνυμα στο view
        label.setText(
        "Διαγραφή δεδομένων, παρακαλώ περιμένετε...");
        //εμφάνιση του loading label
        view.getLoadingLabel().setVisible(true);
    }
}
});

```



```
//κουμπί deleteCountry
buttons[3].addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JLabel label = view.getInfoText();
        //παίρνω τους δίκτες που έχουν επιλεγεί
        int[] selected =
            view.getCountryJList().getSelectedIndices();
        //Εκτελώ ενέργειες μόνο αν υπάρχει επιλογή
        if( selected.length!=0 ) {
            //Ρωτάω τον χρήστη αν επιθυμεί να προχωρήσει
            int x =
                JOptionPane.showConfirmDialog (view,
                    "<html>Είστε σίγουροι ότι θέλετε να διαγράψετε τις
                    επιλεγμένες χώρες;<br>"
                    "<span style=\"color:red\">Σημείωση: Κατά την διαγραφή
                    μιας χώρας, θα διαγραφούν και Covid data αν
                    υπάρχουν.</span></html>",
                    "Προσοχή",
                    JOptionPane.YES_NO_OPTION);
            if(x==0) {
                ();
                //Λόγω νήματος απενεργοποιούμε όλα τα κουμπιά
                disableAllButtons
                //ορίζω το νήμα
                new SwingWorker<Void, ArrayList<Object>> () {
                    List<Country> obsCountryList =
                        view.getCountryList();
                    List<Coviddata> obsCoviddataList =
                        view.getCoviddataList();
                    List<Country> countriesToRemove =
                        new ArrayList<>();
                    List<Coviddata> coviddataToRemove =
                        new ArrayList<>();

                    @Override
                    //λειτουργείες στο background ώστε τα γραφικά να μην κολάνε.

                    protected Void doInBackground() throws
                    InterruptedException {
                        //Τοποθετώ σε άλλη λίστα για μην έχω πρόβλημα
                        με τους δείκτες κατά την αφαίρεση αντικειμένων από τα observable lists
                        countriesToRemove.addAll(obsCountryList);
                        coviddataToRemove.addAll(obsCoviddataList);
                        ArrayList<Object> toPublish =
                            new ArrayList<>();
                        for(int i: selected){
                            //Αφαιρώ πρώτα τα δεδομένα coviddata που
                            ενδεχομένως υπάρχουν, λόγω foreign key
                            dBm.deleteCoviddataByCountry(
                                countriesToRemove.get(i)
                            );
                            //Βρίσκω το data που υπάρχει στο JTable
                            for(Coviddata d: coviddataToRemove)
                                if(d.getCountry().equals(
```

```

        countriesToRemove.get(i))) {
            //Το προσθέτω στη λίστα που θα γίνει publish
            toPublish.add(d);
        }
        //Αφαιρώ την χώρα από το database
        dbName.deleteCountry(
            countriesToRemove.get(i));
        //Την προσθέτω στην λίστα που θα γίνει publish
        toPublish.add(countriesToRemove.get(i));
        publish(toPublish);
//δίνω λίγο χρόνο για να προλάβει να εκτελεστεί το publish πριν την επόμενη
κλήση
        Thread.sleep(10);
    }
    return null;
}
@Override
//Εκτελείται με την κλήση της publish στο event dispatch thread
//Εδώ σε συγχρονισμό με το worker Thread επηρεάζω αντικείμενα του GUI
protected void process(
    List<ArrayList<Object>> chunks) {
    ArrayList<Object> mostRecentValue =
        new ArrayList<>();
    //Σε κάθε κλήση παίρνω το τελευταίο στοιχείο που στάλθηκε
    mostRecentValue.addAll(
        chunks.get(chunks.size()-1)
    );
    //Αφαίρεση από τις λίστες και κατ επέκταση από τα JTables
    for(Object o: mostRecentValue) {
        if(o instanceof Coviddata)
            obsCoviddataList.remove(o);
        else obsCountryList.remove(o);
    }
    //ενέργειες όταν τελειώσει το νήμα
    @Override
    public void done() {
        label.setText(
            "Η διαγραφή δεδομένων ολοκληρώθηκε με
            επιτυχία"
        );//εμφανίζει μήνυμα στο view
        //Μετά τις ενέργειες του νήματος επαναφέρω τα κουμπιά
        enableAllButtons();
        deleteButtonsVisibility();
        view.getLoadingLabel().setVisible(false);
    }
    }.execute();
    label.setText("Διαγραφή δεδομένων, παρακαλώ περιμέ-
    νετε...");//εμφανίζει μήνυμα στο view
    view.getLoadingLabel().setVisible(true);//θα εκτελε-
    στεί μαζί με το νήμα
}
}
});

```

```
//Κουμπί deleteData
buttons[4].addActionListener(new ActionListener() { //Εσωτερική ανώ-
νυμη κλάση ActionListener
@Override
public void actionPerformed(ActionEvent e) {
    JLabel label = view.getInfoText();
    //παίρνω τον δείκτη που έχει επιλεγεί
    int[] index = view.getDataTable().getSelectedRows();
    if(index.length>0){
        int x =
        JOptionPane.showConfirmDialog (view,
        "Είστε σίγουροι ότι θέλετε να διαγράψετε τα επιλεγμένα
        δεδομένα;",
        "Προσοχή",
        JOptionPane.YES_NO_OPTION);
        if(x == 0) {
            disableAllButtons(); //Λόγω νήματος απενεργοποιούμε
            όλα τα κουμπιά

            new SwingWorker<Void, Coviddata> () { //ορίζω το νήμα
                List<Coviddata> obsCoviddatalist =
                view.getCoviddatalist(); //Παίρνω τη λίστα του GUI
                ArrayList<Coviddata> toRemove =
                new ArrayList<>();
                @Override
                //λειτουργίες στο background ώστε τα γραφικά να μην κολλάνε
                protected Void doInBackground() throws
                InterruptedException {
                //Δημιουργώ πρώτα μια λίστα toRemove, για να μην έχω πρόβλημα με τους δείκτες
                κατά την αφαίρεση αντικειμένων
                    for(int i=0; i<index.length; i++){
                        toRemove.add(
                            obsCoviddatalist.get(index[i])
                        );
                    }
                    for(Coviddata c : toRemove) {
                        //Εκτελώ query που διαγράφει τα data με βάση την επιλεγμένη εγγραφή
                        dBm.deleteCoviddataByCountryNtype(c);
                        publish(c);
                        Thread.sleep(10);
                    }
                    return null;
                }
                @Override
                protected void process(List<Coviddata> chunks) {
                    Coviddata mostRecentValue =
                    chunks.get(chunks.size()-1);
                    obsCoviddatalist.remove(mostRecentValue);
                }
                //ενέργειες όταν τελειώσει το νήμα
                @Override
                public void done() {
                    label.setText(
                        "Η διαγραφή δεδομένων ολοκληρώθηκε με
                        επιτυχία"
                    ); //εμφανίζει μήνυμα στο view
                }
            }
        }
    }
}
```

```

        //Μετά τις ενέργειες του νήματος επαναφέρω τα κουμπιά
        enableAllButtons();
        deleteButtonsVisibility();
        view.getLoadingLabel().setVisible(false);
    }
    }.execute();
    label.setText("Διαγραφή δεδομένων, παρακαλώ περιμέ-
νετε...");//εμφανίζει μήνυμα στο view
    view.getLoadingLabel().setVisible(true);//θα εκτελε-
στεί μαζί με το νήμα
    }
}
});

//κουμπί εισαγωγή χωρών
buttons[1].addActionListener(new ActionListener() { //Εσωτερική ανώ-
νυμη κλάση ActionListener
@Override
public void actionPerformed(ActionEvent e) {
    JLabel label = view.getInfoText();
    label.setText("Επικοινωνία με τον server, παρακαλώ περιμέ-
νετε...");//εμφανίζει μήνυμα στο view
    List<Country> serverList = view.getServerList();
    CovidDataType datatype;//Αντικείμενο enum CovidDataType
    //παίρνει το επιλεγμένο type
    datatype = CovidDataType.getByLabelName(
        (String) view.getTypeComboBoc().getSelectedItem()
    );
    disableAllButtons();
    new SwingWorker<Void, Void> () {
        List<Country> countries;
        List<Country> obsCountryList = view.getCountryList();
        @Override
        protected Void doInBackground() {
            //Καλώ την static μέθοδο και ανακτώ όλα τα δεδομένα που υπάρχουν στον server
            //κατεβάζει τη λίστα με όλες τις χώρες και τα δεδομένα
            countries = ServerConnection.receiveData(datatype);
            //Στέλνω τα δεδομένα για εγγραφή των χωρών στην βάση δεδομένων
            if(countries != null)
                setTempData(
                    //καλεί το DbManager να εισάγει τα δεδομένα στη βάση
                    dbm.insertCountries(countries)
                );
            return null;
        }
        @Override
        public void done() {
            if(countries != null){ //Αν έχει γίνει κανονικά η σύν-
δεση με τον server
                //Στην περίπτωση που ο server δεν έχει χώρες για
                το type που έγινε αίτημα τις αφαιρώ από την λίστα
                serverList.retainAll(countries);
                for(Country country: countries) { //ενημερώνω την
λίστα για να ενημερώσει τον πίνακα

```

```

        if(!serverList.contains(country))
            serverList.add(country);
        if(!obsCountryList.contains(country))
            obsCountryList.add(country);
    }
    Collections.sort(obsCountryList);
}
//Μετά τις ενέργειες του νήματος επαναφέρω τα κουμπιά
enableAllButtons();
deleteButtonsVisibility();
view.getLoadingLabel().setVisible(false);
}
}.execute();
//Εκτελούνται παράλληλα με το νήμα του Swigworker
serverList.clear(); //Αδειάζει τον πίνακα του GUI
label = view.getDataLabel();
label.setText(
"<html>Πατώντας εισαγωγή θα εισάγετε δεδομένα είδος: "
+ datatype.getLabelName()
+ "<br><span style=\"color:red\">Για να εισάγετε άλλο είδος
επιλέξτε από τη λίστα "
+ "το είδος που επιθυμείτε και πατήστε λήψη δεδομένων
ξανά</span></html>");
label.setForeground(Color.BLUE);
view.getLoadingLabel().setVisible(true);
}
});

//κουμπί εισαγωγή δεδομένων
//αν δεν έχουμε κάνει λήψη είναι απενεργοποιημένο
buttons[2].setEnabled(!tempData.isEmpty());
buttons[2].addActionListener(new ActionListener() { //Εσωτερική ανώ-
νυμη κλάση ActionListener
@Override
public void actionPerformed(ActionEvent e) {
    List<Country> serverList = view.getServerList();
    String value =
serverList.get(view.getjServerList()
                .getSelectedIndex())
                .getName();
    if(value!=null) {
        disableAllButtons();
        insertingData = true;
        JLabel label = view.getInfoText();
        //εμφάνιση μηνύματος
        label.setText("Εισαγωγή δεδομένων στη βάση");
        new SwingWorker<Void, Coviddata> () {
            List<Coviddata> obsCoviddataList =
view.getCoviddataList();
            //Θέτει στο Selected τη λίστα με τα δεδομένα της επιλεγμένης χώρας
            List<Coviddata> selected = tempData.get(value);
            @Override
            protected Void doInBackground() {
                if(value != null) {
                    //καλώ τον DbManager να εισάγει τα δεδομένα

```

```

        dBm.insertData(selected);
    }
    //Θέλω ο πίνακας για είναι εύχρηστος να δείχνει μόνο 1 εγγραφή
    //δείχνοντας στον χρήστη μόνο όνομα χώρας και τύπο δεδομένων
    //Οπότε στο observable list θα εισάγω μόνο 1 εγγραφή
    //Έχω ορίσει κατάλληλα την equals στην κλάση Coviddata
    if(!obsCoviddataList.contains(
        selected.get(selected.size()-1)))
        publish(selected.get(selected.size()-1));
    return null;
}
@Override
protected void process(List<Coviddata> chunks) {
    //Κάνω add κάθε φορά το τελευταίο αντικείμενο
    Coviddata mostRecentValue =
        chunks.get(chunks.size()-1);
    //Ελέγχει αν υπάρχει για την χώρα και το είδος αναφορά στη λίστα
    for(Coviddata c: obsCoviddataList)
        if(c.getCountry().equals(
            mostRecentValue.getCountry())
            && c.getDatakind()
                == mostRecentValue.getDatakind()) {
            //Την αφαιρεί για να δείχνει η λίστα την νέα ημερομηνία έως
            obsCoviddataList.remove(c);
            break;
        }

        obsCoviddataList.add(mostRecentValue);
    //Ταξινομώ την λίστα πρώτα με βάση το όνομα χώρας και μετά με βάση τον τύπο
    //δεδομένων
    obsCoviddataList.sort(
        (Coviddata c1, Coviddata c2) -> {
            if(c1.getCountry().equals(c2.getCountry()))
                return c1.getDatakind()-c2.getDatakind();
            else
                return
                    c1.getCountry().compareTo(c2.getCountry());
        });
}
@Override
public void done() {
    //Μετά τις ενέργειες του νήματος επαναφέρω τα
    κουμπιά

    insertingData = false;
    enableAllButtons();
    deleteButtonsVisibility();
    view.getLoadingLabel().setVisible(false);
}
}.execute();
view.getLoadingLabel().setVisible(true); //θα εκτελεστεί
μαζί με το νήμα
}
});
//κουμπί backToOptions

```

```

        buttons[0].addActionListener(new ActionListener() { //Εσωτερική ανώ-
νυμη κλάση ActionListener
            @Override
            public void actionPerformed(ActionEvent e) {
                //Καλεί τον mainFrame να αλλάξει View
                appSystem.getMainFrame().showOptionsView();
                //Δημιουργεί τον Presenter για το νέο View
                Presenter p =
                new OvPresenter( appSystem.getMainFrame().getPanel(), appSystem);
                //Στέλνει στο σύστημα τον νέο Presenter και τον παλιό
για ενέργειες
                appSystem.setPresenter(p, appSystem.getPresenter());
            }
        });
        //Αν διαγραφεί μια χώρα δεν μπορούμε να κάνουμε εισαγωγή δεδομένων
view.getJServerList().addListSelectionListener((ListSelectionEvent e)
-> {
            //Κάθε φορά που ο χρήστης επιλέγει μια χώρα ελέγχω αν θα λειτουρ-
γεί το insertButton
            insertDataButtonState();
        });
    private void insertDataButtonState() {
        List<Country> obsCountryList = view.getCountryList();
        List<Country> serverList = view.getServerList();
        Country selected = null;
        //Αν υπάρχει επιλογή από τον χρήστη την βάζω στο selected
        if(view.getJServerList().getSelectedIndex() != -1)
            selected =
            serverList.get(view.getJServerList().getSelectedIndex());
        //Εάν δεν έχω κατεβάσει δεδομένα το insert δεν λειτουργεί
        if(serverList.isEmpty())
            buttons[2].setEnabled(false);
        //Εάν η χώρα δεν υπάρχει στην βάση δεδομένων το insert δεν λειτουργεί
        else if(!obsCountryList.contains(selected))
            buttons[2].setEnabled(false);
        //Εάν πραγματοποιείται εισαγωγή δεδομένων το insert δεν λειτουργεί
        else if(insertingData)
            buttons[2].setEnabled(false);
        //Σε όλες τις άλλες περιπτώσεις λειτουργεί
        else buttons[2].setEnabled(true);
    }
    //εμφανίζω τα deleteButtons με βάση την λίστα του GUI
    private void deleteButtonsVisibility() {
        buttons[3].setVisible(!view.getCountryList().isEmpty());
        buttons[4].setVisible(!view.getCoviddataList().isEmpty());
        //Το delete all data μπορεί να εμφανίζεται και χωρίς να έχω coviddata
στο database
        buttons[5].setVisible(!view.getCountryList().isEmpty());
    }
    //Κάνει όλα τα κουμπιά μη διαθέσιμα
    private void disableAllButtons() {
        for(JButton button: buttons)
            button.setEnabled(false);
    }

```

```
//Κάνει όλα τα κουμπιά διαθέσιμα
private void enableAllButtons() {
    for(JButton button: buttons)
        button.setEnabled(true);
    //Το insertData ελέγχεται κατά περίπτωση
    insertDataButtonState();
}
//Διαχειρίζεται το GUI σε σχέση με τις αλλαγές στο σύστημα
@Override
public void propertyChange(PropertyChangeEvent evt) {
    String propName = evt.getPropertyName();
    Object newVal = evt.getNewValue();
    Object oldVal = evt.getOldValue();

    //Μήνυμα για το αποτέλεσμα του αιτήματος από τον server και την
    αποθήκευση στο Database
    if("server connection".equalsIgnoreCase(propName) ||
        "Countries insert".equalsIgnoreCase(propName) ||
        "Covid data insert".equalsIgnoreCase(propName)){
        JLabel label = view.getInfoText();
        label.setText(String.valueOf(newVal));
    }
}

public void setTempData(TreeMap<String, List<Coviddata>> tempData) {
    this.tempData = tempData;
}
}
```

### Κλάση CountryDataView,

Εδώ με custom έχουμε αρχικοποιήσει σαν κενές λίστες, με τον ίδιο τρόπο, όπως στο προηγούμενο view, τις λίστες deathList, confirmedList και recoveredList. Ο entityManager, με τον ίδιο τρόπο γίνεται get από το DbManager. Έχουμε κάνει customize τον κώδικα του countryQuery, που δίνει τις χώρες στο countryList, το οποίο είναι συνδεδεμένο με το JComboBox, όπως παρακάτω:

```
countryQuery = java.beans.Beans.isDesignTime() ?
    null : em.createNamedQuery("Country.findByData", Country.class);
```

Επειδή παρατηρήθηκε, ότι όταν δημιουργούμε μια νέα βάση δεδομένων, και στην συνδέουμε στο σύστημα, το query που μας δίνει την λίστα με τις χώρες, σε κάθε χώρα επιστρέφει στο πεδίο coviddataList, ένα undirectedList, έχουμε εισάγει τον παρακάτω custom κώδικα, με επιλογή post-init στην λίστα του query:

```
for(Country c: countryList) {
    if(c.getCoviddataList() == null || c.getCoviddataList().isEmpty())
        c.setCoviddataList(
            em.createNamedQuery("Coviddata.findByCountry", Coviddata.class)
                .setParameter("countryname", c.getName())
                .getResultList()
        );
}
```

Εδώ να αναφέρουμε ότι έχουμε βάλει στα components του view, ένα JDialog που στην αρχή είναι κρυφό, που περιέχει τον χάρτη, και εμφανίζεται μετά από ενέργειες του Presenter. Το customize code με επιλογή post-init είναι:



```
map.setIconImage(new ImageIcon(getClass().getResource("/img/map.png")).getImage());  
map.setTitle("Προβολή δεδομένων σε χάρτη");  
map.add(AppSystem.getBrowser());  
map.pack();
```

**Κλάση CdvPresenter**, είναι ο Presenter του παραπάνω view, ορίζει όλα τα actionListeners. Χρησιμοποιεί την βοηθητική κλάση DatePicker, που θα παρουσιαστεί παρακάτω, για την επιλογή ημερομηνιών από τον χρήστη.

```
package presenters;  
  
import boundaryclasses.DbManager;  
import boundaryclasses.HtmlWriter;  
import coviddataapp.AppSystem;  
import java.awt.Dimension;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.beans.PropertyChangeEvent;  
import java.io.File;  
import java.time.Instant;  
import java.time.LocalDate;  
import java.time.ZoneId;  
import java.time.format.DateTimeFormatter;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.TreeSet;  
import javafx.application.Platform;  
import javax.swing.DefaultComboBoxModel;  
import javax.swing.JButton;  
import javax.swing.JCheckBox;  
import javax.swing.JComboBox;  
import javax.swing.JComponent;  
import javax.swing.JDialog;  
import javax.swing.JOptionPane;  
import javax.swing.JPanel;  
import javax.swing.JTextField;  
import javax.swing.SwingWorker;  
import model.Country;  
import model.Coviddata;  
import views.CountryDataView;  
import views.DataChartModal;  
import views.DatePicker;  
  
/**  
 *  
 */  
public class CdvPresenter implements Presenter{//CountryDataView Presenter  
  
    private final CountryDataView view;  
    private final AppSystem appSystem;  
    private JComponent[] buttons;  
    private final DbManager dBm;  
    private final TreeSet<Date> dateLimitData = new TreeSet<>();
```

```
private boolean writerError;

public CdvPresenter(final JPanel view, final AppSystem appSystem) {
    this.dbm = appSystem.getDbManager();
    this.view = (CountryDataView) view; //Κάνει cast το JPanel στην κλάση
    του View
    this.appSystem = appSystem; //Θέτει το σύστημα για να μπορεί να κάνει
    ενέργειες όταν πατάει ο χρήστης κουμπιά
    setUpViewEvents();
}

@Override
public final void setUpViewEvents() { //Ορίζει τα Event του View
    buttons = new JComponent[] {
        view.getBackToOptionsButton(), //0 = BackToOptionsButton
        view.getCountryComboBox(), // 1 = CountryComboBox
        view.getFromDateButton(), // 2 = FromDateButton
        view.getToDateBuuton(), // 3 = ToDateBuuton
        view.getDatesSearchButton(), // 4 = DatesSearchButton
        view.getCheckDeaths(), // 5 = CheckDeaths
        view.getCheckRecovered(), // 6 = CheckRecovered
        view.getCheckConfirmed(), // 7 = CheckConfirmed
        view.getRadioDay(), // 8 = RadioDay
        view.getRadioProod(), // 9 = RadioProod
        view.getDiagramButton(), // 10 = DiagramButton
        view.getMapButton(), // 11 = MapButton
        view.deleteDataButton() // 12 = DeleteDataButton()
    };

    //Επιλογέας χώρας
    ((JComboBox) buttons[1]).addActionListener((ActionEvent e) -> {
        //Εάν ο χρήστης έχει επιλέξει κάποια χώρα
        if(((JComboBox) buttons[1]).getSelectedIndex() != -1) {
            //Καθάρισε τις οριακές ημερομηνίες
            dateLimitData.clear();
            List<Coviddata> deathList = view.getDeathList();
            List<Coviddata> confirmedList = view.getConfirmedList();
            List<Coviddata> recoveredList = view.getRecoveredList();
            List<Integer> tabsWithData = new ArrayList<>();
            //Καθάρισε τις observable λίστες
            deathList.clear();
            confirmedList.clear();
            recoveredList.clear();
            //Πάρε το countryName που επέλεξε ο χρήστης
            Country country = (Country) view.getCountryComboBox().get-
            SelectedItem();
            //Βάλε όλες τις εγγραφές από το query στο deathList
            deathList.addAll(dbm.findByCountryNtype(country, 1));
            //Αν δεν είναι άδεια η λίστα
            if(!deathList.isEmpty()) {
                //Βάλε στις οριακές ημερομηνίες την πρώτη και την τελευταία ημερομηνία
                dateLimitData.add(deathList.get(0).getTrndate());
                dateLimitData.add(
                    deathList.get(deathList.size()-1).getTrndate())
            }
        }
    });
}
```

```
);  
//Κρύψε το label  
view.getDeathsTabLabel().setVisible(false);  
//Βάλε τον δείκτη του tab για θανάτους στη λίστα  
tabsWithData.add(0);  
}  
else//Αλλιώς εμφάνισε το label  
view.getDeathsTabLabel().setVisible(true);  
//Κάνε τα ίδια για το recoveredList  
recoveredList.addAll(dBm.findByCountryNtype(country,2));  
if(!recoveredList.isEmpty()) {  
dateLimitData.add(recoveredList.get(0).getTrndate());  
dateLimitData.add(  
recoveredList.get(recoveredList.size()-1).getTrndate()  
);  
view.getRecoveredTabLabel().setVisible(false);  
tabsWithData.add(1);  
}  
else  
view.getRecoveredTabLabel().setVisible(true);  
//Κάνε τα ίδια για το confirmedList  
confirmedList.addAll(dBm.findByCountryNtype(country,3));  
if(!confirmedList.isEmpty()) {  
dateLimitData.add(confirmedList.get(0).getTrndate());  
dateLimitData.add(  
confirmedList.get(confirmedList.size()-1).getTrndate()  
);  
view.getConfirmedTabLabel().setVisible(false);  
tabsWithData.add(2);  
}  
else  
view.getConfirmedTabLabel().setVisible(true);  
//Εάν έχουμε ορισμένες ημερομηνίες, άρα και δεδομένα  
if(!dateLimitData.isEmpty()) {  
JTextField fromDate = view.getFromDateTextField();  
JTextField toDate = view.getToDateTextField();  
//Πάρε την πρώτη ημερομηνία  
LocalDate localdate =  
Instant.ofEpochMilli(dateLimitData.first().getTime())  
.atZone(ZoneId.systemDefault())  
.toLocalDate();  
//Θέσε το κείμενο στο fromDate σύμφωνα με την 1η ημερομηνία  
fromDate.setText(localdate.format(  
DateTimeFormatter.ofPattern("dd/MM/yyyy")));  
//Πάρε την τελευταία ημερομηνία  
localdate =  
Instant.ofEpochMilli(dateLimitData.last().getTime())  
.atZone(ZoneId.systemDefault())  
.toLocalDate();  
//Θέσε το κείμενο στο toDate σύμφωνα με την τελευταία ημερομηνία  
toDate.setText(localdate.format(  
DateTimeFormatter.ofPattern("dd/MM/yyyy")));  
//Ενεργοποίησε το κουμπί fromDate  
buttons[2].setEnabled(true);  
//Ενεργοποίησε το κουμπί toDate
```

```
        buttons[3].setEnabled(true);
        //Έλεγχε την ενεργοποίηση των κουμπιών
        allButtonsEnability();
        //Ενεργοποίησε το κουμπί του χάρτη
        buttons[11].setEnabled(true);
        //Ενεργοποίησε το κουμπί για διαγραφή δεδομένων
        buttons[12].setEnabled(true);
        //Θέσε επιλεγμένο tab το πρώτο που έχει δεδομένα
        view.getJTabbedPane().setSelectedIndex(tabsWithData.get(0));
    }
}
});
//Κουμπί fromDate
((JButton) buttons[2]).addActionListener((ActionEvent e) -> {
    //Οριακές ημερομηνίες ημερολογίου, σύμφωνα με τις οριακές ημερο-
    μηνίες των δεδομένων
    LocalDate fromDate =
        Instant.ofEpochMilli(dateLimitData.first().getTime())
            .atZone(ZoneId.systemDefault())
            .toLocalDate();
    LocalDate toDate =
        Instant.ofEpochMilli(dateLimitData.last().getTime())
            .atZone(ZoneId.systemDefault())
            .toLocalDate();
    //Τρέχουσα επιλεγμένη ημερομηνία
    LocalDate currentDate =
        toLocalDate(view.getFromDateTextField().getText());
    //Κάλεσε το DatePicker
    String fromDateString =
        new DatePicker(view.getFromDateTextField(),
            fromDate,
            toDate,
            currentDate
        ).setPickedDate();
    //Εάν ο χρήστης δεν έκλεισε το παράθυρο
    if(!fromDateString.equals(""))
        //Θέσε το κείμενο με την ημερομηνία που επέλεξε ο χρήστης
        view.getFromDateTextField().setText(fromDateString);
    //Έλεγχε την ενεργοποίηση του searchButton
    searchButtonEnability();
});
//Κουμπί toDate
((JButton) buttons[3]).addActionListener((ActionEvent e) -> {
    //Οριακή ημερομηνία σύμφωνα με την επιλογή του fromDate
    String fromDateString = view.getFromDateTextField().getText();
    LocalDate fromDate = toLocalDate(fromDateString);
    //Οριακή ημερομηνία σύμφωνα με τα δεδομένα
    LocalDate toDate =
        Instant.ofEpochMilli(dateLimitData.last()
            .getTime())
            .atZone(ZoneId.systemDefault())
            .toLocalDate();
    //Τρέχουσα επιλεγμένη ημερομηνία
    LocalDate currentDate =
```

```

toLocalDate(view.getToDateTextField().getText());
//Κάλυψε το DatePicker
String toDateString =
new DatePicker(view.getFromDateTextField(),
               fromDate,
               toDate,
               currentDate
               ).setPickedDate();
//Εάν ο χρήστης δεν έκλεισε το παράθυρο
if(!toDateString.equals(""))
    //Θέσε το κείμενο με την ημερομηνία που επέλεξε ο χρήστης
    view.getToDateTextField().setText(toDateString);
//Έλεγχε την ενεργοποίηση του searchButton
searchButtonEnability();
});
//Κουμπί Search
((JButton) buttons[4]).addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //Αρχικοποίηση μεταβλητών
        Date fromDate =
            java.sql.Date.valueOf(toLocalDate(
                                   view.getFromDateTextField().getText()));

        Date toDate =
            java.sql.Date.valueOf(toLocalDate(
                                   view.getToDateTextField().getText()));

        List<Coviddata> deathList = view.getDeathList();
        List<Coviddata> confirmed = view.getConfirmedList();
        List<Coviddata> recovered = view.getRecoveredList();
        Country country =
            (Country) view.getCountryComboBox().getSelectedItem();
        //Καθάρισε τη deathList
        deathList.clear();
        //Βρες όλα τα δεδομένα ανάμεσα στις επιλεγμένες ημερομηνίες
        deathList.addAll(
            dBm.findByCoyntryDataBetweenDates(country,
                                                1,
                                                fromDate,
                                                toDate)
        );
        //Καθάρισε τη recovered
        recovered.clear();
        //Βρες όλα τα δεδομένα ανάμεσα στις επιλεγμένες ημερομηνίες
        recovered.addAll(
            dBm.findByCoyntryDataBetweenDates(country,
                                                2,
                                                fromDate,
                                                toDate)
        );
        //Καθάρισε τη confirmed
        confirmed.clear();
        //Βρες όλα τα δεδομένα ανάμεσα στις επιλεγμένες ημερομηνίες
        confirmed.addAll(
            dBm.findByCoyntryDataBetweenDates(country,
                                                3,

```

```

fromDate,
toDate)

    );
}
});
//Κουμπί προβολή σε διάγραμμα
((JButton) buttons[10]).addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        List<Coviddata> dataToDiagram = new ArrayList<>();
        List<Coviddata> deathList = view.getDeathList();
        List<Coviddata> comfirmed = view.getConfirmedList();
        List<Coviddata> recovered = view.getRecoveredList();
        //Τύπος δεδομένων για προβολή ημερήσια δεδομένα
        int qntType = 0;
        int linesNum = 0;
        //Αν είναι επιλεγμένο το CheckDeaths
        if(view.getCheckDeaths().isSelected()) {
            //Βάλε την λίστα θανάτων στη λίστα που θα γίνει διάγραμμα
            dataToDiagram.addAll(deathList);
            //Αύξησε τον μετρητή γραμμών
            linesNum++;
        }
        //Αν είναι επιλεγμένο το CheckRecovered
        if(view.getCheckRecovered().isSelected()) {
            //Βάλε την λίστα με τους ασθενείς που έχουν ανακάμψει στη λίστα που θα γίνει
            διάγραμμα
            dataToDiagram.addAll(recovered);
            //Αύξησε τον μετρητή γραμμών
            linesNum++;
        }
        //Αν είναι επιλεγμένο το getCheckConfirmed
        if(view.getCheckConfirmed().isSelected()) {
            //Βάλε την λίστα με τα επιβεβαιωμένα κρούσματα στη λίστα
            που θα γίνει διάγραμμα
            dataToDiagram.addAll(comfirmed);
            //Αύξησε τον μετρητή γραμμών
            linesNum++;
        }
        //Εάν είναι επιλεγμένο το radio για τα σωρευτικά δεδομένα
        if(view.getRadioProod().isSelected())
            //κάνε τον τύπο 1
            qntType = 1;
        //Αν δεν είναι άδεια η λίστα
        if(!dataToDiagram.isEmpty())
            //Εμφάνισε το διάγραμμα
            new DataChartModal(view,
                dataToDiagram,
                linesNum,
                qntType
            );
    }
});
//κουμπί προβολή σε χάρτη
((JButton) buttons[11]).addActionListener(new ActionListener() {

```

```

@Override
public void actionPerformed(ActionEvent e) {
    //Νήμα που εκτελεί εργασίες εκτός από αλλαγές στα γραφικά
    new SwingWorker<Void, Void> () {
        //Έλεγχος αν τελείωσαν οι εργασίες του Platform thread
        boolean finished = false;
        JDialog map = view.getMap();
        @Override
        //Η μέθοδος εκτελείται στο working thread
        protected Void doInBackground() throws
            InterruptedException {

            //Κατάσταση εγγραφής αρχείου
            writerError = true;
            TreeSet<Coviddata> selectedData = new TreeSet<>();
            selectedData.addAll(view.getConfirmedList());
            selectedData.addAll(view.getDeathList());
            selectedData.addAll(view.getRecoveredList());
            List<Country> countries = new ArrayList<>();
            //Από την επιλεγμένη χώρα, πάρε τα δεδομένα από την βάση δεδομένων
            countries.add(
                (Country)view.getCountryComboBox()
                    .getSelectedItem()
            );
            countries.get(0)
                .getCoviddatalist()
                .retainAll(selectedData);
            //Κάλεσε την writer να γράψει το Html για την χώρα και ζουμ 4
            HtmlWriter.writer(countries, 4, 500, 400);
            //Δίνω χρόνο να κλείσει ο buffer για να ανοίξει το σωστό αρχείο
            Thread.sleep(2000);
            //Δημιουργία του link
            String link = null;
            switch (((Country)view.getCountryComboBox()
                .getSelectedItem()).getName()) {
                //Αν έχει επιλεγεί το MS_Zaandam(Κρουαζιερόπλοιο)
                case "MS_Zaandam":
                    //Link που λέει το ιστορικό της κρουαζιέρας
                    link =
                        "https://en.wikipedia.org/wiki/COVID-19_pandemic_on_cruise_ships#Zaandam";
                    break;
                //Αν έχει επιλεγεί το Diamond_Princess(Κρουαζιερόπλοιο)
                case "Diamond_Princess":
                    //Link που λέει το ιστορικό της κρουαζιέρας
                    link =
                        "https://en.wikipedia.org/wiki/COVID-19_pandemic_on_Diamond_Princess";
                    break;
                default:
                    //To link που προβάλλει τον χάρτη
                    link =
                        new File("html\\mappage.html").toURI().toString();
                    break;
            }
            final String linkToLoad = link;
            //Στο thread του Jfx application
            Platform.runLater(() -> {

```

```
//Κάλεσε την loadLink για να φορτώσει το λινκ που στέλνω
AppSystem.getBrowser()
    .loadLink(linkToLoad,
        new Dimension(500, 400)
    );
    finished = true;
});
//Αυτοσχέδιος μηχανισμός συγχρονισμού 2 thread
//Μπλοκάρω το working thread μέχρι να τελειώσει το platform thread
float i = 0;
//Όσο το platform thread δεν έχει τελειώσει
while(!finished) {
    //Αν πέρασαν 5 δευτερόλεπτα
    if(i>=5.0f) {
        //Δώσε σφάλμα εγγραφής
        writerError = true;
        //Βγες από τη while
        break;
    }
    //Αύξησε κατά 1
    i=i+0.5f;
    //Περίμενε 1 δευτερόλεπτο
    Thread.sleep(500);
    System.out.println("είναι στη while");
}

return null;
}

@Override
//Όταν τελειώσει το working thread
protected void done() {
    //Αν η διαδικασία τελειώσει κανονικά
    if(!writerError) {

        //Κρύψε το loading label
        view.getMapLoadingLabel().setVisible(false);
        //Εμφάνισε τα κουμπιά
        allButtonsEnability();
        //Το JDialog που θα εμφανιστεί, να είναι μέσα στο view
        map.setLocation(view.getLocationOnScreen());
        //Φτιάξε το μέγεθος σύμφωνα με το περιεχόμενο
        map.pack();
        //Εμφάνισε το JDialog
        map.setVisible(true);
    }
    else {
        JOptionPane.showMessageDialog(view,
            "<html>Παρουσιάστηκε σφάλμα κατά την δημιουργία του χάρτη.<br>"
            + "Παρακαλώ πατήστε πάλι προβολή σε χάρτη!!!</html>",
            "Σφάλμα Χάρτη", JOptionPane.ERROR_MESSAGE);
        //Κρύψε το loading label
        view.getMapLoadingLabel().setVisible(false);
        //Εμφάνισε τα κουμπιά
        allButtonsEnability();
    }
}
```



```

        }
    }
    }.execute();
    //Θέσε ορατό το loading label όταν ξεκινήσει το νήμα
    view.getMapLoadingLabel().setVisible(true);
    //Απενεργοποίησε όλα τα κουμπιά
    disableAllButtons();
}
});
//κουμπί deleteData
((JButton) buttons[12]).addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //Ερώτηση στον χρήστη για επιβεβαίωση
        int x = JOptionPane.showConfirmDialog(view,
            "<html>Είστε σίγουροι ότι θέλετε να διαγράψετε<br>
            όλα τα δεδομένα για την επιλεγμένη χώρα;</html>",
            "Προσοχή",
            JOptionPane.YES_NO_OPTION);
        if(x==0) { //Εάν ναι
            List<Coviddata> deathList = view.getDeathList();
            List<Coviddata> confirmedList = view.getConfirmedList();
            List<Coviddata> recoveredList = view.getRecoveredList();
            List<Country> comboList = view.getCountryList();
            //Πάρε τη χώρα που επέλεξε ο χρήστης
            Country country =
                (Country) view.getCountryComboBox().getSelectedItem();
            //Διέγραψε όλα τα δεδομένα Coviddata για την χώρα αυτή
            dBm.deleteCoviddataByCountry(country);
            //Καθάρισε τις observable λίστες
            deathList.clear();
            confirmedList.clear();
            recoveredList.clear();
            dateLimitData.clear();
            //Αφαίρεσε την χώρα από την λίστα
            comboList.remove(country);
            //Θέσε πάλι το το combobox χωρίς την χώρα αυτή
            view.getCountryComboBox()
                .setModel(
                    new DefaultComboBoxModel<Object>(comboList.toArray()
                ));
            view.getCountryComboBox().setSelectedIndex(-1);
            allButtonsEnability();
        }
    }
});
//κουμπί backToOptions
((JButton) buttons[0]).addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //Εσωτερική ανώνυμη κλάση ActionListener με lamda
        //Καλεί τον mainFrame να αλλάξει View
        appSystem.getMainFrame().showOptionsView();
        //Δημιουργεί τον Presenter για το νέο View
        Presenter p =

```

```

        new OvPresenter( appSystem.getMainFrame()
                        .getPanel(), appSystem);
//Στέλνει στο σύστημα τον νέο Presenter και τον Παλιό για ενέργειες
        appSystem.setPresenter(p, appSystem.getPresenter());
    }
}

private void searchButtonEnability() {
    //Αρχικοποίηση μεταβλητών
    JTextField fromField = view.getFromDateTextField();
    JTextField toField = view.getDateTextField();
    //Εάν κάποιο από τα 2 field είναι κενό
    if(fromField.equals("") || toField.equals(""))
        //Απενεργοποίηση το κουμπί
        buttons[4].setEnabled(false);
    //Εάν δεν υπάρχει επιλεγμένη χώρα
    else if(view.getCountryComboBox().getSelectedIndex() == -1)
        //Απενεργοποίηση το κουμπί
        buttons[4].setEnabled(false);
    //Εάν οι ημερομηνία από είναι μετά την ημερομηνία μέχρι
    else if(!toLocalDate(fromField.getText())
            .isBefore(toLocalDate(toField.getText()))){
        //Απενεργοποίηση το κουμπί
        buttons[4].setEnabled(false);
        //Ενεργοποίηση το label
        view.getDateErrorLabel().setVisible(true);
    }
    else {//σε κάθε άλλη περίπτωση
        //Ενεργοποίηση το κουμπί
        buttons[4].setEnabled(true);
        //Απενεργοποίηση το label
        view.getDateErrorLabel().setVisible(false);
    }
}

//Πυθμίζει το enability των checkButtons και του diagramButton, σύμφωνα με
//τις λίστες
private void checkButtonsNdiagramEnability() {
    //Για θανάτους
    buttons[5].setEnabled(!view.getDeathList().isEmpty());
    ((JCheckBox)buttons[5]).setSelected(buttons[5].isEnabled());
    //Για ασθενείς που έχουν ανακάμψει
    buttons[6].setEnabled(!view.getRecoveredList().isEmpty());
    ((JCheckBox)buttons[6]).setSelected(buttons[6].isEnabled());
    //Για επιβεβαιωμένα κρούσματα
    buttons[7].setEnabled(!view.getConfirmedList().isEmpty());
    ((JCheckBox)buttons[7]).setSelected(buttons[7].isEnabled());
    Boolean enable = false;
    //Αν έστω 1 από τα checkButton είναι enabled κάνε και το diagram
    for(int i=5; i<=7; i++)
        if(buttons[i].isEnabled()) {
            enable = true;
            break;
        }
    buttons[10].setEnabled(enable);
}

```

```

    }

    private void disableAllButtons() {
        for(JComponent j: buttons) {
            j.setEnabled(false);
        }
    }

    //Ελέγχει το enablement όλων των κουμπιών
    private void allButtonsEnablement() {
        buttons[0].setEnabled(true);
        buttons[1].setEnabled(true);
        boolean enable = view.getCountryComboBox().getSelectedIndex() != -1;
        for(int i=2; i<buttons.length; i++)
            buttons[i].setEnabled(enable);
        searchButtonEnablement();
        checkButtonsNdiagramEnablement();
    }

    //Μετατροπή ημερομηνίας String σε LocalDate
    private LocalDate toLocalDate (String date) {
        return LocalDate.of(
            Integer.parseInt(date.substring(date.lastIndexOf("/") + 1)), //Έτος
            Integer.parseInt(
                date.substring(date.indexOf("/") + 1, date.lastIndexOf("/")
            ), //Μήνας
            Integer.parseInt(date.substring(0, date.indexOf("/")))); //Μέρα
    }

    @Override
    //Διαχειρίζεται το GUI σε σχέση με τις αλλαγές στο σύστημα

    public void propertyChange(PropertyChangeEvent evt) {
        String propName = evt.getPropertyName();
        Object newVal = evt.getNewValue();
        Object oldVal = evt.getOldValue();

        //Στέλνει το αποτέλεσμα του γραψίματος του αρχείου
        if("Html writer, oldVal->true, newVal->hasIOError"
            .equalsIgnoreCase(propName)) {
            writerError = (boolean) newVal;
        }
    }
}

```

#### Κλάση DatePicker,

βρήκαμε τον κώδικα στην πηγή που αναφέρεται στα σχόλια και τον παραμετροποιήσαμε αρκετά, ώστε να έχει την επιθυμητή συμπεριφορά. Πχ να μην μπορεί ο χρήστης να επιλέξει ημερομηνία πριν το fromDate ή μετά το toDate.

```

package views;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;

```

```
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;
import javax.swing.border.Border;

/**
 * Πηγή https://www.roseindia.net/tutorial/java/swing/datePicker.html
 */
public class DatePicker extends JDialog{
    private LocalDate fromDate; //Ελάχιστη ημερομηνία
    private LocalDate toDate; //Μέγιστη ημερομηνία
    private LocalDate cal; //Ημερομηνία εκκίνησης
    private LocalDate selectedDate;
    private int month; //μεταβλητή που διαχειρίζεται τον μήνα
    private int year; //μεταβλητή που διαχειρίζεται τον χρόνο
    //label που γράφει μήνα και έτος
    private JLabel l = new JLabel("", JLabel.CENTER);
    private String day = ""; //μεταβλητή που διαχειρίζεται την ημέρα

    private JButton[] button = new JButton[42];
    private JLabel dayLabel;
    private Border blackline = BorderFactory.createLineBorder(Color.GRAY);
    private JButton previous;
    private JButton next;

    public DatePicker(Component parent,
                      LocalDate fromDate,
                      LocalDate toDate,
                      LocalDate currentDate) {
        //Αρχικοποίηση τιμών
        this.fromDate = fromDate;
        this.toDate = toDate;
        if(currentDate.isBefore(fromDate))
            this.cal = fromDate;
        else this.cal = currentDate;
        this.month = cal.getMonthValue();
        this.year = cal.getYear();
        this.selectedDate = currentDate;
        initDatePicker(parent);
    }

    private void initDatePicker(Component parent) {
        setTitle("Επιλέξτε ημερομηνία");
        //Αλλαγή εικονιδίου στο παράθυρο
        setIconImage(
```

```
new ImageIcon(getClass().getResource("/img/calendar.png")).getImage()
);
setModal(true); //Αποκλειστικό παράθυρο
setResizable(false);
setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
String[] header =
{ "Mon", "Tue", "Wed", "Thur", "Fri", "Sat", "Sun" };
//Πάνελ που θα μπουν τα περιεχόμενα
JPanel p1 = new JPanel(new GridLayout(7, 7));
//Επιθυμητές διαστάσεις πάνελ
p1.setPreferredSize(new Dimension(430, 120));
//Αρχικοποίηση label με τις μέρες
for(int x = 0; x < 7; x++){
    dayLabel = new JLabel(header[x], SwingConstants.CENTER);
    dayLabel.setForeground(Color.red);
    dayLabel.setBackground(Color.white);
    dayLabel.setBorder(blackline);
    p1.add(dayLabel);
}
//Αρχικοποίηση κουμπιών
for (int x = 0; x < button.length; x++) {
    final int selection = x;
    button[x] = new JButton();
    button[x].setFocusPainted(false);
    button[x].setBackground(Color.white);
    button[x].addActionListener(new ActionListener() {
        //Αν πατηθεί το κουμπί, θέτει το day και κλείνει το παράθυρο
        public void actionPerformed(ActionEvent ae) {
            day = button[selection].getActionCommand();
            dispose();
        }
    });
    p1.add(button[x]);
}
//Αρχικοποίηση κουμπιού previous
JPanel p2 = new JPanel(new GridLayout(1, 3));
previous = new JButton("<< Previous");
//Αν πατηθεί το κουμπί αφαιρεί έναν μήν από το cal και το χτίζει πάλι
previous.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        cal = cal.minusMonths(1);
        next.setEnabled(true);
        displayDate();
    }
});
p2.add(previous);
p2.add(l);
//Αρχικοποίηση του κουμπιού next
next = new JButton("Next >>");
//Αν πατηθεί αυξάνει 1 μήνα το cal και το χτίζει πάλι
next.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        cal = cal.plusMonths(1);
```

```

        previous.setEnabled(true);
        displayDate();
    }

});
p2.add(next);
//τοποθέτηση του πάνελ με τα κουμπιά
this.add(p1, BorderLayout.CENTER);
//τοποθέτηση του πάνελ με τα previews next
this.add(p2, BorderLayout.SOUTH);
this.pack();
this.setLocationRelativeTo(parent);
displayDate();
this.setVisible(true);
}

//Εμφανίζει τις μέρες στα κουμπιά για κάποιο μήνα και κάποιο έτος
public void displayDate() {
    month = cal.getMonthValue();
    year = cal.getYear();
    //Ξεκινάω από 1 του μήνα για να γράψω τις τιμές
    cal = LocalDate.of(year, month, 1);
    int dayOfWeek = cal.getDayOfWeek().getValue();
    int daysInMonth = cal.lengthOfMonth();
    //Για κάθε κουμπί
    for (int x = 0; x < button.length; x++){
        //Αν ο δείκτης είναι πριν την πρώτη μέρα του μήνα ή μετά την τελευταία μέρα
        if(x < dayOfWeek-1 || x > dayOfWeek+daysInMonth-2){
            button[x].setEnabled(false); //απενεργοποίησε το κουμπί
            button[x].setText(""); //Σβήσε το κείμενο
        }
        //Αλλιώς
        else {
            button[x].setText(""); //Σβήσε το κείμενο
            button[x].setEnabled(true); //Ενεργοποίησε το κουμπί
        }
        //Σε κάθε κουμπί το χρώμα της γραμματοσειράς να είναι μαύρο
        button[x].setForeground(Color.BLACK);
    }

    //Διαπερνάω τα κουμπιά μόνο που ο δείκτης τους είναι μέσα σε ημέρα του μήνα
    for (int x = dayOfWeek-1, day1 = 1; day1 <= daysInMonth; x++, day1++){
        {
            //Γράψε τη μέρα στο κουμπί
            button[x].setText("" + day1);
            //Αν η μέρα αντιστοιχεί με την προηγούμενη επιλογή του χρήστη
            if(selectedDate.getDayOfMonth() == day1
            && selectedDate.getMonthValue() == month && selectedDate.getYear() == year) {
                button[x].setForeground(Color.red); //Κάνε κόκκινα τα γράμματα
                button[x].setEnabled(false); //Απενεργοποίησε το κουμπί
            }
        }
        //Γράψε στο label μήνα και έτος
        l.setText(cal.getMonth()+" "+String.valueOf(cal.getYear()));
        //Εάν ο μήνας και το έτος είναι ίσα με το fromDate
        if ((month == fromDate.getMonthValue())
            && (year == fromDate.getYear())) {
            previous.setEnabled(false); //Απενεργοποίησε το κουμπί previews

```

```
//Διαπέρασε όλα τα κουμπιά πριν την αποδεκτή ημερομηνία
for(int x = 0; x < dayOfWeek+fromDate.getDayOfMonth()-2; x++)
    button[x].setEnabled(false); //απενεργοποίησε τα
}
//Εάν ο μήνας και το έτος είναι ίσα με το toDate
if (month == toDate.getMonthValue() && year == toDate.getYear()){
    next.setEnabled(false); //απενεργοποίησε το next
    //Διαπέρασε όλα τα κουμπιά μετά την αποδεκτή ημερομηνία
    for(int x = dayOfWeek+toDate.getDayOfMonth()-1;
        x < button.length;
        x++)
        button[x].setEnabled(false); //απενεργοποίησε τα
    }
}
//Καλείται μαζί με τον constructor
public String setPickedDate() {
    //Αν ο χρήστης έχει πατήσει X για να κλείσει το παράθυρο
    if (day.equals(""))
        return day; //Επιστρέφει το κενό
    //Ημερομηνία που επέλεξε ο χρήστης
    LocalDate date = LocalDate.of(year, month, Integer.parseInt(day));
    //Την επιστρέφει στο επιθυμητό format
    return date.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"));
}
}
```

#### Κλάση DataMapView,

στο JComboBox, με την επιλογή customize code και post-init, έχει προστεθεί ο κώδικας:

```
countriesComboBox.setModel(
new javax.swing.DefaultComboBoxModel<>(mainCountryList.toArray()));
countriesComboBox.setSelectedIndex(-1);
```

Επειδή παρατηρήθηκε, ότι όταν δημιουργούμε μια νέα βάση δεδομένων, και στην συνδέουμε στο σύστημα, το query που μας δίνει την λίστα με τις χώρες, σε κάθε χώρα επιστρέφει στο πεδίο coviddataList, ένα undirectedList, έχουμε εισάγει τον παρακάτω custom κώδικα, με επιλογή post-init στην λίστα του query:

```
for(Country c: mainCountryList) {
    if(c.getCoviddataList() == null || c.getCoviddataList().isEmpty())
        c.setCoviddataList(
            em.createNamedQuery("Coviddata.findByCountry", Coviddata.class)
                .setParameter("countryname", c.getName())
                .getResultList()
        );
}
```

Ο entityManager αρχικοποιείται με τον ίδιο τρόπο, όπως τα υπόλοιπα view, ενώ το restCountiresList, αρχικοποιείται σαν κενή λίστα και με επιλογή post-init, κάνουμε addAll, τα περιεχόμενα της mainCountryList. Στο query για την mainCountryList έχει γίνει customize code όπως παρακάτω:

```
countryListQuery = java.beans.Beans.isDesignTime() ?
null :
em.createNamedQuery("Country.findByData", Country.class);
```

Σε αυτό το view, ο χάρτης δεν εμφανίζεται σε ξεχωριστό παράθυρο, αλλά έχουμε τοποθετήσει ένα JPanel μέσα στο view, με GridLayout, και ανάλογα τις επιλογές του χρήστη ο presenter του view κάνει add ή remove το JfxPanel του χάρτη.

#### Κλάση DmvPresenter,

όπως και οι προηγούμενοι presenter, θέτει τα actionListeners και διαχειρίζεται όλα τα event του view.

```
package presenters;

import boundaryclasses.HtmlWriter;
import coviddataapp.AppSystem;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.beans.PropertyChangeEvent;
import java.io.File;
import java.time.Instant;
import java.time.LocalDate;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.TreeSet;
import javafx.application.Platform;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingWorker;
import model.Country;
import model.Coviddata;
import views.DataMapView;
import views.DatePicker;

/**
 *
 *
 */
public class DmvPresenter implements Presenter{//DataMapView Presenter

    private final DataMapView view;
    private final AppSystem appSystem;
    private JComponent[] buttons;
    private final TreeSet<Date> dateLimitData = new TreeSet<>();
    private final List<Country> selectedCountries = new ArrayList<>();
    private boolean writerError;

    public DmvPresenter(final JPanel view, final AppSystem appSystem) {
        //Κάνει cast το JPanel στην κλάση του View
        this.view = (DataMapView) view;//Θέτει το σύστημα για να μπορεί να
        //κάνει ενέργειες όταν πατάει ο χρήστης κουμπιά
        this.appSystem = appSystem;
        setUpViewEvents();
    }
}
```



```
@Override
public final void setUpViewEvents() { //Ορίζει τα Event του View
    buttons = new JComponent[] {
        view.getBackToOptionsButton(), //0 = BackToOptionsButton
        view.getCountriesComboBox(), // 1 = CountriesComboBox
        view.getFromButton(), //      2 = FromButton
        view.getToButton(), //      3 = ToButton
        view.getShowMapButton() //    4 = ShowMapButton
    };
    //Αρχικοποίηση των οριακών ημερομηνιών
    for (Country c: view.getMainCountryList()) {
        TreeSet<CovidData> countryData = new TreeSet<>();
        countryData.addAll(c.getCovidDataList());
        dateLimitData.add(countryData.first().getTrndate());
        dateLimitData.add(countryData.last().getTrndate());
    }
    //Αρχικοποίηση των textField με τις οριακές ημερομηνίες
    if (!dateLimitData.isEmpty()) {
        JTextField fromDate = view.getFromTextField();
        JTextField toDate = view.getToTextField();
        //Πάρε την πρώτη ημερομηνία
        LocalDate localdate =
            Instant.ofEpochMilli(dateLimitData.first().getTime())
                .atZone(ZoneId.systemDefault())
                .toLocalDate();
        //Θέσε το κείμενο στο fromDate σύμφωνα με την 1η ημερομηνία
        fromDate.setText(
            localdate.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"))
        );
        //Πάρε την τελευταία ημερομηνία
        localdate =
            Instant.ofEpochMilli(dateLimitData.last().getTime())
                .atZone(ZoneId.systemDefault())
                .toLocalDate();
        //Θέσε το κείμενο στο toDate σύμφωνα με την τελευταία ημερομηνία
        toDate.setText(
            localdate.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"))
        );
    }
    //Επιλογήας χώρας
    ((JComboBox)buttons[1]).addActionListener((ActionEvent e) -> {
        List<Country> allCountries = view.getMainCountryList();
        List<Country> restCountries = view.getRestCountriesList();
        //Η χώρα που επέλεξε ο χρήστης
        Country selected =
            (Country) ((JComboBox)buttons[1]).getSelectedItem();
        JPanel mapPanel = view.getMapPanel();
        //Για να αφαιρέσω την επιλεγμένη χώρα από την λίστα με τις υπόλοιπες χώρες
        restCountries.clear();
        restCountries.addAll(allCountries);
        restCountries.remove(selected);
        //Για να ανανεώσει το view
        view.getCountriesJList().repaint();
        selectedCountries.clear();
    });
}
```

```
//Στις επιλεγμένες χώρες στην θέση 0 μπαίνει η κύρια χώρα
selectedCountries.add(0, selected); //όποτε αλλάζει κύρια χώρα ο
χρήστης καθαρίζω τις επιλογές στις υπόλοιπες χώρες
view.getCountriesJList().clearSelection();
//Μετά την επιλογή χώρας ενεργοποιούνται τα κουμπιά
for(int i=2; i<=4; i++) {
    buttons[i].setEnabled(true);
}
});
//κουμπί from
((JButton) buttons[2]).addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
//Οριακές ημερομηνίες ημερολογίου, σύμφωνα με τις οριακές ημερομηνίες των δε-
δομένων

        LocalDate fromDate =
            Instant.ofEpochMilli(dateLimitData.first().getTime())
                .atZone(ZoneId.systemDefault())
                .toLocalDate();

        LocalDate toDate =
            Instant.ofEpochMilli(dateLimitData.last().getTime())
                .atZone(ZoneId.systemDefault())
                .toLocalDate();

        //Τρέχουσα επιλεγμένη ημερομηνία
        LocalDate currentDate =
            toLocalDate(view.getFromTextField().getText());
        //Κάλεσε το DatePicker
        String fromDateString =
            new DatePicker(view.getFromTextField(),
                           fromDate,
                           toDate,
                           currentDate
                           ).setPickedDate();
        //Εάν ο χρήστης δεν έκλεισε το παράθυρο
        if(!fromDateString.equals(""))
            //Θέσε το κείμενο με την ημερομηνία που επέλεξε ο χρήστης
            view.getFromTextField().setText(fromDateString);
        //Έλεγχε την ενεργοποίηση του searchButton
        showMapButtonEnability();
    }
});
//Κουμπί to
((JButton) buttons[3]).addActionListener((ActionEvent e) -> {
    //Οριακή ημερομηνία σύμφωνα με την επιλογή του fromDate
    String fromDateString = view.getFromTextField().getText();
    LocalDate fromDate = toLocalDate(fromDateString);
    //Οριακή ημερομηνία σύμφωνα με τα δεδομένα
    LocalDate toDate =
        Instant.ofEpochMilli(dateLimitData.last().getTime())
            .atZone(ZoneId.systemDefault())
            .toLocalDate();

    //Τρέχουσα επιλεγμένη ημερομηνία
    LocalDate currentDate =
        toLocalDate(view.getToTextField().getText());
    //Κάλεσε το DatePicker
```

```
String toDateString =
new DatePicker(view.getFromTextField(),
               fromDate,
               toDate, currentDate
               ).setPickedDate();
//Εάν ο χρήστης δεν έκλεισε το παράθυρο
if(!toDateString.equals(""))
    //Θέσε το κείμενο με την ημερομηνία που επέλεξε ο χρήστης
    view.getToTextField().setText(toDateString);
//Έλεγχε την ενεργοποίηση του searchButton
showMapButtonEnability();
});
//κουμπί προβολή χάρτη
((JButton) buttons[4]).addActionListener(new ActionListener() {
    JPanel mapPanel = view.getMapPanel();
    @Override
    public void actionPerformed(ActionEvent e) {
        //Νήμα που εκτελεί εργασίες εκτός από αλλαγές στα γραφικά
        new SwingWorker<Void, Void> () {
            boolean finished = false;
            @Override
            //Η μέθοδος εκτελείται στο working thread
            protected Void doInBackground() throws
                InterruptedException {

                Date from =
                java.sql.Date.valueOf(
                    toDate(view.getFromTextField().getText())
                );
                Date to =
                java.sql.Date.valueOf(
                    toDate(view.getToTextField().getText())
                );
                //Διαπερνάει όλες τις επιλεγμένες χώρες από τις υπόλοιπες χώρες
                for(int i:
                    view.getCountriesJList().getSelectedIndices())
                    //τις βάζει στη λίστα μαζί με την βασική χώρα
                    selectedCountries.add(
                        (Country) view.getRestCountriesList().get(i)
                    );
                //Δημιουργία λίστα που θα στείλουμε στο htmlwriter
                List<Country> toHtmlCountries = new ArrayList<>();
                //Εάν ο χρήστης έχει αλλάξει τις επιθυμητές ημερομηνίες
                if(!from.equals(dateLimitData.first())
                || !to.equals(dateLimitData.last())) {
                    //Διαπερνάμε όλες τις επιλεγμένες χώρες
                    for(Country country: selectedCountries) {
                        //Δημιουργία νέας χώρας
                        Country c = new Country();
                        //με ίδιο id
                        c.setCountry(country.getCountry());
                        //ίδιο όνομα
                        c.setName(country.getName());
                        //ίδιες συντεταγμένες
                        c.setLat(country.getLat());
                        c.setLong1(country.getLong1());
                    }
                }
            }
        };
    }
});
```

```

//Τροποποίηση της λίστας με τα coviddata
List<Coviddata> newList = new ArrayList<>();
//Διαπερνάμε τα coviddata της χώρας
for(Coviddata data:
    country.getCoviddataList()) {
    //Προσθέτουμε μόνο αυτά που είναι ανάμεσα στις ημερομηνίες που θέλουμε
    if(!data.getTrndate().before(from)
        && !data.getTrndate().after(to))
        newList.add(data);
    }
    //θέτουμε στην νέα χώρα την λίστα
    c.setCoviddataList(newList);
    //προσθέτουμε την χώρα στη λίστα που θα πάει για το htmlwriter
    toHtmlCountries.add(c);
}
//Αν ο χρήστης έχει επιλέξει τις αρχικές ημερ/νίες
else
    //η λίστα για το html δείχνει στην λίστα των επιλεγμένων χωρών
    toHtmlCountries = selectedCountries;
    //Κατάσταση εγγραφής αρχείου
    writerError = true;
    System.out.println("Πάει για writer");
    //Κάλεσε την writer να γράψει το Html για την χώρα και ζουμ 2
    HtmlWriter.writer(toHtmlCountries, 2, 834, 441);
    System.out.println(
        "τελείωσε writer writerError="+writerError);
    //Δίνω χρόνο να κλείσει ο buffer για να ανοίξει το σωστό αρχείο
    Thread.sleep(2000);
    //Αν η διαδικασία τελείωσε κανονικά
    String link = null;
    link =
        new File("html\\mappage.html").toURI().toString();
    final String linkToLoad = link;
    //Στο thread του Jfx application
    Platform.runLater(() -> {
        System.out.println("Πάει για loadlink");
        //Κάλεσε την loadLink για να φορτώσει το link που στέλνω
        AppSystem.getBrowser()
            .loadLink(linkToLoad,
                new Dimension(834, 441)
            );
    });
    //Αν έχει τελειώσει το thread με την φόρτωση του link θέτει το finished true
    finished = true;
    System.out.println(
        "τελείωσε loadlink, finished="+finished);
    });
    //Αυτοσχέδιος μηχανισμός συγχρονισμού 2 thread
    //Μπλοκάρω το working thread μέχρι να τελειώσει το platform thread
    int i = 0;
    //Όσο το platform thread δεν έχει τελειώσει
    while(!finished) {
        //Αν πέρασαν 5 δευτερόλεπτα
        if(i>=5) {
            //Δώσε σφάλμα εγγραφής

```

```
        writerError = true;
        //Βγες από τη while
        break;
    }
    //Αύξησε κατά 1
    i++;
    //Περίμενε 1 δευτερόλεπτο
    Thread.sleep(1000);
    System.out.println("είναι στη while");
}

return null;
}

@Override
//Όταν τελειώσει το working thread
protected void done() {
    System.out.println("Μπήκε done");
    if(!writerError) {
        //κρύβω το LoadingLabel
        view.getLoadingLabel().setVisible(false);
        //Το αφαιρώ από το πάνελ μαζί με το text
        mapPanel.remove(view.getInfoTextLabel());
        mapPanel.remove(view.getLoadingLabel());
        mapPanel.repaint();
        //Βάζω στο πάνελ το JFX Πάνελ με τον browser
        mapPanel.add(AppSystem.getBrowser());
//Καθαρίζω την λίστα με τις επιλεγμένες χώρες για την επόμενη επιλογή του
//χρήστη
        selectedCountries.clear();
        //Καθαρίζω τον επιλογέα βασικής χώρας
        ((JComboBox)buttons[1]).setSelectedIndex(-1);
//Καθαρίζω το jlist με τις επιλογές υπόλοιπων χωρών
        view.getCountriesJList().clearSelection();
        //Εμφάνισε τα κουμπιά
        enableAllButtons();
    }
    else {
        JOptionPane.showMessageDialog(view,
            "<html>Παρουσιάστηκε σφάλμα κατά την δημιουργία του χάρτη.<br>"
+ "Παρακαλώ επιλέξτε πάλι προβολή σε χάρτη!!!</html>",
            "Σφάλμα Χάρτη",
            JOptionPane.ERROR_MESSAGE);
        //Κρύψε το loading label
        view.getLoadingLabel().setVisible(false);
        //Εμφάνισε τα κουμπιά
        enableAllButtons();
    }
}

}.execute();
//Εκτελείται με το πάτημα του κουμπιού και κρύβει τον browser μέχρι να γρα-
φτεί το αρχείο
mapPanel.remove(AppSystem.getBrowser());
mapPanel.repaint();
mapPanel.add(view.getLoadingLabel());
```

```

        //Θέσε ορατό το loading label όταν ξεκινήσει το νήμα
        view.getLoadingLabel().setVisible(true);
        //Απενεργοποίησε όλα τα κουμπιά
        disableAllButtons();
    }
});
//Κουμπί πίσω στο μενού
//Εσωτερική ανώνυμη κλάση ActionListener με lamda
((JButton)buttons[0]).addActionListener((ActionEvent e) -> {
    //Καλεί τον mainFrame να αλλάξει View
    appSystem.getMainFrame().showOptionsView();
    //Δημιουργεί τον Presenter για το νέο View
    Presenter p =
        new OvPresenter( appSystem.getMainFrame().getPanel(), appSystem)
    //Στέλνει στο σύστημα τον νέο Presenter και τον Παλιό για ενέργειες
    appSystem.setPresenter(p, appSystem.getPresenter());
});

}
//Απενεργοποιεί όλα τα κουμπιά
private void disableAllButtons() {
    for(JComponent j: buttons)
        j.setEnabled(false);
}
//Ενεργοποιεί όλα τα κουμπιά
private void enableAllButtons() {
    for(JComponent j: buttons)
        j.setEnabled(true);
    showMapButtonEnability();
}
//Ενεργοποιεί το showMapButton ανάλογα την περίπτωση
private void showMapButtonEnability() {
    //Αρχικοποίηση μεταβλητών
    JTextField fromField = view.getFromTextField();
    JTextField toField = view.getToTextField();
    //Εάν κάποιο από τα 2 field είναι κενό
    if(fromField.equals("") || toField.equals(""))
        //Απενεργοποίησε το κουμπί
        buttons[4].setEnabled(false);
    //Εάν δεν υπάρχει επιλεγμένη χώρα
    else if(view.getCountriesComboBox().getSelectedIndex() == -1)
        //Απενεργοποίησε το κουμπί
        buttons[4].setEnabled(false);
    //Εάν οι ημερομηνία από είναι μετά την ημερομηνία μέχρι
    else if(!toLocalDate(fromField.getText())
        .isBefore(toLocalDate(toField.getText()))){
        //Απενεργοποίησε το κουμπί
        buttons[4].setEnabled(false);
        //Ενεργοποίησε το label
        view.getDateErrorLabel().setVisible(true);
    }
    else {//σε κάθε άλλη περίπτωση
        buttons[4].setEnabled(true); //Ενεργοποίησε το κουμπί
        //Απενεργοποίησε το label
        view.getDateErrorLabel().setVisible(false);
    }
}

```

```

    }
}
//Μετατροπή ημερομηνίας String σε LocalDate
private LocalDate toLocalDate (String date) {
    return LocalDate.of(
Integer.parseInt(date.substring(date.lastIndexOf("/") + 1)), //Έτος
Integer.parseInt(date.substring(date.indexOf("/") + 1,
    date.lastIndexOf("/") + 1)), //Μήνας
Integer.parseInt(date.substring(0, date.indexOf("/") + 1))); //Μέρα
    }

@Override
//Διαχειρίζεται το GUI σε σχέση με τις αλλαγές στο σύστημα
public void propertyChange(PropertyChangeEvent evt) {
    String propName = evt.getPropertyName();
    Object newVal = evt.getNewValue();
    Object oldVal = evt.getOldValue();

    //Στέλνει το αποτέλεσμα του γραψίματος του αρχείου
    if ("Html writer, oldVal->true, newVal->hasIOError"
        .equalsIgnoreCase(propName)) {
        writerError = (boolean) newVal;
    }
}
}

```

Οι κλάσεις HtmlBrowser και DataChartModal, θα παρουσιαστούν στο ερώτημα για την παρουσίαση χάρτη και γραφήματος.

## 3.2 Ερώτημα Γ – Παρουσίαση Χάρτη και Γραφημάτων

### Κλάση DataChartModal,

στην ουσία από τον κώδικα που μας δόθηκε με την εκφώνηση αλλάξαμε κάποια πράγματα, όπως ότι η κλάση αυτή επεκτείνει ένα JDialog, οπότε όταν δημιουργούμε ένα αντικείμενο, εμφανίζεται στην οθόνη ένα JDialog που περιλαμβάνει το γράφημα. Ο κατασκευαστής έχει τροποποιηθεί ώστε να παίρνει σαν όρισμα μια λίστα με τα coviddata που θέλουμε να παρουσιάσει, τον αριθμό των γραμμών και τον τύπο(δεδομένα ημέρας ή σωρευτικά δεδομένα).

```

package views;

import coviddataapp.CovidDataType;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Component;
import java.awt.FlowLayout;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.List;

```

```
import javax.swing.ImageIcon;
import javax.swing.JDialog;
import javax.swing.JPanel;
import model.Coviddata;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.CategoryAxis;
import org.jfree.chart.axis.CategoryLabelPositions;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.LineAndShapeRenderer;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;

public class DataChartModal extends JDialog {

    public DataChartModal(JPanel parent, List<Coviddata> coviddataList, int
linesNum, int qntType) {

        final CategoryDataset dataset = createDataset(coviddataList,
qntType);
        final JFreeChart chart = createChart(dataset, coviddataL-
ist.get(0).getCountry().getName(), linesNum);
        final ChartPanel chartPanel = new ChartPanel(chart);

        initModal(chartPanel, parent);
    }
    private void initModal(ChartPanel chartPanel, Component parent){
        //LayoutManager
        setLayout(new FlowLayout());
        //Βάλει το γράφημα στο JDialog
        add(chartPanel);
        //Παράθυρο αποκλειστικής χρήσης
        setModal(true);
        //Θέτει τον τίτλο
        setTitle("Προβολή δεδομένων σε διάγραμμα");
        //Αλλαγή εικονιδίου στο παράθυρο
        setIconImage(
new ImageIcon(getClass().getResource("/img/chart.png")).getImage());
        //Αν πατηθεί X, κλείνει το παράθυρο, δεν το κρύβει
        setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        //Δεν μπορεί ο χρήστης να αλλάξει το μέγεθος
        setResizable(false);
        //Εμφανίζεται στην τοποθεσία του parent view
        setLocation(parent.getLocationOnScreen());
        //Φτιάχνει το μέγεθος του παραθύρου σύμφωνα με το περιεχόμενο
        pack();
        //Εμφανίζει το παράθυρο
        setVisible(true);
    }
    private CategoryDataset createDataset(List<Coviddata> coviddataList,
int qntType) {

        // create the dataset...
```



```

final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
//Βάλτε όλα τα coviddata στο γράφημα
for (Coviddata d: coviddataList)
    //row Value
    dataset.addValue(d.getQty() : d.getProodqty(),
        //row Key
        CovidDataType.getNameByValue(d.getDatakind()),
        //column key
        Instant.ofEpochMilli(d.getTrndate().getTime())
            .atZone(ZoneId.systemDefault())
            .toLocalDate()
            .format(
                DateTimeFormatter.ofPattern(
                    "dd/MM/yyyy"
                )
            ));

return dataset;
}

private JFreeChart createChart(final CategoryDataset dataset,
    String country,
    int linesNum) {

    // create the chart...
    final JFreeChart chart = ChartFactory.createLineChart(
        country + ": διάγραμμα δεδομένων Covid", // chart title
        "Χρόνος", // domain axis label
        "Ποσότητα", // range axis label
        dataset, // data
        PlotOrientation.VERTICAL, // orientation
        true, // include legend
        true, // tooltips
        false // urls
    );
    chart.setBackgroundPaint(Color.white);

    final CategoryPlot plot = (CategoryPlot) chart.getPlot();
    plot.setBackgroundPaint(Color.lightGray);
    plot.setRangeGridlinePaint(Color.white);
    //Αξωνας X με τις ημερομηνίες
    final CategoryAxis catAxis = plot.getDomainAxis();
    //Τις δείχνει με γωνία 45 μοιρών
    catAxis.setCategoryLabelPositions(CategoryLabelPositions.DOWN_45);
    //Λόγω του μεγάλου πλήθους των ημερομηνιών δείχνουμε μόνο την πρώτη
    //την τελευταία και 3 ενδιάμεσες αν το εύρος το επιτρέπει
    for (int i=1; i<dataset.getColumnCount()-1; i++) {
        if (i == (int) (dataset.getColumnCount()*0.25) ||
            i == (int) (dataset.getColumnCount()*0.5) ||
            i == (int) (dataset.getColumnCount()*0.75) )
            continue;
        //Κρύβει τις υπόλοιπες ημερομηνίες
        catAxis.setTickLabelPaint(dataset.getColumnKey(i),
            new Color(0,0,0,0));
    }
}

```

```

    }
    // customise the range axis...
    final NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
    rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
    rangeAxis.setAutoRangeIncludesZero(true);

    // customise the renderer...
    final LineAndShapeRenderer renderer =
        (LineAndShapeRenderer) plot.getRenderer();
    //Φτιάχνει τις 3 γραμμές, σε πάχος και στηλ
    for(int i=0; i<linesNum; i++) {
        renderer.setSeriesStroke(
            i, new BasicStroke(
                3.0f, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND,
                1.0f, new float[] {1.0f}, 0.0f
            )
        );
    }
    return chart;
}
}

```

#### Κλάση HtmlBrowser,

για την παρουσίαση του χάρτη, επιλέχθηκε να χρησιμοποιηθεί ένα javafxPanel, το οποίο έχει μια webEngine και δίνει την δυνατότητα να προβάλλουμε ιστοσελίδες html, που περιέχουν javascript κώδικα. Λόγω του ότι το javafxPanel είναι στην ουσία μια ενσωματωμένη εφαρμογή JavaFx, δημιουργούμε ένα αντικείμενο αυτής της κλάσης, με την αρχικοποίηση του συστήματος και απλά προβάλλουμε το panel και κάνουμε ξανά load το link οπότε ο χρήστης επιλέξει να προβάλει έναν νέο χάρτη. Μεγάλο μέρος της δουλειάς γίνεται στην κλάση HtmlWriter που παρουσιάστηκε παραπάνω.

```

package views;

import java.awt.Dimension;
import java.io.File;
import javafx.application.Platform;
import javafx.embed.swing.JFXPanel;
import javafx.scene.Scene;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;

/**
 * //https://stackoverflow.com/a/26028556/15090628 source
 */
public class HtmlBrowser extends JFXPanel {
    private WebView webView;
    private WebEngine webEngine;
    private String link;
    private Dimension dimension;

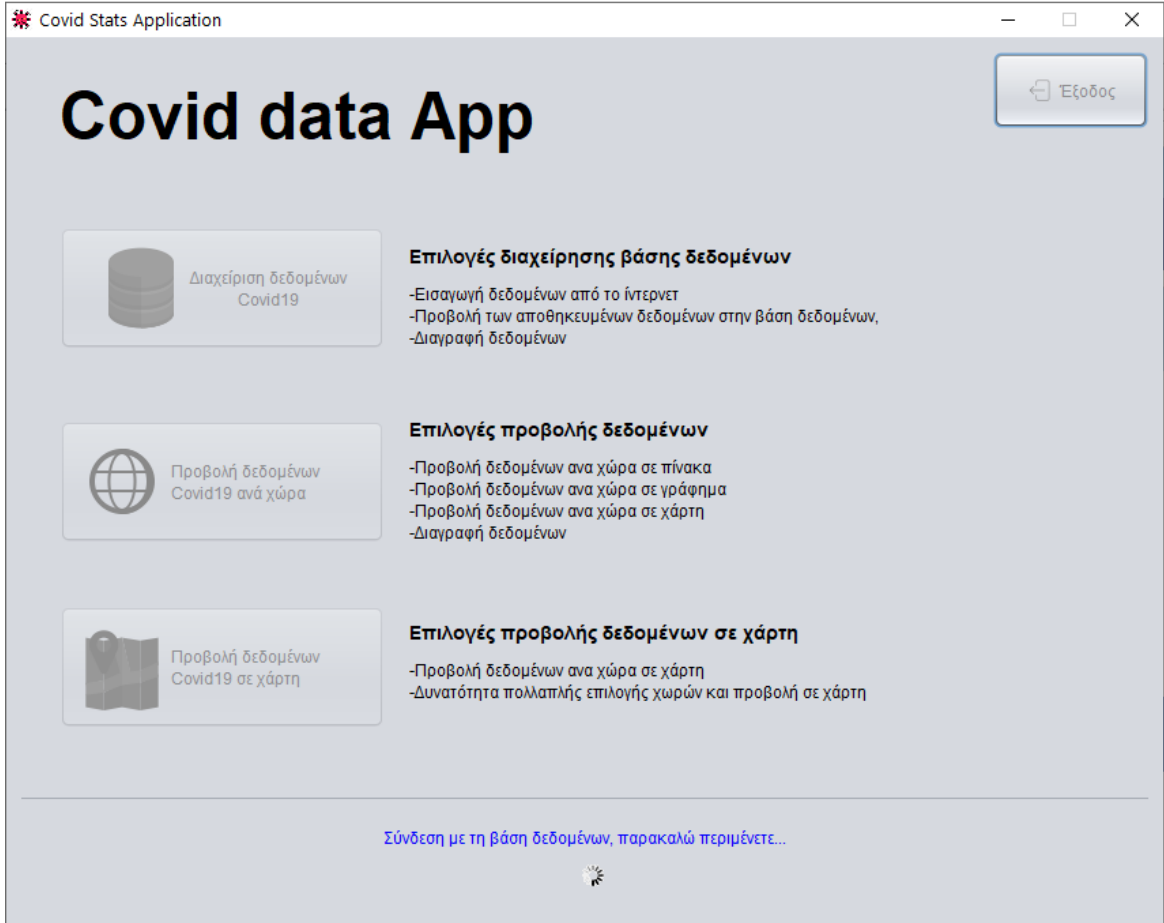
    public HtmlBrowser() {
        //Τρέχει σε Jfx thread
        Platform.runLater(() -> {

```

```
        initialiseJavaFXScene();  
    });  
}  
  
private void initialiseJavaFXScene() {  
    //Ορισμός μεγέθους  
    dimension = new Dimension(500, 400);  
    setPreferredSize(dimension);  
    //Το λινκ που θα δείχνει με την αρχικοποίηση  
    link = new File("html\\mappage.html").toURI().toString();  
    //Νέο webView για την προβολή ιστοσελίδων  
    webView = new WebView();  
    webEngine = webView.getEngine();  
    //Φορτώνει το λινκ  
    webEngine.load(link);  
    //Δημιουργεί ένα scene για να το τοποθετήσει στο βασικό scene του Jfx panel  
    Scene scene = new Scene(webView);  
    setScene(scene);  
}  
//Φορτώνει το λινκ που του στέλνουμε  
public void loadLink(String link, Dimension dimension) {  
    //this.link = link;  
    if(!this.dimension.equals(dimension)) {  
        this.dimension = dimension;  
        this.setSize(dimension);  
    }  
    webEngine.load(link);  
}  
}
```

### 3.3 Ερώτημα Δ – Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής

Παρακάτω η ροή της εφαρμογής και καθοδήγηση ανά εικόνα για την εκτέλεση της εφαρμογής .

Απαίτηση	Εικόνα
<b>R1</b>	 <p>Η αρχική οθόνη φόρτωσης. Εάν η σύνδεση με την βάση δεδομένων αποτύχει, βγάζει το αντίστοιχο μήνυμα. Μέχρι να ολοκληρωθεί η φόρτωση δεν αφήνει δυνατότητα επιλογής στον χρήστη και τα εικονίδια φαίνονται με γκριζο χρώμα.</p>

 Covid Stats Application

Εξοδος

# Covid data App



Διαχείριση δεδομένων  
Covid19

### Επιλογές διαχείρισης βάσης δεδομένων

- Εισαγωγή δεδομένων από το ίντερνετ
- Προβολή των αποθηκευμένων δεδομένων στην βάση δεδομένων,
- Διαγραφή δεδομένων



Προβολή δεδομένων  
Covid19 ανά χώρα

### Επιλογές προβολής δεδομένων

- Προβολή δεδομένων ανα χώρα σε πίνακα
- Προβολή δεδομένων ανα χώρα σε γράφημα
- Προβολή δεδομένων ανα χώρα σε χάρτη
- Διαγραφή δεδομένων



Προβολή δεδομένων  
Covid19 σε χάρτη

### Επιλογές προβολής δεδομένων σε χάρτη

- Προβολή δεδομένων ανα χώρα σε χάρτη
- Δυνατότητα πολλαπλής επιλογής χωρών και προβολή σε χάρτη

Η σύνδεση με την βάση δεδομένων ολοκληρώθηκε

Όλες οι επιλογές με βάση την προδιαγραφή έχουν εκτελεστεί επιτυχώς και η σύνδεση με την βάση πραγματοποιήθηκε. (4 επιλογές της εφαρμογής με πράσινο). Ο χρήστης πλέον μπορεί να επιλέξει την λειτουργία που επιθυμεί.

Στην συνέχεια θα αναλυθούν οι 3 οθόνες αναλόγως την επιλογή που θα γίνει στην βασική οθόνη.  
**(προδιαγραφές R2,R3,R4)**

ΤΜΗΜΑ ΗΛΕ48  
ΕΡΓΑΣΙΑ Νο 3

72

R2

Covid Stats Application
Κεντρικό Μενού

### Διαχείριση Βάσης Δεδομένων

Επιλέξτε το είδος δεδομένων για λήψη

Θάνατοι

Λήψη δεδομένων

Δεδομένα που έχουν αποθηκευτεί στη βάση δεδομένων

Αποτελέσματα αναζήτησης

Χώρες Αποθηκευμένες στη Βάση Δεδομένων

Afghanistan

Albania

Algeria

Andorra

Angola

Antigua\_and\_Barbuda

Argentina

Armenia

Διαγραφή Χώρων

Διαγραφή όλων των δεδομένων

Κάντε κλικ για να γίνει λήψη των δεδομένων από τον server  
Οι διαθέσιμες χώρες θα αποθηκευτούν αυτόματα

Εισαγωγή >>

Είδος	Έως
1	7 Mar 2021
2	12 Mar 2021
3	7 Mar 2021
1	4 Mar 2021
3	10 Φεβ 2021
2	17 Φεβ 2021
3	10 Φεβ 2021
3	10 Φεβ 2021
3	10 Φεβ 2021
2	4 Mar 2021
1	12 Mar 2021
3	10 Mar 2021

Διαγραφή δεδομένων

Πηγαίνετε το ποντίκι πάνω σε κάποιο κουμπί για να δείτε πληροφορίες

Δεν υπάρχουν δεδομένα για εισαγωγή, παρακαλώ πατήστε λήψη δεδομένων

Η οθόνη διαχείρισης δεδομένων, χωρίς να έχει γίνει λήψη δεδομένων ακόμη. Αρχικά οι πληροφορίες που παρέχονται είναι :

- στο κάτω μέρος οι χώρες που έχουν αποθηκευτεί στην βάση δεδομένων,
- και δεξιά τις τα δεδομένα των χωρών που είναι αποθηκευμένα. Για κάθε χώρα φαίνεται η κατηγορία που ανοίκει και επίσης μέχρι ποιά ημερομηνία που έχουν αποθηκευτεί τα δεδομένα.

Covid Stats Application

Κεντρικό  
Μενού

### Διαχείριση Βάσης Δεδομένων

Επιλέξτε το είδος δεδομένων για λήψη

Θάνατοι

Λήψη δεδομένων

Αποτελέσματα αναζήτησης

Afghanistan  
Albania  
Algeria  
Andorra  
Angola  
Antigua\_and\_Barbuda  
Argentina  
Armenia  
Australia

Χώρες Αποθηκευμένες στη Βάση Δεδομένων

Fiji  
Finland  
France  
Gabon  
Gambia  
Georgia  
Germany  
Ghana

Διαγραφή Χώρων

Διαγραφή όλων των δεδομένων

Εισαγωγή >>

Δεδομένα που έχουν αποθηκευτεί στη βάση δεδομένων

Όνομα Χώρας	Είδος	Έως
Andorra	1	7 Mar 2021
Andorra	2	12 Mar 2021
Andorra	3	7 Mar 2021
Angola	1	4 Mar 2021
Argentina	3	10 Φεβ 2021
Austria	2	17 Φεβ 2021
Azerbaijan	3	10 Φεβ 2021
Belize	3	10 Φεβ 2021
Bolivia	3	10 Φεβ 2021
Costa_Rica	2	4 Mar 2021
Cote_d'Ivoire	1	12 Mar 2021
Cote_d'Ivoire	2	12 Mar 2021

Διαγραφή δεδομένων

Δεν υπάρχουν νέες χώρες για αποθήκευση

Πατώντας εισαγωγή θα εισάγετε δεδομένα είδος: Θάνατοι

Για να εισάγετε άλλο είδος επιλέξτε από τη λίστα το είδος που επιθυμείτε και πατήστε λήψη δεδομένων ανά

Σε περίπτωση που δεν έχουμε δεδομένο για μια χώρα που μας ενδιαφέρει, κάνουμε νέα λήψη δεδομένων όλων των χωρών και μετά επιλέγουμε την χώρα που μας ενδιαφέρει για εισαγωγή των δεδομένων (π.χ. θάνατοι) στην βάση.

Τα πλήκτρα διαγραφή δεδομένων και χωρών συμπεριλαμβάνονται όπως περιγραφουν οι προδιαγραφές έργου.

R3

**Covid Stats Application**

**Προβολή Δεδομένων Covid**

Κεντρικό Μενού

Θάνατοι | Ασθενείς που έχουν ανακάμψει | Επιβεβαιωμένα κρούσματα

Επιλέξτε Χώρα: Greece

Από: 22/01/2020 | Έως: 19/02/2021

Αναζήτηση

Επιλογές διαγράμματος

☒ Θάνατοι

☒ Ανέκαμψαν

☒ Επιβεβ. κρούσματα

☐ Δεδομένα ημέρας

☒ Σωρευτικά δεδομένα

Προβολή σε διάγραμμα

Προβολή σε χάρτη

Διαγραφή δεδομένων

Ημερομηνία	Σωρευτικό πλήθος	Ημερήσιο πλήθος
22 Ιαν 2020	0	0
23 Ιαν 2020	0	0
24 Ιαν 2020	0	0
25 Ιαν 2020	0	0
26 Ιαν 2020	0	0
27 Ιαν 2020	0	0
28 Ιαν 2020	0	0
29 Ιαν 2020	0	0
30 Ιαν 2020	0	0
31 Ιαν 2020	0	0
1 Φεβ 2020	0	0
2 Φεβ 2020	0	0
3 Φεβ 2020	0	0
4 Φεβ 2020	0	0
5 Φεβ 2020	0	0
6 Φεβ 2020	0	0
7 Φεβ 2020	0	0
8 Φεβ 2020	0	0
9 Φεβ 2020	0	0
10 Φεβ 2020	0	0
11 Φεβ 2020	0	0
12 Φεβ 2020	0	0
13 Φεβ 2020	0	0
14 Φεβ 2020	0	0
15 Φεβ 2020	0	0
16 Φεβ 2020	0	0
17 Φεβ 2020	0	0
18 Φεβ 2020	0	0
19 Φεβ 2020	0	0
20 Φεβ 2020	0	0

Αρχικά επιλέγουμε την χώρα (της οποία προηγουμένως έχουμε κατεβάσει τουλάχιστον ένα τύπο δεδομένων) , αλλιώς δεν εμφανίζεται.

Όλα τα στοιχεία φαίνονται αριστερα ανάλογα με το tab που θα επιλέξουμε.

Υπάρχει δυνατότητα αλλαγής ημερομηνίας , για να εμφανίσουμε μόνο τα στοιχεία στο χρονικό διάστημα που μας ενδιαφέρει. Παρακάτω το ημερολόγιο που ανοίγει για επιλογή ημερομηνίας. Εχουν τεθει κανονες ώστε το ημερολόγιο να αφηνει επιλογη μόνο εγκυρων ημερομηνιών. (από 22/01/2020 μέχρι σήμερα) Οση ώρα είναι ανοιχτο το ημερολόγιο η υπολοιπη εφαρμογη μπαινει σε On hold status.

Επιλέξτε ημερομηνία

Mon	Tue	Wed	Thur	Fri	Sat	Sun
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

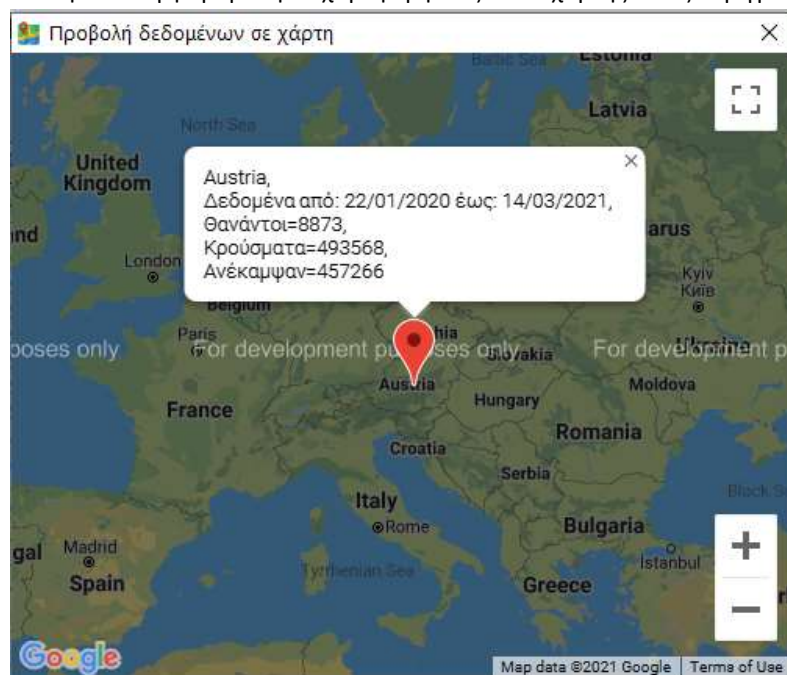
<< Previous JANUARY 2020 Next >>



Υπάρχουν έξτρα επιλογές για Δεδομένα ημέρας και Σωρευτικά δεδομένα. (με βάση της προδιαγραφές)  
Με την επιλογή για προβολή σε διάγραμμα ανοίγει το παρακάτω διάγραμμα.



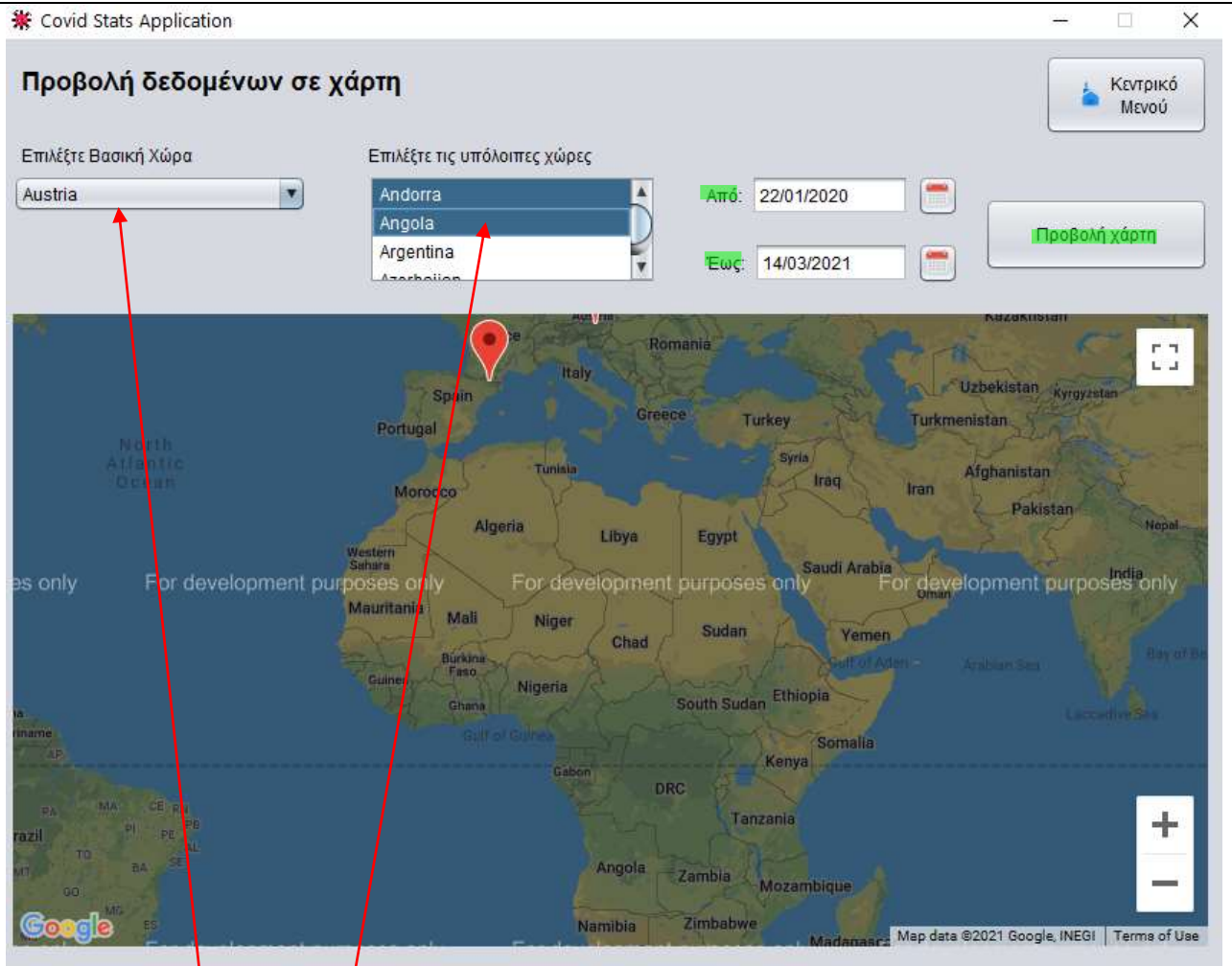
Με την επιλογή προβολή σε χάρτη εμφανίζεται ο χάρτης όπως περιγράφουν οι προδιαγραφές



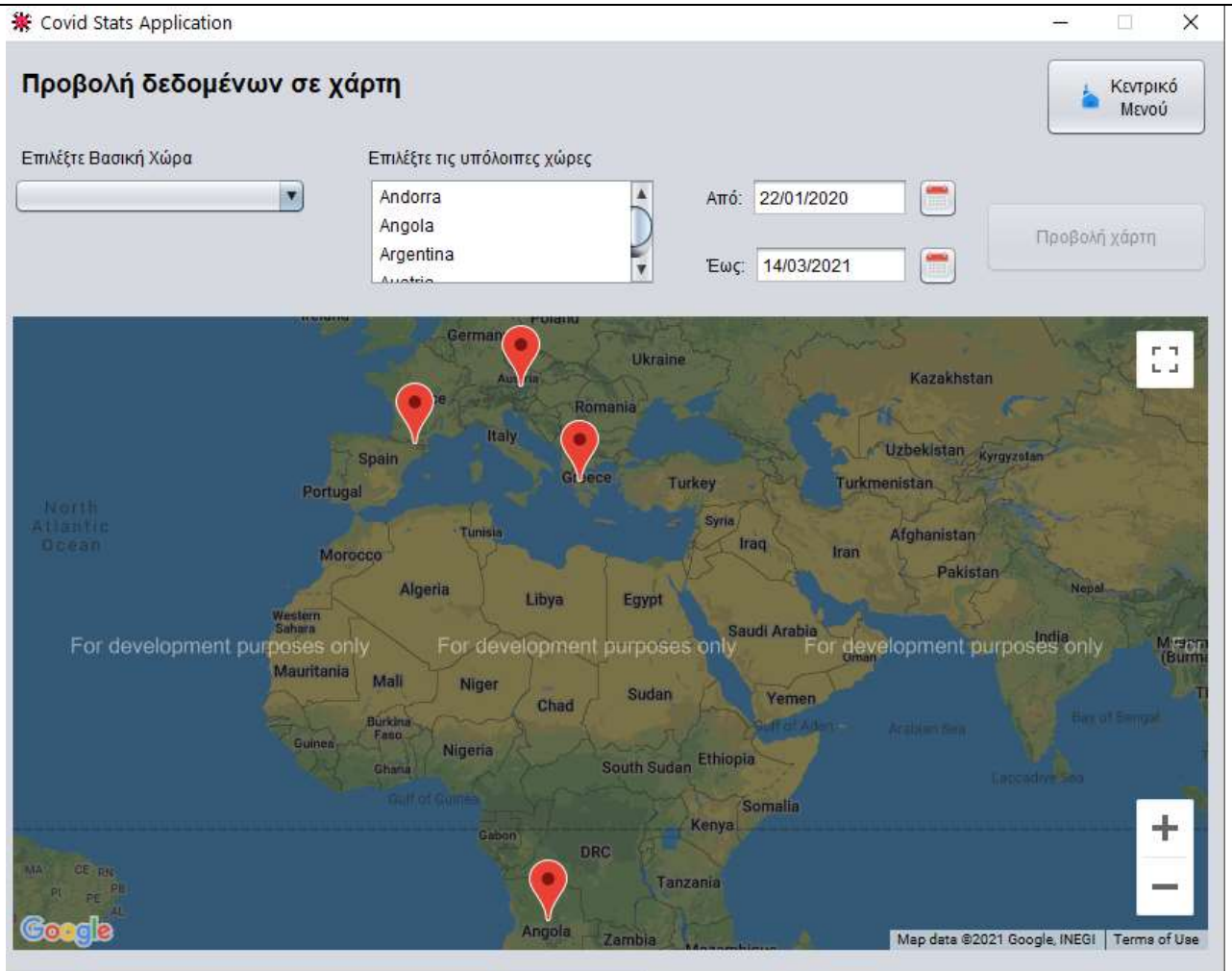
Τέλος υπάρχει και το πλήκτρο διαγραφής δεδομένων



R4



Αρχικά επιλέγουμε μία βασική χώρα από το πεδίο πολλαπλής επιλογής  
Επιλέγουμε και τις άλλες χώρες που μας ενδιαφέρουν  
Επιλέγουμε ημερομηνία και πατάμε το πλήκτρο προβολή χάρτη



Και τέλος βλέπουμε το αποτέλεσμα μετά την επιλογή της Αυστρίας ως βασική χώρα και Ελλάδας, Ανδόρα, Αγκόλα ως λοιπές χώρες.

Ο χάρτης υπό φυσιολογικές συνθήκες εστιάζει στην Αυστρία, αλλά στο κάτω παράδειγμα έχει γίνει zoom out και στις 3 για λόγους καλής συγγραφής.

## 4 ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ

[Κάντε έναν αναλυτικό και κριτικό απολογισμό του έργου αναφέροντας

- 1) τα προβλήματα που αντιμετωπίσατε, τι θα αλλάζατε στο έργο αν είχατε τη δυνατότητα,
- 2) τις αποκλίσεις που είχατε στις επαναλήψεις σε σχέση με αυτά που ορίζονται στην εργασία,
- 3) το χρόνο που απαιτήθηκε για την κάθε δραστηριότητα με βάση τα στοιχεία που κρατήσατε και σε σχέση με το χρόνο που προϋπολογίσατε,
- 4) πως αξιολογείτε το εργαλείο συνεργασίας trello και με ποιο τρόπο σας βοήθησε στην καθημερινή σας εργασία
- 5) το πιο πολύτιμο μέλος της ομάδας σας και τον τρόπο που το επιλέξατε,
- 6) τους κινδύνους που πραγματώθηκαν, καθώς και πώς αποκριθήκατε σε αυτούς,
- 7) τι θα αλλάζατε σε ένα επόμενο αντίστοιχο έργο, καθώς και
- 8) ποια ήταν τα θετικά σημεία που αποκομίσατε από αυτή την εργασία].

### 1) Προβλήματα

Κατά την διάρκεια του έργου η ομάδα μας ήρθε αντιμέτωπη με αρκετά και άγνωστα για εμάς προβλήματα τα οποία χάρη στην καλή συνεργασία που υπήρχε και το ομαδικό πνεύμα που διακατείχε την όλη προσπάθεια ξεπεράστηκαν ομαλά.

Τέτοια ήταν ενδεικτικά τα κάτωθι:

- SCRUM METHOD  
Κανένα μέλος της ομάδας δεν είχε πρότερη εμπειρία αυτής της μεθόδου Διαχείρισης Έργου γεγονός το οποίο αρχικά δημιούργησε ένα πρόσθετο άγχος στην όλη διαδικασία.
- Έλλειψη χρόνου  
Το όλο Project ήταν απαιτητικό και απαιτούσε να αφιερώσουμε αρκετό χρόνο όχι μόνο για την υλοποίηση της εφαρμογής αλλά και για την συγγραφή της εργασίας. Έχοντας ούτως ή άλλως όλοι περιορισμένο ελεύθερο χρόνο στην διάθεσή μας καθημερινά η κατάσταση έγινε ακόμα χειρότερη όταν διαπιστώσαμε ότι οι χρόνοι αυτοί έπρεπε και να συγχρονιστούν για να μπορέσουμε να συνεργαστούμε καθώς η εργασία ήταν ομαδική και προϋπέθετε την συνεννόηση μεταξύ των μελών.
- Σχεδίαση  
Η ήδη αποκτηθείσα γνώση για την γλώσσα JAVA και η σχετική εμπειρία της ομάδας σε εργαλεία όπως Net-Beans και VisualParadigm δεν ήταν αρκετή για να ανταπεξέλθουμε στις απαιτήσεις του Project όπως γραφικά διασύνδεσης χρήστη GUI γεγονός που μας έθεσε ένα ακόμα εμπόδιο προκειμένου να αρχίσουμε άμεσα την υλοποίηση του έργου.
- Ανάθεση απαιτήσεων  
Ήταν κάτι το οποίο δεν το αναμέναμε αλλά τελικά το αντιμετωπίσαμε σαν πρόβλημα όταν κληθήκαμε να μοιράσουμε τις απαιτήσεις που θα έκανε κάθε μέλος. Πολλές από τις απαιτήσεις δεν γνωρίζαμε λόγω έλλειψης πρότερης εμπειρίας πόσο χρόνο θα χρειαστούν για να υλοποιηθούν και κατά συνέπεια δυσκολευτήκαμε πολύ να αποφασίσουμε ποιος θα κάνει τί.
- Ομάδα  
Προσπαθήσαμε να προσομοιώσουμε μία πραγματική ομάδα ανάπτυξης εφαρμογής αλλά παρόλα αυτά η πραγματικότητα παρέμενε η ίδια. Ήμασταν μία ομάδα φοιτητών που κύριο μέλημά της ήταν η κατανόηση νέων τεχνολογιών και η άμεση εφαρμογή τους σε ένα νέο έργο προς υλοποίηση.

Κατόπιν των ανωτέρω και προκειμένου να έχουν ξεπεραστεί ήδη αυτά τα προβλήματα θα μπορούσε να γίνει μία εργασία πιο πριν στην οποία θα είχαν ήδη μελετηθεί οι τεχνολογίες που απαιτούνταν στην παρούσα ούτως ώστε τώρα να εξετάζονταν καθαρά η ομαδικότητα και η καλή συνεργασία της ομάδας βάζοντας τους φοιτητές να δουν πως δουλεύει μία ομάδα ανάπτυξης λογισμικού στην πράξη με δεδομένο όμως ότι όλα τα μέλη ξεκινούν από την ίδια αφετηρία γνώσεων (από την προηγούμενη εργασία).

## 2) Αποκλίσεις

Η ομάδα δεν παρουσίασε πολλές αποκλίσεις ως προς το αρχικό πλάνο διαχείρισης έργου που αποφασίστηκε από την αρχή παρά μόνο στα 3 ακόλουθα έργα

- R3 Διαμόρφωση GUI, για λειτουργία επιλογής χώρας
- R3 Λειτουργίες Presenter για το GUI R3 (Πλήκτρων προβολών και εμφάνιση χάρτη και Chart)
- R4.1 Διαμόρφωση GUI για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χωρών

τα οποία και μεταφέρθηκαν από το Sprint 1 στο Sprint 2 όπου και υλοποιήθηκαν

## 3) Χρόνος

Ο χρόνος καθώς και η διάρκεια των ημερών που απαιτήθηκε για κάθε δραστηριότητα σύμφωνα με τα στοιχεία που κρατήσαμε φαίνεται στον πίνακα που ακολουθεί

A/A	ΑΠΑΙΤΗΣΗ	ΕΝΑΡΞΗ	ΠΕΡΑΣ	ΔΙΑΡΚΕΙΑ ΗΜΕΡΩΝ
1	Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση	03/02	03/02	1
2	Υπολογισμός των προτεραιοτήτων	03/02	03/02	1
3	Οργάνωση ομάδας και αναθέσεις αρμοδιοτήτων	03/02	03/02	1
4	Δημιουργία ενός αρχικού διαγράμματος κλάσεων	03/02	03/02	1
5	Δημιουργία του βασικού Frame με τις μεθόδους για την προβολή των GUI του συστήματος	03/02	04/02	2
6	Δημιουργία κλάσεων Presenter για τα GUI	04/02	05/02	2
7	Δημιουργία του GUI_R1	04/02	05/02	2
8	Δημιουργία κλάσης για την διαχείριση της Βάσης Δεδομένων	05/02	06/02	2
9	Δημιουργία κλάσης για την επικοινωνία με τον απομακρυσμένο Server	06/02	07/02	2
10	Δημιουργία GUI για την προδιαγραφή R2	06/02	07/02	2
11	Λειτουργίες Presenter για το Data Manage View	08/02	12/02	5
12	Debug παραδοτέου 1ου Sprint	13/02	14/02	2
13	Διαμόρφωση GUI , για λειτουργία επιλογής χώρας	15/02	18/02	4



14	Λειτουργία Πλήκτρου «Διαγραφή δεδομένων» (λειτουργικότητα)	25/02	25/02	1
15	Λειτουργίες Presenter για το GUI R3 (Πλήκτρων προβολών και εμφάνιση χάρτη και Chart)	18/02	19/02	6
		22/02	25/02	
16	Ενσωμάτωση ημερολογίου σε GUI	25/02	25/02	1
17	Διαμόρφωση GUI για λειτουργία επιλογής βασικής χώρας και πολλαπλής επιλογής χωρών	26/02	27/02	2
18	Λειτουργίες Presenter για το GUI R4 - Χάρτη	27/02	27/02	1
19	Debug παραδοτέου 2ου Sprint	27/02	28/02	2
20	Εργασία 1.Εισαγωγή	06/03	08/03	3
21	Εργασία 2.5 Οργάνωση ομάδας και αναθέσεις αρμοδιοτήτων	10/03	12/03	2
22	Εργασία 2.7 Χρήση εργαλείου trello	12/03	12/03	1
23	Εργασία 3.1 Ερώτημα Α– Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java	12/03	13/03	2
24	Εργασία 2.1 Υπολογισμός της απαιτούμενης προσπάθειας ανά απαίτηση	11/03	14/03	3
25	Εργασία 2.2 Υπολογισμός των προτεραιοτήτων	09/03	14/03	4
26	Εργασία 2.3 Χρονοδιάγραμμα του έργου	11/03	15/03	4
27	Εργασία 2.4 Το product backlog	10/03	16/03	5
28	Εργασία 3.4 Ερώτημα Δ – Συνολικός Έλεγχος και Εκτέλεση της Εφαρμογής	13/03	15/03	2
29	Εργασία 3.2 Ερώτημα Β – Δημιουργία GUI Εφαρμογής (συγγραφή)	15/03	15/03	1
30	Εργασία 3.3 Ερώτημα Γ – Παρουσίαση Χάρτη και Γραφημάτων	15/03	15/03	1
31	ΑΝΑΦΟΡΕΣ- ΒΙΒΛΙΟΓΡΑΦΙΑ	15/03	15/03	1
32	Debug παραδοτέου 3ου Sprint	08/03	12/03	3
33	Εργασία 4. ΚΡΙΤΙΚΟΣ ΑΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΡΓΟΥ	15/03	16/03	2
34	Εργασία 2.6 Παρακολούθηση της προσπάθειας κατά τη διάρκεια του έργου	09/03	16/03	4

#### 4) Αξιολόγηση Trello

Το Trello είναι ένα καταπληκτικό εργαλείο Διαχείρισης Έργων, ιδιαίτερα στην περίπτωσή μας που η ομάδα δούλεψε εξ αποστάσεων και ασύγχρονα, που οργανώνει τις απαιτήσεις που προκύπτουν σε απλά και κατανοητά βήματα με

τρόπο απτό και ξεκάθαρο. Μέσω αυτού διαχωρίσαμε το έργο σε κομμάτια και αυτό βοήθησε όλα τα μέλη της ομάδας στην άμεση οργάνωση του Project σε βήματα που έγιναν κατανοητά με μια ματιά.

Μας βοήθησε στην καθημερινή εργασία οργανώνοντας και παρακολουθώντας εύκολα την εξέλιξη του Project μέσα από δυνατότητες που έχει όπως τα δικαιώματα που δόθηκαν στα μέλη της ομάδας προκειμένου να έχουν πρόσβαση σε πίνακες καθώς και σχόλια σε κάρτες τα οποία οδήγησαν σε περαιτέρω συζητήσεις και αναλύσεις επί των απαιτήσεων που έγιναν.

### 5) Πολυτιμότερο μέλος

Χωρίς επιλογή με κάποιον συγκεκριμένο τρόπο αναμφισβήτητα πολυτιμότερο μέλος αναδείχθηκε ο Scrum Master (Μπέρκοβιτς-Ιωαννίδης Βασίλειος).

Οι υποδείξεις του ήταν καθοριστικές και οι λύσεις που έδωσε στην υλοποίηση του κώδικα (τόσο εντός της ομάδας όσο και μέσω του forum σε φοιτητές άλλων τμημάτων) σίγουρα αποτέλεσαν σημαντικό παράγοντα στο να ολοκληρωθεί και τελικά να παραδοθεί αυτή η εργασία.

### 6) Κίνδυνοι

Οι βασικότεροι κίνδυνοι που αντιμετωπίσαμε ήταν οι εξής

- Ανεπάρκεια σωστής διαχείρισης του έργου λόγω απειρίας προγραμματισμού και επιλογής στόχων καθώς και λόγω δυσκολίας στην συχνή επικοινωνία εξαιτίας οικογενειακών και επαγγελματικών υποχρεώσεων των μελών.
- Ανεπαρκής γνώση εξαιτίας ελλιπούς κατανόησης των νέων εργαλείων και τεχνικών καθώς και ανεπαρκούς πρακτικής εμπειρίας στο συγκεκριμένο αντικείμενο.

Παρόλα αυτά η ομάδα κατάφερε και αντιμετώπισε με επιτυχία τους ανωτέρω κινδύνους με

- Έρευνα και Παραδοχή όπου η ομάδα παραδέχτηκε τις αδυναμίες της και μέσω έρευνας έγιναν προσπάθειες ώστε αυτές να ξεπεραστούν
- Ανάληψη κινδύνου οπότε η ομάδα ανέπτυξε διαδικασίες ελέγχου (π.χ Debug) ώστε να περιοριστεί ο κίνδυνος όσο το δυνατόν σε αποδεκτά επίπεδα.

### 7) Τι θα αλλάζαμε

Έχοντας κατά νου όλα τα παραπάνω η ομάδα διαπίστωσε ότι σε ένα επόμενο αντίστοιχο έργο θα έδινε ακόμη μεγαλύτερη προσοχή στην οργάνωση των απαιτήσεων, πιο συγκεκριμένα ίσως χρειαζόταν μια πιο λεπτομερή ανάλυση, και κάποιες από τις εργασίες θα έπρεπε να σπάσουν σε μικρότερες για να είναι πιο εύκολα διαχειρίσιμες.

Επίσης εάν είχαμε την δυνατότητα για πιο συχνή επικοινωνία των μελών, η ιδέα μιας 10λεπτης πρωινής συνάντησης θα βοηθούσε στην καλύτερη επικοινωνία και άμεση επίλυση ζητημάτων. Δυστυχώς οι υποχρεώσεις που όλοι είχαμε δεν κατέστησαν κάτι τέτοιο δυνατόν.

### 8) Θετικά σημεία

Η όλη διαδικασία υπήρξε μία πολύ ευχάριστη και χρήσιμη εμπειρία την οποία μπορεί κανείς να συναντήσει σε περιβάλλοντα εργασίας μεγάλων ομάδων οι οποίες αναπτύσσουν μεγαλύτερα Project.

Σίγουρα πάντως στα θετικά σημεία δεν μπορούμε παρά να σημειώσουμε ότι υπήρξαν η ανάθεση έργου σε κομμάτια Scrum, ο διαμοιρασμός του κώδικα μέσω github, οι μέθοδοι αξιολόγησης των απαιτήσεων Planning Poker και επίσης η εργασία μας μέσω Microsoft Teams. Συγκεκριμένα για την συγγραφή βοήθησε πολύ στην παράλληλη εργασία και των τεσσάρων με το online Word.

Τέλος το πιο σημαντικό είναι ότι όλη η ομάδα συνεργάστηκε και ξεπέρασε τα τυχόν προβλήματα που προέκυψαν με υπομονή και προσωπική προσπάθεια καταφέροντας τελικά να ολοκληρώσει και να παραδώσει μία άρτια εργασία πληρώνοντας όλες τις προδιαγραφές που τέθηκαν από την αρχή.



## 5 ΑΝΑΦΟΡΕΣ

### ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] Βασίλειος Βεσκούκης, Τεχνολογία Λογισμικού II, Τόμος Β, Ά Έκδοση, ΕΑΠ, Πάτρα 2001
- [2] Κλεάνθης Θραμπουλίδης, Γλώσσες Προγραμματισμού II, Τόμος Γ, ΕΑΠ, Πάτρα 2001
- [3] Φιτσιλής Πάνος, Ιωάννης Σταμέλος, Προγραμματισμός Έργων Πληροφορικής- Αντικειμενοστρεφείς Μεθοδολογίες, Τόμος Δ, ΕΑΠ, Πάτρα 2008
- [4] Paul Deitel- Harvey Deitel, Java Προγραμματισμός, 10<sup>η</sup> Έκδοση, Εκδόσεις Μ. Γκιούρδας, 2015
- [5] Εκπαιδευτική πύλη ΕΑΠ, Συνοδευτικό Εκπαιδευτικό Υλικό
- [6] Εκπαιδευτική πύλη ΕΑΠ, Γραπτές Εργασίες παρελθόντων ετών
- [7] Εκπαιδευτική πύλη ΕΑΠ, Υλικό ΟΣΣ

### ΧΡΗΣΙΜΑ ΕΡΓΑΛΕΙΑ

- [1] Microsoft office Professional Plus 2016, Microsoft Word 2016, Microsoft Corporation
- [2] Microsoft office Professional Plus 2016, Microsoft Exel 2016, Microsoft Corporation
- [3] NetBeans IDE8.2
- [4] Visual Paradigm 16.2, Visual Paradigm International, 1999-2020
- [5] Trello agile boards, <https://trello.com/>

### ΔΙΑΔΙΚΤΥΑΚΕΣ ΠΗΓΕΣ

- [1] Java SE Documentation, <https://docs.oracle.com/javase/7/docs/api/overview-summary.html>
- [2] How to change look and feel, <https://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html#programmatic>
- [3] Convert java util Date to java LocalDate, <https://stackoverflow.com/a/48429952/15090628>
- [4] Gson Library documentation, <https://github.com/google/gson>
- [5] How to change JOptionPane buttons text, <https://stackoverflow.com/a/1399523/15090628>
- [6] JavaFX WebView in a Swing application, <https://stackoverflow.com/a/26028556/15090628>
- [7] How to write a custom typeAdapter, <https://stackoverflow.com/a/25947499/15090628>
- [8] Date picker using java Swing, <https://www.roseindia.net/tutorial/java/swing/datePicker.html>
- [9] Simple MVP Example, <https://riptutorial.com/swing/example/14137/simple-mvp-example>
- [10] Icon finder, <https://www.iconfinder.com/>
- [11] Loading gif generator, <http://www.ajaxload.info/>
- [12] Swing Worker Class, <https://docs.oracle.com/javase/7/docs/api/javax/swing/SwingWorker.html>