ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Πτυχιακή Εργασία

Δημιουργία Portal Εργαζομένων σε Roster

Έκτορας Σιμιστήρας
201187

Επιβλέπων:

Βασίλης Πουλόπουλος
Επίκουρος Καθηγητής

Σπάρτη, Οκτώβρης 2021

Περιεχόμενα

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

"Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας."

Σιμιστήρας Έκτορας Παναγιώτης

10 / 11 / 2021

Ευχαριστίες

Θα ήθελα να ευχαριστώ θερμά τον συνάδελφο που ανέπτυξε και «ανέβασε» το service που χρησιμοποιήθηκε στο project δίνοντας μου την ευκαιρία να δημιουργήσω ένα όσο το δυνατόν πιο ρεαλιστικό client για την εφαρμογή.

Θα ήθελα επίσης να ευχαριστήσω τον κ. Βασίλη Πουλόπουλο για την καθοδήγηση και την επίβλεψη που μου παρείχε κατά την διάρκεια της ανάπτυξης του project.

Τέλος οφείλω να απευθύνω ένα μεγάλο ευχαριστώ στους καθηγητές που με στηρίξαν και συμβούλεψαν κατά την διάρκεια των σπουδών μου.

Εισαγωγή

Η Πτυχιακή εργασία με τίτλο «Βάρδιες Εργαζομένων Σε Roster» είναι ένα Mobile First, SPA (Single Page Application) που χρησιμοποιείται για την **προβολή** και **διαχείριση** του προγράμματος εργασίας πολλαπλών χρηστών από οποιαδήποτε συσκευή (Smartphone, Tablet, Laptop, Desktop). Οι χρήστες επισκέπτονται την ιστοσελίδα της επιχείρησης που εργάζονται **(e-port)** και από εκεί αιτούνται να συνδεθούν. Αφού η ταυτότητα τους επιβεβαιωθεί από το σύστημα αποκτούν πρόσβαση στο portal που τους επιτρέπει να προβάλλουν, εκτυπώσουν , παραμετροποιήσουν το πρόγραμμα τους ή να επισκεφτούν τον εξωτερικό σύνδεσμο που τους οδηγεί σε ένα πλήρες διαφημιστικό documentation που αφορά τις λειτουργίες ενός συστήματος roster.

Σκοπός της εργασίας είναι να αξιοποιήσουμε τις πλέον σύγχρονες τεχνολογίες προγραμματισμού και να χρησιμοποιήσουμε μια ορθή τεχνική ανάπτυξης που θα ήταν κατάλληλη για την δόμηση οποιουδήποτε project, ανεξαρτήτως μεγέθους. Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε η αρχιτεκτονική προγραμματισμού **MV-VM** (Model View, View Model), Οι γλώσσες προγραμματισμού HTML, Javascript, CSS / SASS-CSS, C#, SQL τα frameworks Bootstrap, Knockout.js καθώς και η βιβλιοθήκη JQUERY. Τέλος για την υποστήριξη της αρχιτεκτονικής που επιλέχθηκε ήταν απαραίτητη η χρήση του WEBPACK για την υποστήριξη των imports (Javascript & HTML), την σωστή ιεραρχία των dependencies και το «χτίσιμο» των output αρχείων της Javascript που τελικά συμπεριλαμβάνουμε στην HTML.

Συστήματα Roster

Περιγραφή

Τα roster συστήματα έχουν σαν στόχο την οργάνωση υπάρχοντος ανθρώπινου δυναμικού, ως προς την εργασία, σε δεδομένο επιχειρησιακό περιβάλλον. Δεδομένου δε ότι τα επιχειρησιακά περιβάλλοντα διαφοροποιούνται σημαντικά μεταξύ τους, σε μία σειρά παραγόντων, έχουν ανάγκη να εφαρμόζουν διαφορετικούς τρόπους οργάνωσης του ΑΔ.
Η κάλυψη από τα roster συστήματα, κατά το δυνατόν, των περισσοτέρων χαρακτηριστικών αυξάνει τον βαθμό πολυπλοκότητας και συνθετότητας αυτών. Έτσι προκύπτουν συστήματα roster που έχουν τη δυνατότητα να εξυπηρετούν, εν γένει, σταθερό εργασιακό περιβάλλον έως και τα πλέον σύνθετα, που έχουν στόχο την δυναμική προσαρμογή του ΑΔ σε μια διαρκώς μεταβαλλόμενη ζήτηση.

Υποστηριζόμενοι Τύποι εργαζομένων.

Οι τύποι εργαζομένων, που μπορεί να υποστηρίζονται από τα roster συστήματα, είναι:
Σταθερού / Κλιμακωτού Ωραρίου: ο εργαζόμενος έχει σταθερό ωράριο στη διάρκεια της οριζόμενης περιόδου από την σύμβαση. Κύρια χαρακτηριστικά:

- Εργάζεται καθορισμένες ημέρες τις εβδομάδος, όσες ορίζονται στη σύμβαση, συνήθως Δευτέρα - Παρασκευή.

- Έχει σταθερή βάρδια, Πρωινή ή Απογευματινή. Οπωσδήποτε δεν εργάζεται σε Νυχτερινή βάρδια.
- Στο Κλιμακωτό ωράριο, ένας εργαζόμενος ή μια υποομάδα μπορεί να έχει διαφορετική ώρα έναρξης στην ίδια βάρδια

Κυλιόμενου Ωραρίου: ο εργαζόμενος έχει κυλιόμενο ωράριο στη διάρκεια της οριζόμενης περιόδου από την σύμβαση. Κύρια χαρακτηριστικά:

- Εργάζεται όσες ημέρες ορίζονται από τη σύμβαση, στο διάστημα από Δευτέρα - Κυριακή.
- Εναλλάσσεται στις λειτουργούσες βάρδιες, βάσει υπάρχουσας κατανομής.

Ανεξάρτητος εργαζόμενος: ο εργαζόμενος έχει σταθερό ή κυλιόμενο ωράριο, με αντίστοιχα χαρακτηριστικά, καλούμενος προς εργασία μέσω υπεργολάβου.

Λειτουργικοί Παράγοντες

Μερικοί από τους σημαντικούς παράγοντες, που επηρεάζουν την οργάνωση της εργασίας, μπορεί να είναι:

- Η λειτουργίας της επιχείρησης ανά εβδομάδα: 5-θήμερη, 6-ήμερη, 7-ήμερη ή άλλη.
- Η λειτουργία βαρδιών: αριθμός βαρδιών ανά 24-ωρο
- Η διεκπεραίωση σταθερού ή κυμαινόμενου φορτίου εργασίας
- Οι δεξιότητες των εργαζομένων: διαθέτουν μία η περισσότερες εξειδικεύσεις
- Η επιχειρησιακή διάρθρωση: αξιοποίηση ΑΔ σε διαφορετικά εργασιακά κέντρα.

Περίοδος λειτουργίας

Ο αριθμός των ημερών λειτουργίας ανά εβδομάδα, είναι ένας από τους βασικούς παράγοντες ενός roster συστήματος. Σε συνδυασμό με τα υποστηριζόμενα μοντέλα εργασίας, πλήρους απασχόλησης ή εκ περιτροπής, και τη λειτουργία ή μη βαρδιών, καθορίζουν **το πρόγραμμα εργασίας** του εργαζόμενου. Το πρόγραμμα εργασίας θα πρέπει να καλύπτει, τουλάχιστον, μία αρχική σταθερή ζήτηση για τις ημέρες λειτουργίας ανά εβδομάδα, με βάση τις συμβάσεις των εργαζομένων.

Εργασία σε βάρδιες

Η λειτουργία σε βάρδιες, είναι σημαντικός παράγοντας για την κατασκευή του προγράμματος εργασίας των εργαζομένων, θα καλύπτει την ζητούμενη κατανομή στις ζώνες εργασίας, πχ. 4/2/1, που σημαίνει 4 εργαζόμενοι στην Πρωινή βάρδια, 2 εργαζόμενοι στην Απογευματινή βάρδια και 1 εργαζόμενος στην Νυχτερινή βάρδια. Προφανώς, το πρόγραμμα εργασίας για την παραπάνω κατανομή, ικανοποιεί και οποιαδήποτε με ακέραιο πολλαπλασιαστή, δηλ. 8/4/2, 12/6/3 κοκ.
Επιπλέον, το πρόγραμμα εργασίας θα πρέπει να συμμορφώνεται με τις ισχύουσες νομικές δεσμεύσεις, όπως: ομαλή διαδοχή βαρδιών (χωρίς κόντρα βάρδια), ελεύθερη (ρεπό) Κυριακή, τουλάχιστον, μία ανά 7 εβδομάδες, ελάχιστη απόσταση μεταξύ διαδοχικών βαρδιών μεγαλύτερη των 11 ωρών, κλπ. Πρόσθετα, το πρόγραμμα εργασίας θα πρέπει να εξασφαλίζει την ισότητα των εργαζομένων που συμμετέχουν στην ομάδα. Αυτό σημαίνει ότι όλοι οι εργαζόμενοι της ομάδας, στη διάρκεια συγκεκριμένου χρονικού διαστήματος, που είναι η περίοδος του προγράμματος, έχουν κάνει τον ίδιο αριθμό βαρδιών σε κάθε ζώνη.

Φόρτος Εργασίας

Σε μία επιχείρηση ή σε ένα τμήμα της, ο φόρτος εργασίας που αντιμετωπίζουν μπορεί να είναι σταθερός ή να μεταβάλλεται σε επίπεδο εβδομάδας, ημέρας ή ακόμα και βάρδιας. Ο σταθερός φόρτος εργασίας καλύπτεται με εργαζόμενους που είναι σε Σταθερό / Κλιμακωτό Ωράριο ή σε Κυλιόμενο

Ωράριο με σταθερή αναλογία εργαζομένων ανά βάρδια (Σταθερή Ζήτηση). Για την αντιμετώπιση μεταβαλλόμενου φόρτου εργασίας θα πρέπει να προσδιορίζεται η εκάστοτε ζήτηση και μάλιστα ανεξάρτητα για κάθε υπάρχουσα εξειδίκευση (Κυμαινόμενη Ζήτηση). Η κυμαινόμενη ζήτηση είναι δυνατόν να μεταβάλλει τον αρχικό προγραμματισμό των εργαζομένων, πχ κλήση από ρεπό, τροποποίηση της ώρας έναρξης κ.λπ.
Ο προγραμματισμός των παραπάνω μεταβολών, απαιτεί την γρήγορη και όσο το δυνατόν αυτοματοποιημένη επικοινωνία του εκάστοτε υπεύθυνου με τον εργαζόμενο. Βοηθητικά για την κάλυψη της κυμαινόμενης ζήτησης είναι:

- Χρήση κινητού για την ενημέρωση του εργαζόμενου με το τρέχον πρόγραμμα εργασίας του και τις άδειες του
- Άμεση ενημέρωση του εργαζόμενου σχετικά με τυχόν αλλαγές (SMS, Viber, κλπ.)
- Χρήση portal για δήλωση διαθεσιμότητας και γενικότερα αυτοματοποιημένη διαχείριση αιτημάτων.

Δεξιότητες εργαζομένων

Ένα πρόσθετο χαρακτηριστικό των roster συστημάτων είναι η δυνατότητα να διαχειρίζονται περισσότερες από μία εξειδικεύσεις που τυχόν διαθέτει ο εργαζόμενος. Το χαρακτηριστικό αυτό δίνει τη δυνατότητα βέλτιστης αξιοποίησης του, κάθε φορά διαθέσιμου, ανθρώπινου δυναμικού. Συγκεκριμένα, επιτρέπει την ικανοποίηση της ζήτησης με τις λιγότερες δυνατές μεταβολές.

Ροές εργαζομένων

Ορισμένες φορές, εργαζόμενοι που ανήκουν σε διαφορετικά εργασιακά κέντρα της επιχείρησης, μπορεί να έχουν κοινές εξειδικεύσεις / δεξιότητες. Υπάρχουν περιπτώσεις, είτε λόγω εποχικότητας ή λόγω συγκυρίας, οι εργαζόμενοι του ενός εργασιακού κέντρου να είναι σε περίσσεια και ενός άλλου να είναι σε έλλειψη. Όταν εμφανίζεται τέτοια περίπτωση είναι επιθυμητό να υπάρχει δυνατότητα “ροής” από το εργασιακό κέντρο, που έχει περίσσεια, προς το εργασιακό κέντρο, που έχει έλλειψη.

Ανάλυση Συστημάτων

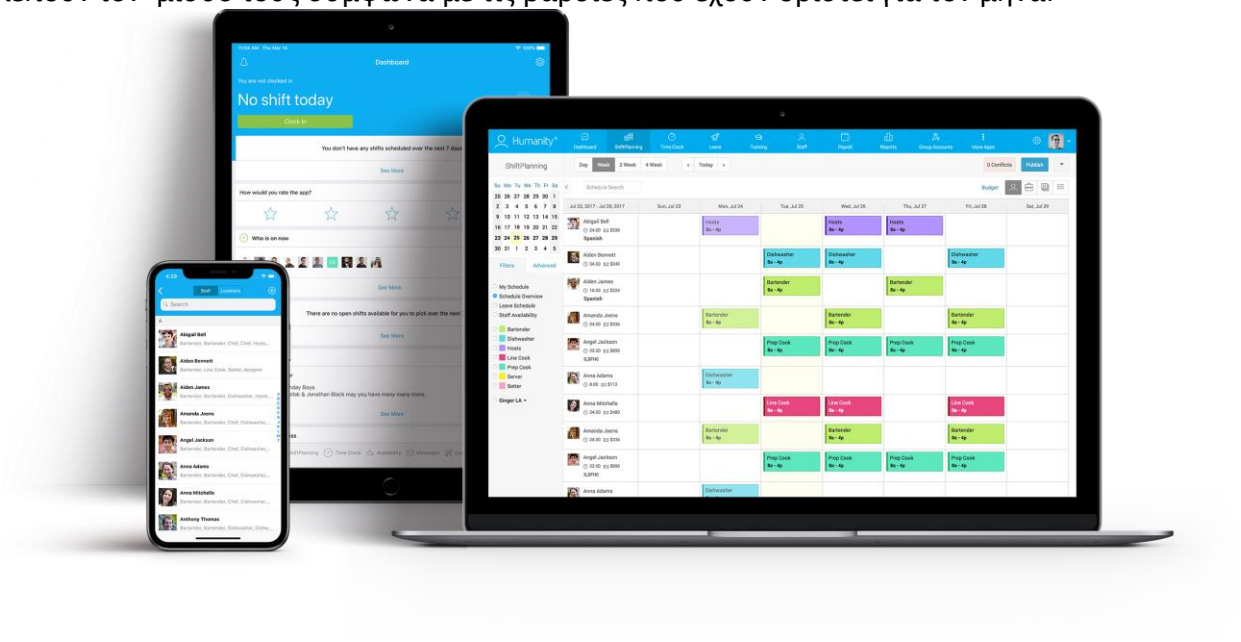
Θα πρέπει να λάβουμε υπόψη πως τα συστήματα roster, χωρίζονται σε δύο άτυπες κατηγορίες, high & low end. Τα Low End συστήματα εστιάζουν σε επιχειρήσεις όπως εστιατόρια δίνοντας έμφαση στην επικοινωνία μεταξύ των εργαζομένων, προσφέροντας υπηρεσίες παρόμοιες με τα μέσα κοινωνικής δικτύωσης, δηλαδή παρέχουν πληροφορίες σχετικά με συναδέλφους, και υπηρεσίες μηνυμάτων, είτε σε προσωπικό επίπεδο είτε κοινοποιήσεις προς ομάδες ανθρώπων. Το πλεονέκτημα τέτοιων συστημάτων είναι η ευκολία που προσφέρετε στους εργαζόμενους να συνεννοηθούν σχετικά με τις βάρδιες τους καθώς και να βρίσκουν εύκολα αντικαταστάτες στην περίπτωση που επιθυμούν να λείψουν από την εργασία τους.
Τα High End συστήματα κυρίως εμβαθύνουν στο προγραμματιστικό κομμάτι, με περισσότερους και πιο πολύπλοκους υπολογισμούς που αφορούν τις εργατοώρες, τις άδειες, την τήρηση της νομοθεσίας, την εξοικονόμηση πόρων και την αύξηση της παραγωγικότητας.
Παρακάτω θα μιλήσουμε για μερικά δημοφιλή roster συστήματα που υπόκεινται και στις δύο κατηγορίες :

- **Humanity**
- **Kronos**
- **Connecteam**

- **Ea-roster**

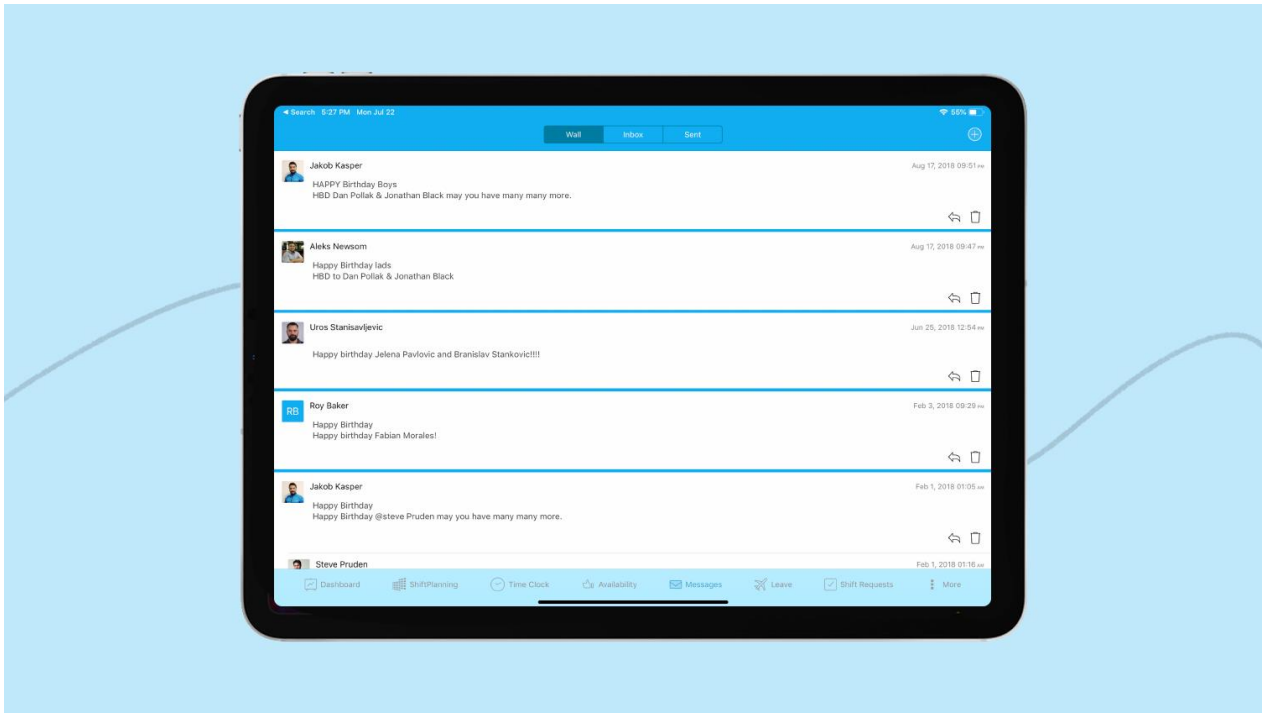
Humanity

Το Humanity είναι ένα από τα πιο διαδεδομένα roster συστήματα, υπόκειται στην κατηγορία των Low End συστημάτων και χρησιμοποιείται κυρίως σε επιχειρήσεις με σταθερό πρόγραμμα εργασίας όπως εστιατόρια. Το Humanity επιτρέπει στους εργαζόμενους να βλέπουν με πολλές παραλλαγές το πρόγραμμα εργασίας τους, να αιτούνται άδειες και να βλέπουν τον μισθό τους σύμφωνα με τις βάρδιες που έχουν οριστεί για τον μήνα.



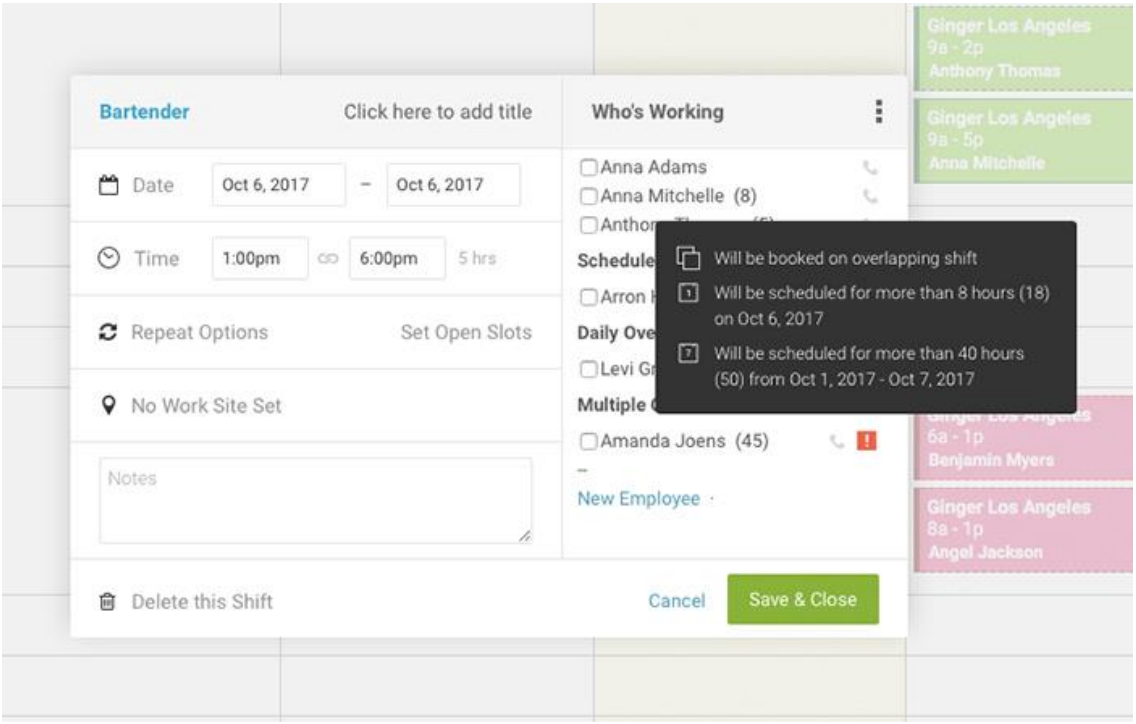
1. Πρόγραμμα Εργασίας

Το Humanity δίνει μεγάλη έμφαση στο User Experience προσφέροντας ένα όμορφο, καθαρό και εύχρηστο περιβάλλον είτε σε μορφή mobile app, είτε σε μορφή προγράμματος. Οι χρήστες του μπορούν εκτός από τις λειτουργίες που αφορούν άμεσα το πρόγραμμα και την μισθοδοσία τους να έχουν πρόσβαση σε πληροφορίες για τους συναδέλφους τους (προσωπικά μηνύματα, ενημέρωση για γενέθλια κα) και να ειδοποιούνται μέσω notification για αλλαγές ή έκδοση νέων προγραμμάτων.



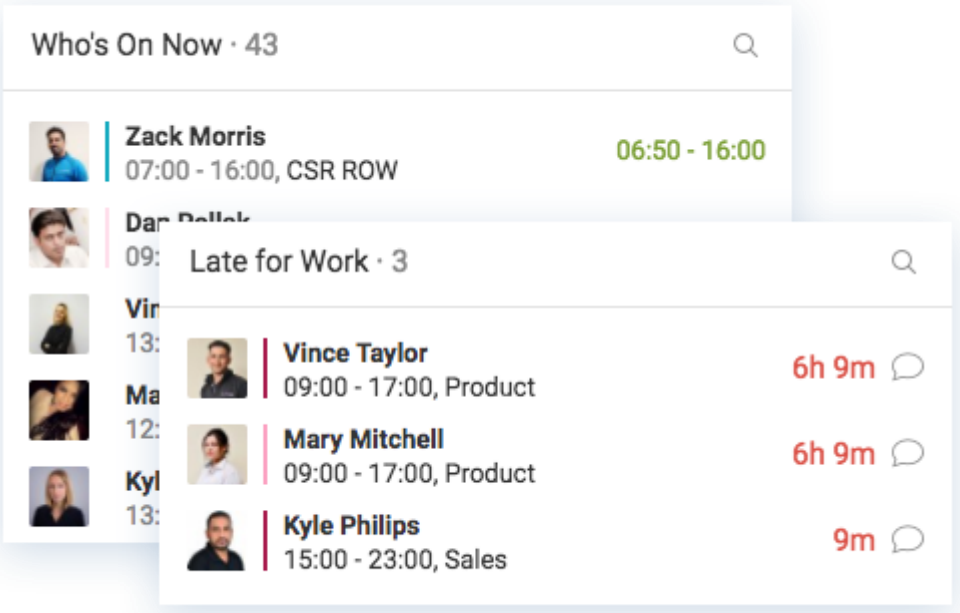
2.Chat & Notifications

Οι διαχειριστές του προγράμματος μπορούν να δημιουργούν εύκολα σε ένα φιλικό περιβάλλον τα προγράμματα εργασίας ενώ παράλληλα το ίδιο το πρόγραμμα αναλαμβάνει πλήρη έλεγχο σχετικά με το αν κάθε εργαζόμενος έχει πάρει τις ώρες που του αναλογούν και εάν υπάρχουν conflicts βαρδιών στο πρόγραμμα που δημιουργήθηκε.



3. Έλεγχος Conflicts

Επίσης υπάρχει πλήρης έλεγχος των εργαζομένων, κάθε διαχειριστής μπορεί να δει την ώρα που έφτασε για δουλειά κάθε εργαζόμενος, πότε πήρε και πόσο κράτησε το διάλειμμά του καθώς και πόση ώρα συνολικά καθυστέρησε μέσα στο μήνα.



4. Έλεγχος Εργαζομένων

Μια επίσης πολύ χρήσιμη λειτουργία είναι η εκπαίδευση όπου οι διαχειριστές μπορούν να κάνουν 'live' σεμινάρια ή και να ανεβάζουν υλικό που θέλουν να κοινοποιήσουν σε συγκεκριμένες ομάδες εργαζομένων. Παραδείγματος χάρη σε ένα εστιατόριο ο διαχειριστής θα μπορούσε να «μοιράσει» στους μάγειρες οδηγίες για το καινούργιο πιάτο που θέλει να συμπεριληφθεί στο menu του καταστήματος συμπεριλαμβάνοντας και φωτογραφικό υλικό που αφορά την παρουσίαση.
Τέλος πολύ σημαντική κρίνεται η δυνατότητα των διαχειριστών να δημιουργούν custom reports σχετικά με τα data που θέλουν να μελετήσουν χρησιμοποιώντας τα φίλτρα που τους παρέχονται, όπως ημερομηνία/τοποθεσία/εργαζόμενοι και ειδικότητα.
Αυτή την στιγμή το "Humanity" είναι το πιο διαδεδομένο cloud based, low end roster σύστημα, ας δούμε όμως τα υπέρ και τα κατά.

Υπέρ

1. Εύκολη επικοινωνία μεταξύ εργαζομένων αλλά και διαχειριστών
2. Έξυπνο σύστημα παρακολούθησης βαρδιών και εργαζομένων
3. Κατάλληλο για προβολή μικρών δειγμάτων δεδομένων
4. Εύχρηστο και κατανοητό Interface

Κατά

1. Ακριβό σε σχέση με τον ανταγωνισμό
2. Περιορισμένη ανάλυση δεδομένων και αναφορών

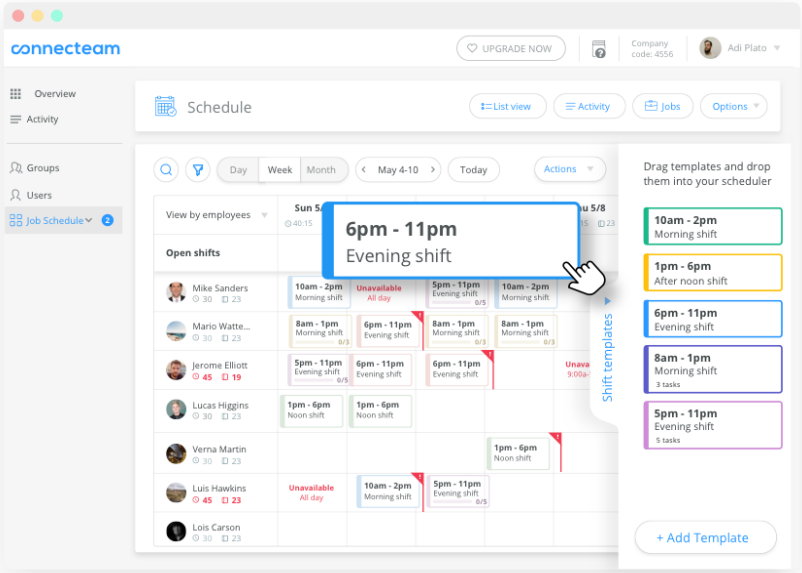
Kronos

Το Kronos υπόκειται στην κατηγορία των High End roster συστημάτων.
Είναι μια cloud LOB(Line of Business) σχεδιαστικά εφαρμογή που δίνει έμφαση στην διαχείριση των εργαζομένων και σε εργαλεία που αφορούν αναφορές, στατιστικές, τήρηση νομικών υποχρεώσεων και μισθοδοσίας με στόχο όχι μόνο την σωστή διαχείριση της επιχείρησης και του προγράμματος εργασίας αλλά και την βελτίωση της παραγωγικότητας των εργαζομένων.
Ένα από τα μεγαλύτερα πλεονεκτήματα του Kronos είναι η παρακολούθηση δεδομένων σε πραγματικό χρόνο που επιτρέπει στους διαχειριστές του συστήματος να αντιληφθούν και αντιμετωπίσουν προβλήματα χωρίς να περιμένουν τις καθιερωμένες εβδομαδιαίες ή και μηνιαίες αναφορές.
Επίσης θα πρέπει να λάβουμε υπόψη πως το Kronos έχει τη δυνατότητα να προβάλλει τα δεδομένα για πολύ συγκεκριμένες κατηγορίες και με επίσης πολύ συγκεκριμένη παραμετροποίηση. Έτσι πολλές φορές δεν χρειάζεται καν η μεσολάβηση αναλυτών για να παράγει ένας διαχειριστής τα αποτελέσματα που χρειάζεται για να λάβει τις απαραίτητες αποφάσεις. Καταλαβαίνουμε πως αυτά τα χαρακτηριστικά κάνουν το Kronos κατάλληλο για μεγάλες επιχειρήσεις, στο πελατολόγιο τους συμπεριλαμβάνονται εταιρίες όπως η (Tesla, Puma, MGM κα) ενώ χρησιμοποιείται και από μεγάλους κυβερνητικούς οργανισμούς.
Το Kronos είναι ένα πολύπλοκο σύστημα με πολλές παραμέτρους και εργαλεία, παρότι ένας εκπαιδευμένος διαχειριστής μπορεί να το αξιοποιήσει κατάλληλα για την δουλειά του παράγοντας στοχευμένα αποτελέσματα στις αναζητήσεις του η διαδικασία εκμάθησης είναι χρονοβόρα και δύσκολη και πολλές φορές χρειάζεται η μεσολάβηση ειδικών trainers για να να εκπαιδεύσουν τους χρήστες του προγράμματος.
Στα αρνητικά επίσης συγκαταλέγεται το ότι επειδή το Κρόνος έχει σχεδιαστεί για μεγάλες επιχειρήσεις η απομόνωση μικρών και συγκεκριμένων κομματιών πληροφορίας είναι εξαιρετικά δύσκολη.

Υπερ.	Κατά.
1. Διαχείριση και ανάλυση δεδομένων σε πραγματικό χρόνο	1. Δύσκολος χειρισμός
2. Μεγάλη γκάμα εργαλείων	2. Δύσκολη απομόνωση μικρών κομματιών πληροφορίας.

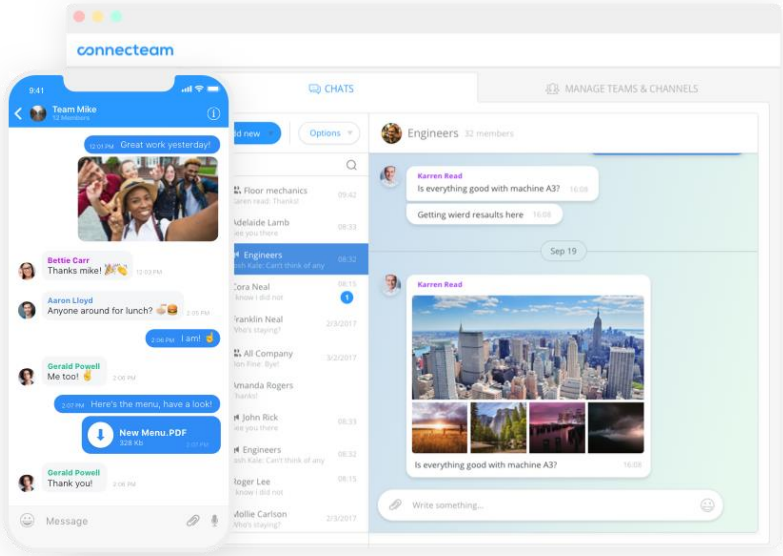
Connecteam

Πρόκειται για ακόμη μια Low End roster εφαρμογή. Όπως είδαμε και νωρίτερα στόχος τέτοιου τύπου εφαρμογών δεν είναι η ανάλυση σύνθετων δεδομένων αλλά η εύκολη επικοινωνία μεταξύ των χρηστών και η δημιουργία προγραμμάτων εργασίας αποτελεσματικά και γρήγορα σε ένα απλό και κατανοητό Interface.
Το connecteam έχει χαρακτηριστικά που βοηθούν τους διαχειριστές να δημιουργούν προγράμματα εργασίας με απλό και γρήγορο τρόπο. Πολλές φορές σε μικρότερες ειδικά επιχειρήσεις υπάρχουν περιπτώσεις που τα εβδομαδιαία/μηνιαία προγράμματα ή ακόμα και συγκεκριμένες βάρδιες ατόμων αλλάζουν από λίγο έως καθόλου. Για αυτό τον λόγο οι διαχειριστές δημιουργούν προγράμματα εργασίας τα οποία μπορούν να αποθηκεύουν ως templates και επαναχρησιμοποιούν καθιστώντας την διαδικασία γρήγορη και αποτελεσματική.



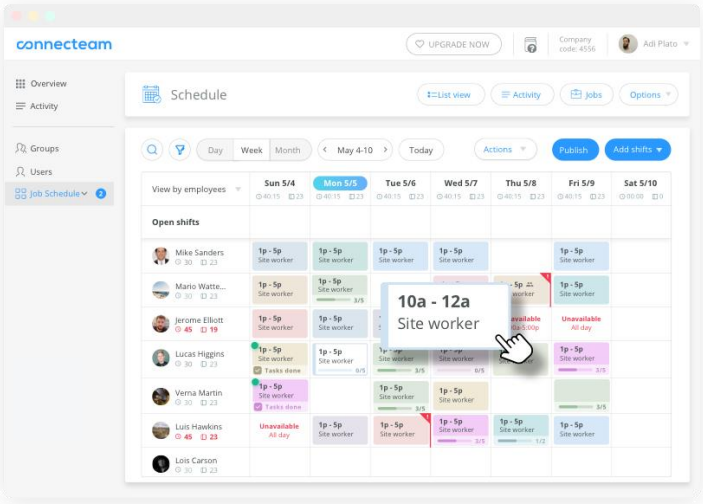
5. Template

Στην εφαρμογή υπάρχει λειτουργία επικοινωνίας παρόμοια με αυτή που προσφέρουν τα μέσα κοινωνικής δικτύωσης, σε μια εφαρμογή όμως που ο στόχος δεν είναι οι κοινωνικοποίηση αλλά η συνεννόηση μεταξύ των εργαζομένων θέλουμε η ανταλλαγή μηνυμάτων να γίνεται στοχευμένα με σκοπό την σωστή ενημέρωση χωρίς ανούσια μηνύματα και άχρηστες ειδοποιήσεις. Έτσι το connecteam δημιουργεί αυτόματα ομάδες επικοινωνίας ανάλογα με την ειδικότητα και τις βάρδιες που αναλαμβάνει το προσωπικό με σκοπό να εξοικονομήσει χρόνο στους χρήστες και να τους απαλλάξει από περιττές κινήσεις και μηνύματα.



6.Group Chat for Engineers

Τέλος η δημιουργία προγράμματος είναι σχεδιασμένη σε ένα τέτοιο περιβάλλον όπου κατά την υλοποίηση του προγράμματος εργασίας οι βάρδιες μπορούν να εύκολα και γρήγορα να γίνουν αντιγραφή, να μετακινηθούν με drag n drop ή και να προστεθούν σημειώσεις σε αυτές.



7. Shift Scheduling

Κάνοντας λοιπόν την επικοινωνία εύκολη μεταξύ των εργαζομένων/υπευθύνων αλλά και παρέχοντας στους διαχειριστές έναν γρήγορο, εύκολο και αξιόπιστο τρόπο να δημιουργούν προγράμματα εργασίας το Connecteam εξασφαλίζει σε επιχειρήσεις έναν γρήγορο και αξιόπιστο τρόπο να προγραμματίζουν την εργασία του προσωπικού τους καθώς και να επικοινωνούν γρήγορα και έξυπνα για οτιδήποτε αφορά την λειτουργία της επιχείρησης. Καταλαβαίνουμε λοιπόν πως όχι μόνο στην συγκεκριμένη περίπτωση αλλά και γενικότερα τα Low End συστήματα απευθύνονται σε μικρότερες επιχειρήσεις δίνοντας τους την δυνατότητα να οργανώνουν και να σχεδιάζουν το άμεσο μέλλον με αποτελεσματικό τρόπο μέσω μιας εφαρμογής στην οποία δεν χρειάζεται να εκπαιδευτούν.

Υπερ.	Κατά.
1. Εύκολη χρήση, Φιλικό περιβάλλον	1. Περιορισμένη γκάμα εργαλείων
2. Στοχευμένη επικοινωνία	2. Περιορισμένη έκδοση reports
3. Πολύ εύκολη δημιουργία προγραμμάτων εργασίας	
4. Δεν απαιτείται εκπαίδευση	

ea-roster

Το ea-roster είναι μια high end ολοκληρωμένη υπηρεσίας για τον προγραμματισμό, την διαχείρηση και την βέλτιστη αξιοποίηση του ανθρώπινου δυναμικού μιας Επιχείρησης ή Οργανισμού.

Στόχος του ea-roster είναι ο βέλτιστος προγραμματισμός της εργασίας και η δυνατότητα προσαρμογής του ανθρώπινου δυναμικού σε κυμαινόμενη ζήτηση. Το ea-roster προσφέρει τα απαραίτητα υποσυστήματα, που είναι

- Προγραμματισμός της εργασίας
- Δημιουργία - Διαχείριση Βαρδιών
- Ενημέρωση Σ.ΕΠ.Ε. (“ΕΡΓΆΝΗ”)
- Εργαζόμενοι
- Διαχείριση αδειών
- Σύνδεση με σύστημα Προσέλευσης - Ωρομέτρησης.
- Πληροφόρηση των εργαζομένων μέσω Κινητού
- Reporting

Προγραμματισμός της Εργασίας

Κάθε εταιρεία έχει, πιθανότατα, το δικό της διαφορετικό πρόγραμμα εργασίας, βάσει του κανονισμού, της εργατικής νομοθεσίας, των ειδικοτήτων που απασχολεί κ.λπ. Ακόμα και εργαζόμενοι της εταιρείας στο ίδιο τμήμα είναι πιθανό να ακολουθούν διαφορετικά προγράμματα εργασίας, ανάλογα με τις ανάγκες, που καλύπτουν.

Το **ea-roster** προσφέρει την υποδομή και διαθέτει τα απαραίτητα εργαλεία για την λειτουργική υποστήριξη οποιασδήποτε απαίτησης.

Τύποι Προγραμμάτων Εργασίας

Υποστηρίζει όλους τους διαφορετικούς τύπους εργασίας, που συναντώνται στις παραγωγικές επιχειρήσεις και είναι:

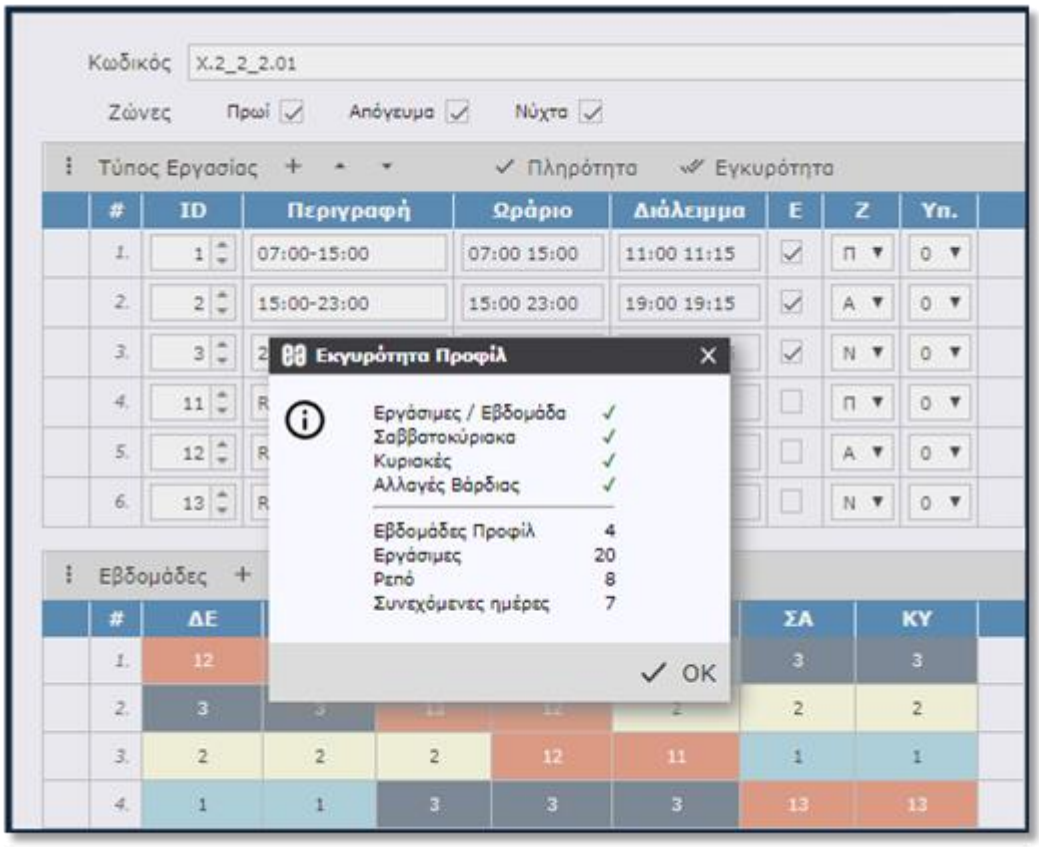
Σταθερού Ωραρίου: οι εργαζόμενοι, ακολουθούν σταθερό ωράριο για καθορισμένες ημέρες εργασίας, συνήθως από Δευτέρα έως Παρασκευή.

Σταθερής Ζήτησης: οι εργαζόμενοι εναλλάσσονται σε βάρδιες με καθορισμένο και σταθερό, εν γένει, αριθμό εργαζομένων σε κάθε βάρδια.

Κυμαινόμενης Ζήτησης: ο αριθμός των εργαζόμενων, που επανδρώνει την κάθε βάρδια, είναι δυνατόν να μεταβάλλεται ανάλογα με την ζήτηση.

Δημιουργία Προγραμμάτων Εργασίας

Το **ea-roster** έχει ενσωματωμένο εργαλείο, για την δημιουργία των προγραμμάτων εργασίας των εργαζομένων. Ο διαχειριστής κατασκευάζει τα πρότυπα προγράμματα εργασίας, τα οποία καλύπτουν τις λειτουργικές ανάγκες όλων των τμημάτων της επιχείρησης, χωρίς περιορισμούς.



Εικόνα 1: Δημιουργία Προγράμματος – Έλεγχος Συμβατότητας

Για το λόγο αυτό, κάθε πρόγραμμα, μετά την δημιουργία του και πριν από την εφαρμογή του, ελέγχεται για την ορθότητά του και την συμβατότητά του με τους βασικούς κανόνες της εργατικής νομοθεσίας (αριθμός συνεχόμενων εργασίμων ημερών, ελεύθερα Σαββατοκύριακα και Κυριακές, αλλαγές βάρδιας, κλπ.).

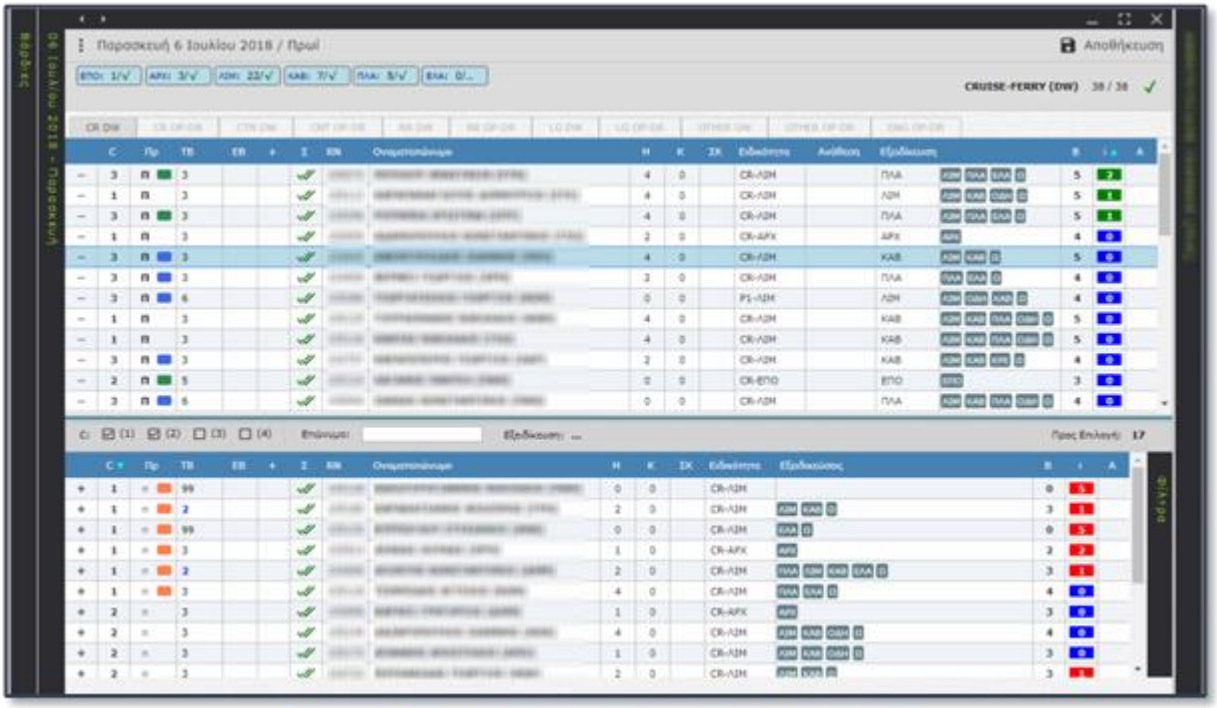
Σε κάθε εργαζόμενο εκχωρείται το πρόγραμμα εργασίας του, το οποίο αναπτύσσεται για οποιαδήποτε μελλοντική περίοδο και ισχύει μέχρι την αντικατάστασή του από άλλο.

Επιπλέον, ο διαχειριστής έχει τη δυνατότητα **να εφαρμόσει δοκιμαστικά τα νέα προγράμματα εργασίας,** πριν από την τελική εφαρμογή τους και να επαληθεύσει τη επιθυμητή διαθεσιμότητά, ανά βάρδια και ημέρα εργασίας.

Δημιουργία – Διαχείριση Βαρδιών

Κάθε εργαζόμενος, ανεξάρτητα από τον τύπο εργασίας που ανήκει, έχει στη διάθεσή του, και ακολουθεί το πρόγραμμα εργασίας του. Πρόσθετα, οι εργαζόμενοι, που ανήκουν σε τμήματα με κυμαινόμενη ζήτηση, θα πρέπει να καλύπτουν τις ζητούμενες θέσεις εργασίας, ανά εξειδίκευση και για κάθε βάρδια.

Έτσι, για κάθε βάρδια κυμαινόμενης ζήτησης αποτυπώνεται αριθμητικά η **ζήτηση των εργαζομένων** για τις εξειδικεύσεις, που περιλαμβάνει το εξεταζόμενο κέντρο εργασίας.



Εικόνα 2: Δημιουργία Βάρδιας – Δυναμική Ζήτηση

Σ

τη συνέχεια, γίνεται η **επίλυση της βάρδιας**, ώστε να καλυφθούν όλες οι ζητούμενες εξειδικεύσεις του κάθε εργασιακού κέντρου. Αυτόματα επιλέγονται εκείνοι οι εργαζόμενοι, οι οποίοι βάσει του προγράμματος τους, εργάζονται στη εξεταζόμενοι βάρδια, εκτός από αυτούς, που έχουν οποιουδήποτε τύπου άδεια.

Οι χειριστές έχουν στη διάθεσή τους μια σειρά δεικτών, όπως: απόσταση από την τελευταία βάρδια που εργάσθηκε, συνεχόμενες ημέρες εργασίας, τελευταία ελεύθερη Κυριακή κ.α. Οι δείκτες αυτοί χρησιμοποιούμενοι συνδυαστικά, απλοποιούν την διαδικασία της επιλογής και σε συνδυασμό με τα εργαλεία επίλυσης επιτρέπουν την γρήγορη και αλάνθαστη επίλυση της βάρδιας.

Πρόσθετα, για τους συμμετέχοντες στην εξεταζόμενη βάρδια, μπορούν να αλλάξουν την ώρα προσέλευσης, νωρίτερα ή αργότερα, ή να προσθέσουν τυχόν ώρες υπερωριών.

Όταν ολοκληρωθεί η διαδικασία επιλογής της βάρδιας, έχει τη **δυνατότητα ελέγχου για τυχόν υπερβάσεις**, όπως ο έλεγχος 11-ώρου, διπλό-βάρδια, επιλογή αδειούχου, κ.α.

Παράλληλα, είναι άμεσα διαθέσιμες όλες οι απαραίτητες εκτυπώσεις, με πολλαπλές επιλογές ταξινόμησης: αλφαβητικά, εξειδίκευση, ώρα προέλευσης κα.

Εργαλεία επίλυσης

Οι χειριστές έχουν στη διάθεσή τους μια σειρά εργαλείων, τα οποία βοηθούν στην γρήγορη επίλυση της βάρδιας με ταυτόχρονη βέλτιστη αξιοποίηση του ανθρώπινου δυναμικού, όπως:

Διαθεσιμότητα εργαζομένων ανά εξειδίκευση, δυναμική λίστα με τις διαθέσιμες εξειδικεύσεις και από πόσους εργαζόμενους καλύπτεται η κάθε μία.

Έλεγχος βέλτιστης εκχώρησης εξειδίκευσης για εργαζόμενους με πολλαπλές εξειδικεύσεις και που έχουν ήδη επιλεγεί για τη βάρδια. Ποιοι εργαζόμενοι έχουν επιλεγεί με συγκεκριμένη εξειδίκευση και ποιοι την διαθέτουν αλλά έχουν επιλεγεί με διαφορετική.

Μηχανισμός ροών επιτρέπει τη μετακίνηση εργαζομένων, της ίδιας εξειδίκευσης, από ένα τμήμα που πλεονάζουν, σε άλλο, στο οποίο υπάρχει έλλειψη, για την επιλυόμενη βάρδια.

Αυτόματη επίλυση βάρδιας με τη χρήση αλγορίθμων και σε δυναμικά καθοριζόμενο σύνολο εργαζομένων, με επιλογή αντίστοιχων κατηγοριών C1 . . . C4 και φίλτρων, επίλυση της κυμαινόμενης ζήτησης.

Αναθέσεις

Ο μηχανισμός των αναθέσεων επιτρέπει την **ανάθεση μηχανήματος ή εργαλείου** σε εργαζόμενο, εφόσον η εξειδίκευσή του το απαιτεί. Κατά την διαδικασία της ανάθεσης, εμφανίζεται λίστα όλων των μηχανημάτων της εξειδίκευσης με το πλήθος των εκχωρήσεων, που έχουν γίνει στον εργαζόμενο σε καθορισμένο προηγούμενο χρονικό διάστημα.
Ο ίδιος μηχανισμός αναθέσεων μπορεί να χρησιμοποιηθεί για την εκχώρηση θέσεων αναφοράς, περιοχών εργασίας κλπ.

Ενημέρωση Σ.ΕΠ.Ε. (“ΕΡΓΑΝΗ”)

Με βάση το πρόγραμμα εργασίας του κάθε εργαζόμενο ο διαχειριστής δημιουργεί αυτόματα το **Μηνιαίο Πρόγραμμα εργασίας** του Οργανισμού, το οποίο και υποβάλλει, πριν από την έναρξη του μήνα, στο Σ.ΕΠ.Ε. (“ΕΡΓΑΝΗ”).
Όλες οι μεταβολές, που γίνονται καθημερινά και σε κάθε βάρδια, σε σχέση με το μηνιαίο πρόγραμμα: *άδειες*, *ρεπό, τροποποίηση ωραρίων*, *κλήση από ρεπό* ή *σε ρεπό*, καταγράφονται με αυτόματο και διαφανή τρόπο, στα αρχεία μεταβολών, που θα ενημερώσουν το Σ.ΕΠ.Ε. (“ΕΡΓΑΝΗ”).

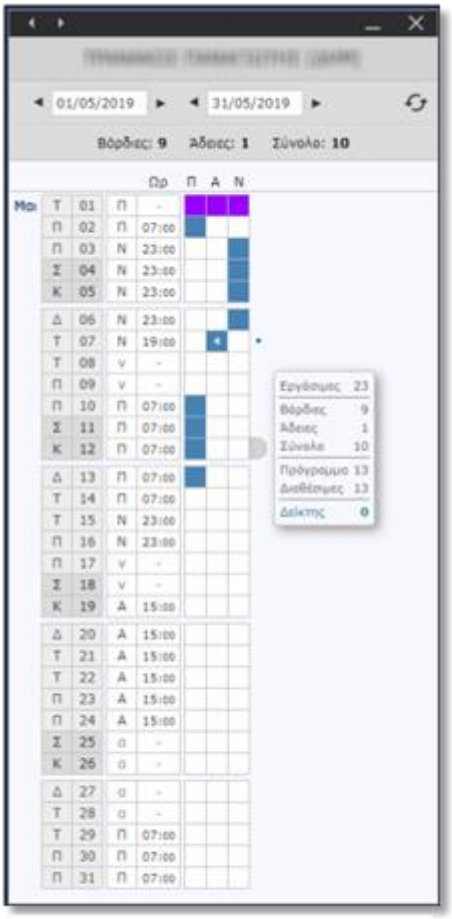
Εικόνα 3: Μηνιαίο Πρόγραμμα Εργαζομένων

Η ενημέρωση αυτή γίνεται σε καθορισμένα ορόσημα της ημέρας, που εξαρτώνται από τα διαφορετικά ωράρια των εργαζομένων, και περιέχουν όλες τις μεταβολές σε σχέση με το Μηνιαίο Πρόγραμμα εργασίας. Οι οριστικοποιήσεις περιλαμβάνουν όλες τις έγκυρες μεταβολές, μέσα στα όρια υποδοχής των αλλαγών, που ορίζεται από το “ΕΡΓΑΝΗ”.
Κατ’ αντιστοιχία παράγονται τα αρχεία ενημέρωσης του Σ.ΕΠ.Ε. (“ΕΡΓΑΝΗ”), με τις **ώρες υπερεργασίας και υπερωριών**, πριν από την λήξη της σχετικής βάρδιας.
Για κάθε ημέρα, υπάρχει αναλυτική **λίστα δημιουργίας των σχετικών αρχείων**, που έχουν γίνει για το Σ.ΕΠ.Ε., με την ακριβή ώρα δημιουργίας, από ποιον χειριστή και αν αφορά τροποποιητικές ή ώρες υπερεργασίας και υπερωριών.

Εργαζόμενοι

Το “**ea-roster**” δίνει τη δυνατότητα λεπτομερούς καταγραφής όλων των απαραίτητων δεδομένων των εργαζομένων. Καλύπτει τις περιπτώσεις, στις οποίες ο εργαζόμενος διαθέτει περισσότερες από μία ενεργές εξειδικεύσεις. Στην περίπτωση αυτή η σειρά εμφάνισης των ειδικοτήτων είναι και η **σειρά προτίμησης επιλογής** του εργαζόμενου.
Ο εργαζόμενος θα επιλεγεί στη βάρδια βάσει μιας από τις διαθέσιμες εξειδικεύσεις του. Η δυνατότητα αυτή παρέχει μεγάλη **ευελιξία στην αξιοποίηση του ανθρώπινου δυναμικού**.
Η **γραφική απεικόνιση** παρέχει, με μια ματιά, χρήσιμες πληροφορίες για **την εργασιακή κατάσταση** του εργαζόμενου, κατά διάρκεια του τρέχοντος μήνα ή μεγαλύτερης περιόδου, όπως:

- ποιο είναι το πρόγραμμα εργασίας του και σε ποιες βάρδιες εργάσθηκε,
- πότε έχει ρεπό και τυχόν άδειες,
- αν έχει κάνει υπερωρίες σε παρελθούσες βάρδιες και πόσες ώρες κάθε φορά,
- δεσμεύσεις εργασίας σε ανταπόδοση ρεπό κα.



Εικόνα 4: Εργασιακή Κατάσταση

Η γραφική εικόνα είναι διαθέσιμη σε όλα τα βασικά υποσυστήματα: σύνταξη βάρδιας, προγραμματισμός αδειών κα

Συνοπτικά, η εργασιακή κατάσταση του εργαζόμενου εκφράζεται με τον "**Δείκτη**" (**index**), ο οποίος **αποτυπώνει αριθμητικά και χρωματικά την εξέλιξή της, σε σχέση με** το πρόγραμμά του, κατά τον τρέχοντα μήνα,.

Οι managers έχουν άμεσα στη διάθεσή τους ιστορικά δεδομένα για τους εργαζόμενους σχετικά με τις βάρδιες εργασίας την κατανομή τους καθώς και για τις άδειες, ανά μήνα και ανά έτος, από την ημερομηνία πρόσληψης.

Διαχείριση Αδειών

Το **ea-roster** διαθέτει **υποσύστημα διαχείρισης αδειών**. Επιτρέπει τον ορισμό κάθε τύπου άδειας, που χρησιμοποιεί η επιχείρηση και την αντιστοίχιση της σε μία από τις βασικές κατηγορίες αδειών: κανονική, αναρρωτική, με αποδοχές, άνευ αποδοχών και ρεπό.

Βασικό πλεονέκτημα είναι η **ευελιξία στη διαχείριση των "Ρεπό"**. Πλέον, μπορεί να προγραμματισθεί με την χρήση του μηχανισμού "**εξαίρεση εργασίας**", από μία βάρδια, χωρίς να δεσμεύεται, απαραίτητα, ολόκληρη η ημέρα.

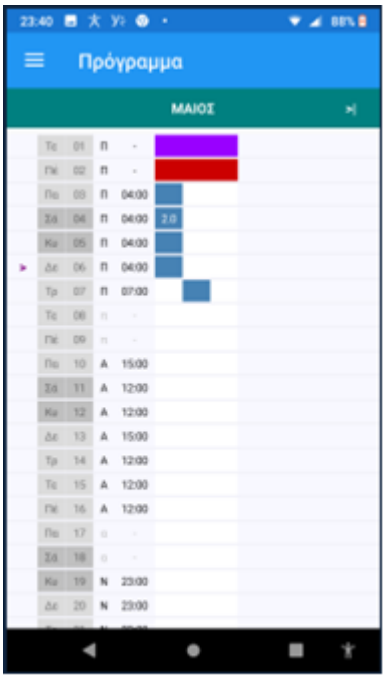
Επίσης, ένα ζητούμενο ρεπό μπορεί να συνδυαστεί με παρεχόμενη, εκ μέρους του εργαζόμενου, "***δέσμευση εργασίας***". Ο εργαζόμενος δεσμεύεται να εργασθεί σε μια

μελλοντική βάρδια, σε ημέρα που έχει ρεπό βάσει του προγράμματος, σε ανταπόδοση του ζητούμενου "ρεπό", (**ανταποδοτικό ρεπό**).
Το παραπάνω χαρακτηριστικά μπορούν να χρησιμοποιηθούν σαν εργαλεία για βέλτιστο προγραμματισμό του ανθρώπινου δυναμικού και την προσαρμογή του σε κυμαινόμενη ζήτηση.
Για την διαχρονικά δίκαιη κατανομή των κανονικών αδειών, κάθε εργαζόμενος εντάσσεται σε συγκεκριμένη ομάδα άδειας.

Σύνδεση με σύστημα παρουσιών

Για την καταγραφή της προσέλευσης των εργαζομένων στη βάρδια, το **ea-roster** έχει τη δυνατότητα σύνδεσης με **μονάδες καταγραφής**, όπως οι αναγνώστες καρτών (card readers). Ο επόπτης παρακολουθεί, σε πραγματικό χρόνο, την έγκαιρη προσέλευση, των εργαζομένων, που απαρτίζουν την βάρδια, ανά εξειδίκευση.
Ο αναγνώστης καρτών συνδέεται με Η/Υ και εμφανίζει, στην οθόνη του, πληροφορίες που αφορούν τον εργαζόμενο, όπως: το μηχάνημα που θα χειρισθεί, τη θέση εργασίας του, γενικές πληροφορίες για τη βάρδια, κα.
Η αναλυτική καταγραφή των ωρών εργασίας και το υποσύστημα διαχείρισης αδειών δίνει τη δυνατότητα παραγωγής αναφοράς, η οποία περιέχει όλες τις πληροφορίες, που είναι απαραίτητες για την τροφοδοσία του **προγράμματος μισθοδοσίας** (αναλυτικά ημερομίσθια, άδειες, νυχτερινά, υπερωρίες, αργίες).
Σε κάθε περίπτωση, η ενημέρωση του υποσυστήματος των αδειών είναι προϋπόθεση για τον απρόσκοπτο προγραμματισμό των βαρδιών.

Πληροφόρηση μέσω Κινητού



Εικόνα 5: Ενημέρωση κινητού

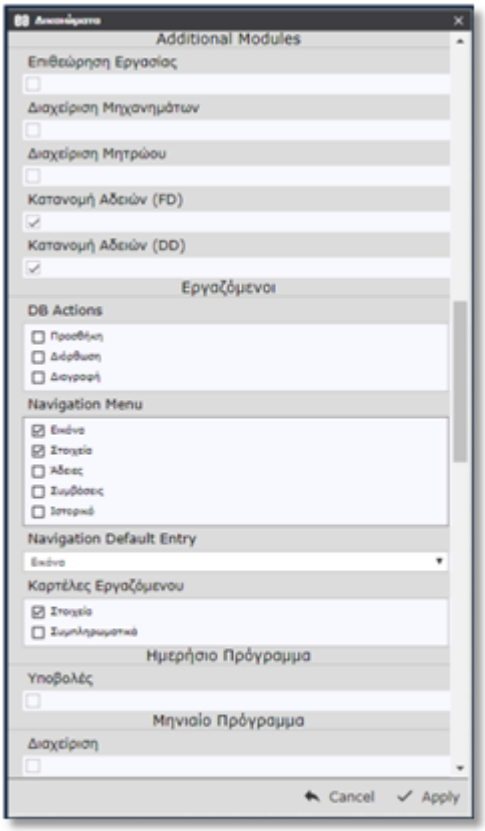
Το υποσύστημα "**ea-roster-mobility**" επεκτείνει τη πληροφόρηση των εργαζομένων με χρήση του κινητού τους. Κάθε εργαζόμενος έχει τη δυνατότητα να βλέπει, οποιαδήποτε στιγμή, πληροφορίες σχετικά με το πρόγραμμα εργασίας και τις άδειες του.
Εμφανίζει γραφικά το ατομικό πρόγραμμα εργασίας του τρέχοντα και του επόμενου ημερολογιακού μήνα, όπως έχει διαμορφωθεί μέχρι εκείνη τη στιγμή.

Πρόσθετα, εμφανίζει αναλυτικά όλες τις άδειες, που έχει πάρει ή προγραμματίσει, από την αρχή του έτους.
Με το τρόπο αυτό, οι εργαζόμενοι έχουν άμεση και πλήρη εικόνα για την εργασία τους και τις άδειές τους.

Αναφορές

Όλες οι αναφορές παράγονται σε **πίνακες Excel®**, και διαθέτουν μεγάλη ευελιξία στη στην παραμετροποίηση. Τα πιο σημαντικά χαρακτηριστικά είναι:

- Επιλογή των πεδίων (στηλών) που θα περιέχει και καθορισμό της σειράς εμφάνισής τους
- Δυνατότητα αποθήκευσης διαφορετικών αναφορών σε χωριστά φύλλα του ίδιου αρχείου Excel (πολλαπλά data sources).
- Για αναφορές με πολύπλοκη παραμετροποίηση, δυνατότητα αποθήκευσης των επιλογών για την επαναχρησιμοποίηση τους μελλοντικά.
- Διαμοιρασμό της παραμετροποίησης μιας αναφοράς σε άλλους χρήστες.
- Ασφάλεια πρόσβασης, με καθορισμό της διαθεσιμότητας της κάθε αναφοράς, ανά χρήστη.
- Σε κάθε αναφορά, εμφάνιση μόνο των δεδομένων, για τα οποία έχει δοθεί δικαίωμα πρόσβασης.



Εικόνα 6: Δικαιώματα - Υποσυστήματα

Η παραγωγή της κάθε αναφοράς σε πίνακα Excel, δίνει τη δυνατότητα πρόσθετης επεξεργασίας της με την άμεση χρήση και αξιοποίηση των διαθέσιμων εργαλείων του Excel: pivoting, χρήση φίλτρων, συναρτήσεων, ταξινόμηση, κα.

Ασφάλεια Δεδομένων

Το **ea-roster** διαθέτει **ενσωματωμένο σύστημα δικαιωμάτων**, με προκαθορισμένους ρόλους, οι οποίοι καθορίζουν το γενικότερο επίπεδο πρόσβασης στην εφαρμογή.

Δικαιώματα Πρόσβασης

Εκτός από τους ρόλους, ο διαχειριστής καθορίζει με ακρίβεια τα δικαιώματα πρόσβασης που έχει **ο κάθε χρήστης** και αφορούν:

- Το τμήμα ή τα **τμήματα του οργανισμού**, στα οποία έχει πρόσβαση
- τα **υποσυστήματα της εφαρμογής**, στα οποία θα έχει πρόσβαση,
- το **επίπεδο πρόσβασης** σε κάθε υποσύστημα, δηλ. οι ενέργειες που είναι επιτρεπτές σε κάθε ένα (εισαγωγή, ενημέρωση, διαγραφή)
- την εκχώρηση πρόσβασης σε συγκεκριμένα υποσυστήματα της εφαρμογής, για ορισμένους χρήστες και χρονικό διάστημα, πχ ενημέρωση Σ.ΕΠ.Ε.
- **τις αναφορές**, επακριβώς, **τις οποίες έχει στη διάθεσή του**, και με δεδομένα, που έχουν ήδη καθορισθεί από τα δικαιώματα πρόσβασης στα αντίστοιχα τμήματα του οργανισμού.

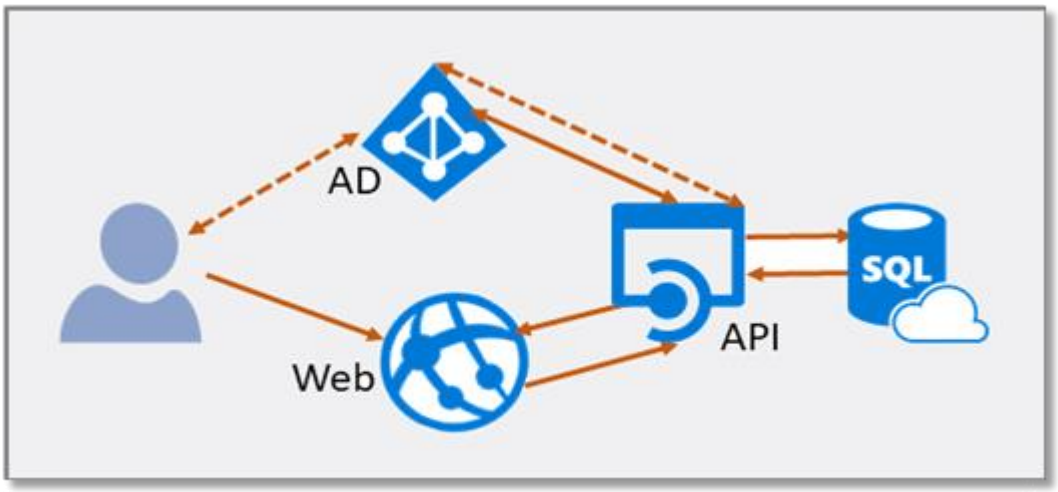
Καταγραφή Ενεργειών (Logging)

Το **ea-roster** διαθέτει **υποσύστημα καταγραφής** για τις κρίσιμες λειτουργίες της εφαρμογής, όπως η ενημέρωση βαρδιών, υπερωριών κ.λπ. Η καταγραφή περιλαμβάνει τον χρήστη, τον ακριβή χρόνο, τον τύπο της μεταβολής κ.α.

Υπηρεσία Cloud

Το **ea-roster** προσφέρεται σαν υπηρεσία cloud στη **πλατφόρμα Microsoft Azure®**, το οποίο και κατοχυρώνει:

- Υψηλό επίπεδο διαθεσιμότητας και υποστήριξης (24x7x365)
- Ασφαλή πρόσβαση και χρήση (Azure Active Directory)
- Άμεση επεκτασιμότητα με τη δυνατότητα ενσωμάτωσης - χρήσης νέων υπηρεσιών
- Τελευταίες εκδόσεις των χρησιμοποιούμενων υπηρεσιών
- Άμεση κάλυψη επιχειρήσεων που καλύπτουν εκτεταμένη περιοχή (λιμάνια, campus)



Εικόνα 7: Αρχιτεκτονική Υπηρεσιών Azure

- Κατοχύρωση της συμβατότητας με τον κανονισμό GDPR.

Περιγραφή Συστήματος (Τεχνολογίες / Αρχιτεκτονική)

Αρχιτεκτονική (MVVM, OOP, Knockout, Webpack)

Για την ανάπτυξη του συγκεκριμένου project αξιοποιήσαμε την τεχνική του Αντικειμενοστραφούς Προγραμματισμού που βασίζεται στις κλάσεις και τα αντικείμενα (παρότι δεν αξιοποιήσαμε πλήρως την «κληρονομικότητα»).
Η αρχιτεκτονική MVVM χωρίζει ουσιαστικά το κομμάτι της λειτουργικότητας (MV) και το κομμάτι της παρουσίασης (VM) και μας επιτρέπει να επαναχρησιμοποιούμε και να επεξεργαζόμαστε τον κώδικα μας εύκολα κρατώντας το project καθαρό και διαχειρίσιμο.
Αναλυτικότερα, με την βοήθεια της Knockout.js και την λειτουργία των imports για Javascript & HTML αρχεία μπορούμε να δημιουργήσουμε αντικείμενα που αποτελούνται από κλάσεις «components»

```
import ko from "knockout"
import template from "./view.html"


class VM{
    constructor(variable){
        //Some JS or ko variables
    }

  //Some JS functions
}


ko.components.register("example1",{

    viewModel:VM,
    template:template
});
```

Με αυτή την τεχνική εξασφαλίζουμε πως τα components είναι τελείως αυτόνομα όσον αφορά την λειτουργικότητα τους και μπορούν να αποθηκεύονται μόνα τους καθώς και να επαναχρησιμοποιούνται στον κώδικα όπου επιλέξουμε χρησιμοποιώντας το “**data-bind**” attribute που δεσμεύει η Knockout στην HTML

```
<div data-bind="component:{name:'example1'}"></div>
```

Με την βοήθεια του webpack όλα τα components θα γίνουν import στο javascript αρχείο που θα χρησιμοποιηθεί σαν **“entry point”** για την εφαρμογή και μέσω της συνάρτησης

```
ko.applyBindings(component);
```

και των **ko.observable** μεταβλητών η knockout αναλαμβάνει να παρακολουθεί δυναμικά τις αλλαγές στα δεδομένα που προβάλλονται. Natively, η Javascript δεν μπορεί να υποστηρίξει την τεχνική των imports που απαιτείται για την δημιουργία των components, εκεί λοιπόν αναλαμβάνει το webpack και οι κατάλληλοι loaders που υποστηρίζουν την μέθοδο των imports για τα Javascript και HTML αρχεία και μας επιτρέπουν να κάνουμε bundle βιβλιοθήκες κατευθείαν από τον κώδικα μας κρατώντας σταθερό version χωρίς την βοήθεια του npm ή

παραθέτοντας CDN links στην HTML καθώς και να είμαστε σίγουροι ότι διατηρείται η σωστή ιεραρχία στα dependencies του προγράμματος καθώς και την δυνατότητα να κάνουμε minimize τον κώδικα όταν ολοκληρώσουμε το project.
Ως "entry point" για την Javascript έχουμε ορίσει την **main.js**. Έτσι λοιπόν το webpack βλέπει όλα τα αρχεία που έχουν γίνει import και χτίζει το output εκεί που το έχουμε ορίσει στο configuration του Webpack

```
    entry: {
        thesis: './app/main.js'
    },
    output: {
        path: path.resolve(__dirname, './dist/_js'),
        filename: '[name].js'
    }
```

Για την υποστήριξη των HTML imports έχουμε φορτώσει τον κατάλληλο loader

```
  module: {
        rules: [{
            test: /\.html$/,
            use: [{
                loader: 'html-loader',
                options: {
                    minimize: false
                }
            }]
```

Ενώ έχουμε ξεχωρίσει τον κώδικα σε 2 chunks, ένα για τον δικό μας κώδικα που θα γίνει bundle στο **thesis.js** όπως είδαμε παραπάνω και ένα δεύτερο για τις βιβλιοθήκες της JQUERY και KNOCKOUT που θα γίνουν bundle σε ένα δεύτερο αρχείο, **vendors.js**

```
            cacheGroups: {
                commons: {
                    test: /[\\/]_libs\\(jquery|knockout)[\\/]/,
                    name: 'vendors',

                }
            }
```

Client και Server

Εισαγωγή

Κάθε εφαρμογή που αναλαμβάνει να διαχειριστεί η να προβάλει δεδομένα αποτελείται από τρία βασικά μέρη.
Τον **Client** , που παρουσιάζει δεδομένα,
Τον **Server**, που αποθηκεύει δεδομένα,
Το **Service**, που καλείται από τον Client και αναλαμβάνει να φέρει ή/και να στείλει δεδομένα στον Server αξιοποιώντας το πρωτόκολλο HTTP.

Client Side

Χρησιμοποιώντας τις γλώσσες προγραμματισμού Javascript, HTML και CSS συνθέτουμε το client κομμάτι του προγράμματος που είναι υπεύθυνο για την επεξεργασία και παρουσίαση των δεδομένων. Η HTML5 είναι η γλώσσα που αναλαμβάνει την οπτικοποίηση των δεδομένων, η CSS αναλαμβάνει να βελτιώσει την εμπειρία του χρήστη όχι μόνο ενισχύοντας το παρουσιαστικό αλλά εξασφαλίζοντας ότι η εφαρμογή θα προσαρμόζεται στο μέγεθος της οθόνης του χρήστη (Responsiveness) κάνοντας έτσι εφικτή την χρήση του προγράμματος από οποιαδήποτε συσκευή έχει στην διάθεση του ο τελικός χρήστης, ενώ τελικά η Javascript είναι η γλώσσα που αναλαμβάνει την επεξεργασία των δεδομένων και την ανταπόκριση των λειτουργιών του προγράμματος σε πραγματικό χρόνο.
Επιπρόσθετα έχουν χρησιμοποιηθεί δύο frameworks ,
Α) Για την CSS, η **Bootstrap**.

Η Bootstrap είναι το πιο διαδεδομένο CSS framework για την εύκολη και γρήγορη ανάπτυξη αισθητικά όμορφων και responsive ιστοσελίδων, χρησιμοποιεί την CSS, την Javascript και την PopperJS (για tooltips, popovers, dropdowns, modals) και δίνει την δυνατότητα στον χρήστη να «χτίζει» λειτουργικό, χρονοβόρο και πολύπλοκο κώδικα εύκολα και γρήγορα εισάγοντας έτοιμες κλάσεις κατευθείαν στην HTML.

Β) Για την Javascript , η **Knockout.js**

Αναφέραμε σε προηγούμενο κεφάλαιο το πως η Knockout μας παρέχει τους απαραίτητους μηχανισμούς για να χτίσουμε τα components που χρειαζόμαστε για το μοντέλο αρχιτεκτονικής που επιλέξαμε. Επιπρόσθετα η Knockout μας εξασφαλίζει ότι το περιβάλλον παρουσίασης θα παρακολουθείται διαρκώς και θα αντιδρά σε οποιαδήποτε αλλαγή συμβεί σε πραγματικό χρόνο μέσω των Observable μεταβλητών. Αυτό είναι κάτι που δεν υποστηρίζεται από την Native Javascript και κρίνεται απολύτως απαραίτητο για την ορθή ανάπτυξη της εφαρμογής μας. Για την καλύτερη λειτουργία της knockout συμπεριλάβαμε και την βιβλιοθήκη της **JQUERY.** Με την JQUERY έχουμε την δυνατότητα να δημιουργούμε pointers τα οποία δένουμε με τα instances των classes όταν επαναχρησιμοποιούμε components εξασφαλίζοντας έτσι ότι το σύστημα πάντα θα φέρνει τα σωστά δεδομένα στην παρουσίαση.

Server

Το κομμάτι του server αποτελείται μια SQL Βάση Δεδομένων.
Η Βάση περιέχει δύο (2) πίνακες

Α)

ZEUS\SQL2019.Thesis - dbo.Employee | ZEUS\SQL2019.Thesis - dbo.Program

	Employee_ID	Surname	Name	Username	Password
	1	Simistiras	Socrates	user-ssimis	#ssimis!123
	3	Simistiras	Hector	user-esimis	#esimis!123
	4	Papadopoulos	Angelos	user-angpap	#angpap!123
▶*	NULL	NULL	NULL	NULL	NULL

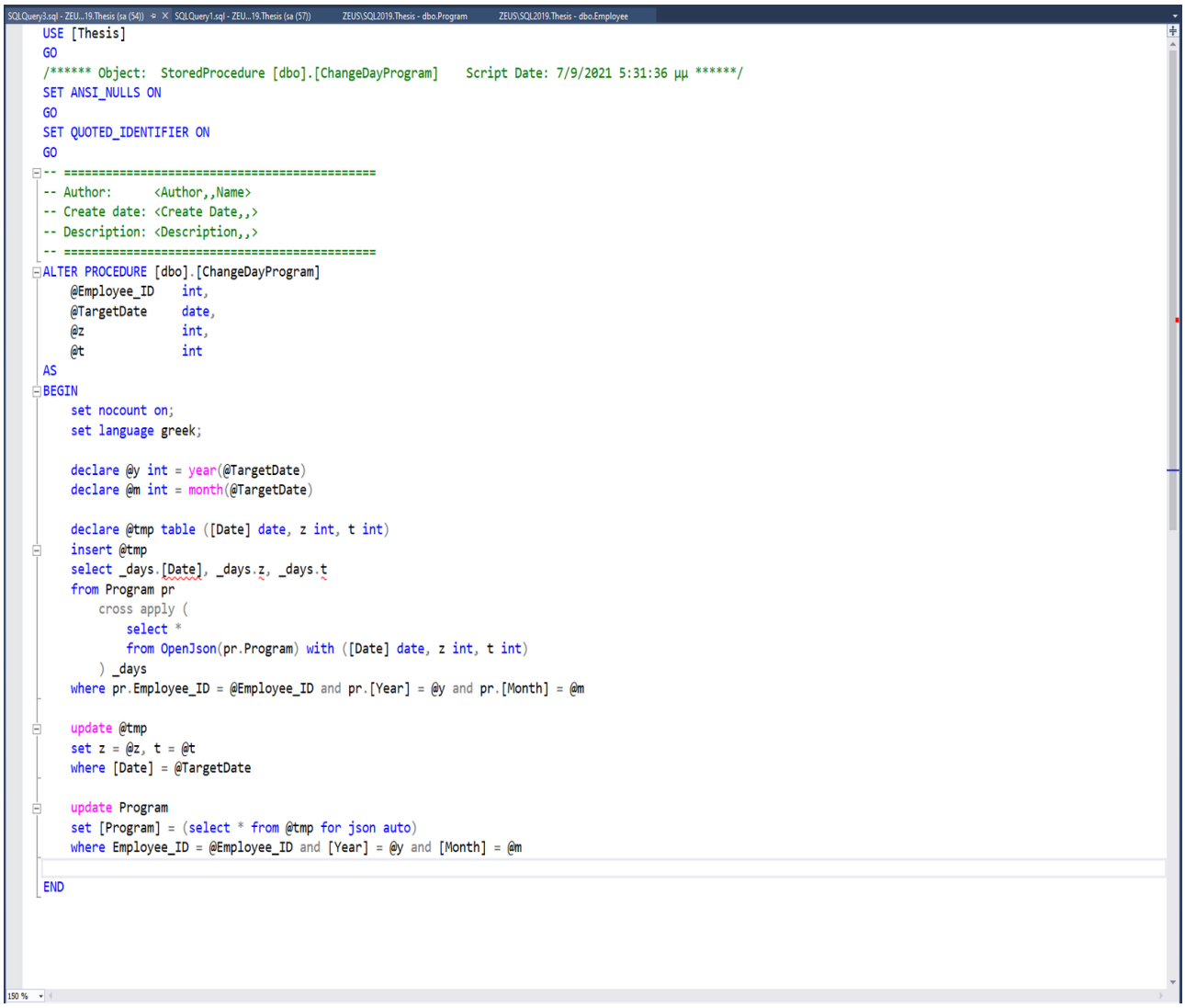
Ένας πίνακας για τους εργαζόμενους με τα στοιχεία πρόσβασης τους στην εφαρμογή καθώς και ένα Employee_ID που είναι και το κύριο κλειδί. Χρησιμεύει στο να ταυτοποιηθεί ο χρήστης κατά την διάρκεια του Login.

Β)

ZEUS\SQL2019.Thesis - dbo.Program

	Program_ID	Employee_ID	Year	Month	Program
▶	1	1	2021	8	[{"Date": "2021-08-01", "z"...
	2	1	2021	9	[{"Date": "2021-09-01", ...
	3	3	2021	8	[{"Date": "2021-08-01", "z"...
	4	3	2021	9	[{"Date": "2021-09-01", ...
	7	4	2021	8	[{"Date": "2021-08-01", "z"...
	8	4	2021	9	[{"Date": "2021-09-01", ...
*	NULL	NULL	NULL	NULL	NULL

Ένας πίνακας που αποθηκεύει το πρόγραμμα εργασίας για τους χρήστες και έχει ως ξένο κλειδί το Employee_ID που εξυπηρετεί στην αντιστοίχιση του προγράμματος εργασίας με τον αντίστοιχο χρήστη, έτσι όταν ζητήσουμε το πρόγραμμα εργασίας για έναν χρήστη θα στείλουμε σαν παραμέτρους το ID του χρήστη και τον μήνα στον οποίο αναφερόμαστε για να μας επιστραφεί το πρόγραμμα που ζητάμε.
Τέλος υπάρχει ο κατάλληλος κώδικας για να αλλάξει τις εγγραφές στην βάση όπως αιτήθηκαν από τον εργαζόμενο όταν αυτό ζητηθεί από το service.



```
USE [Thesis]
GO
/****** Object:  StoredProcedure [dbo].[ChangeDayProgram]    Script Date: 7/9/2021 5:31:36 μμ ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[ChangeDayProgram]
    @Employee_ID    int,
    @TargetDate     date,
    @z              int,
    @t              int
AS
BEGIN
    set nocount on;
    set language greek;

    declare @y int = year(@TargetDate)
    declare @m int = month(@TargetDate)

    declare @tmp table ([Date] date, z int, t int)
    insert @tmp
    select _days.[Date], _days.z, _days.t
    from Program pr
        cross apply (
            select *
            from OpenJson(pr.Program) with ([Date] date, z int, t int)
        ) _days
    where pr.Employee_ID = @Employee_ID and pr.[Year] = @y and pr.[Month] = @m

    update @tmp
    set z = @z, t = @t
    where [Date] = @TargetDate

    update Program
    set [Program] = (select * from @tmp for json auto)
    where Employee_ID = @Employee_ID and [Year] = @y and [Month] = @m

END
```

Ανάπτυξη και Λειτουργία Συστήματος

Δομική Ανάλυση

Όπως αναφέρθηκε στην Εισαγωγή, η εφαρμογή αναπτύχθηκε ως SPA (Single Page Application), ένας τρόπος ανάπτυξης εφαρμογών σε μοναδικό HTML αρχείο που δεν απαιτεί redirect όταν ο χρήστης επιλέγει να αλλάξει σελίδα. Όπως αναλύσαμε σε προηγούμενο κεφάλαιο αυτή η τεχνική απαιτεί την χρήση των components που δημιουργήσαμε με την βοήθεια της Knock-out.js και του Webpack.

Η **index.html** περιέχει 2 μοναδικά Divisions:

A) Το **“Page Host”**

```
<div id="PageHost" style="position:absolute; left:0; top:0; right:0; bottom:0; " data-bind="component:{name:'page-host'}"></div>
```

B) Το **“Printing Host”**

```
<div id="printingHost" style="padding:0px; margin:0px !important; outline:none;">  </div
```

Το Page Host είναι το division το οποίο θα φιλοξενήσει όλα τα components της εφαρμογής μας, ενώ το αρχικά μη ορατό Printing Host θα δέχεται όλα τα δεδομένα προς εκτύπωση όταν το επιλέγει ο χρήστης.
Παρατηρούμε πως το Page Host είναι τοποθετημένο έτσι ώστε να καλύπτει το 100% της οθόνης και περιέχει το "Data-bind" attribute το οποίο φιλοξενεί το component που αρχικοποιεί την σελίδα που προβάλλεται στον χρήστη. Κάθε component που περνάμε στο division αυτό αντιστοιχεί σε μία καινούργια σελίδα (e-port Site, Login, Dashboard). Για να μπορούμε να έχουμε πρόσβαση στο root page από όλα τα components έτσι ώστε να επιτρέπουμε στον χρήστη να πλοηγηθεί στο σύστημα έχουμε στήσει έναν μηχανισμό στην main.js η οποία λειτουργεί ως entry point για την Javascript και δέχεται σαν imports όλα τα javascript αρχεία και κατά συνέπεια τα components του προγράμματος, άρα έχει πάντα πρόσβαση σε αυτά.

```
class App{

    constructor(){
        this.currentPage = ko.observable();
           }
    _selectPage(name){
        this.currentPage(name);
    }
}
```

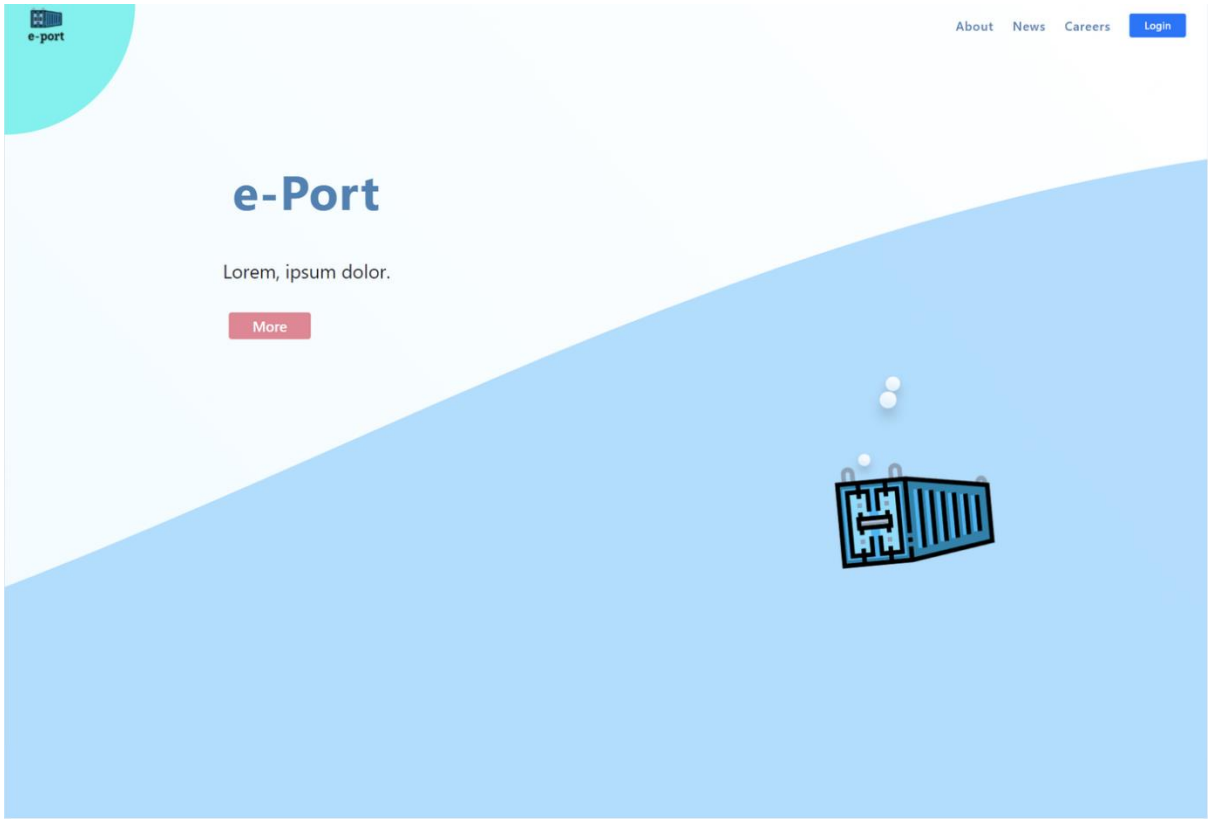
Έτσι από όλα τα components μπορούμε να καλέσουμε την _selectPage χρησιμοποιώντας το component name που θέλουμε και να αλλάξουμε το περιεχόμενο προβολής χωρίς να χρειαστεί να κάνουμε redirect τον χρήστη σε διαφορετικό HTML αρχείο.
Την ίδια ακριβώς τεχνική έχουμε χρησιμοποιήσει και για το Dashboard το οποίο σαν root page περιέχει ένα menu όπου κάθε εγγραφή του είναι «δεμένη» με ένα component για να αλλάζει το περιεχόμενο σε ένα εσωτερικό division.

Ιστοσελίδα e-Port

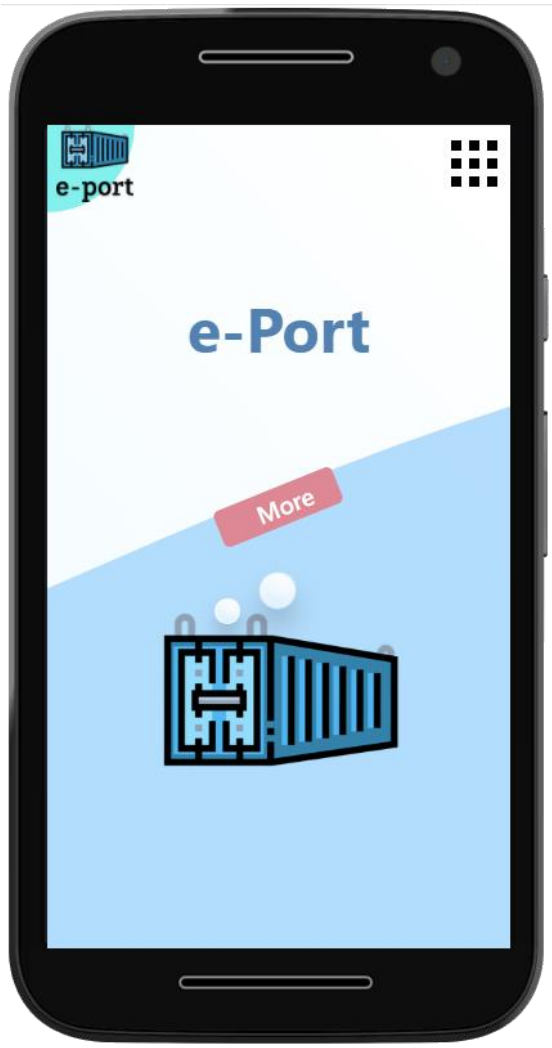
Η αρχική ιστοσελίδα που οδηγείται ο χρήστης πριν ξεκινήσει την διαδικασία πιστοποίησης του, σε ένα πραγματικό σενάριο θα αποτελούσε πηγή ενημέρωσης για τις δραστηριότητες της εταιρίας που αφορούν τους εργαζόμενους καθώς και τους ενδιαφερόμενους προς εργασία, στην δική μας περίπτωση όμως η μοναδική λειτουργικότητα που παρέχεται είναι η μετάβαση στην σελίδα του Login.
Όπως ήδη αναφέραμε η εφαρμογή και όλα τα components που την αποτελούν είναι "fully responsive", δηλαδή με τον κατάλληλο CSS κώδικα εξασφαλίσαμε πως ανταποκρίνονται σε οποιοδήποτε μέγεθος συσκευής. Αρχικά σχεδιάσαμε το component για μικρές συσκευές (smartphone) και στην συνέχεια με την βοήθεια του

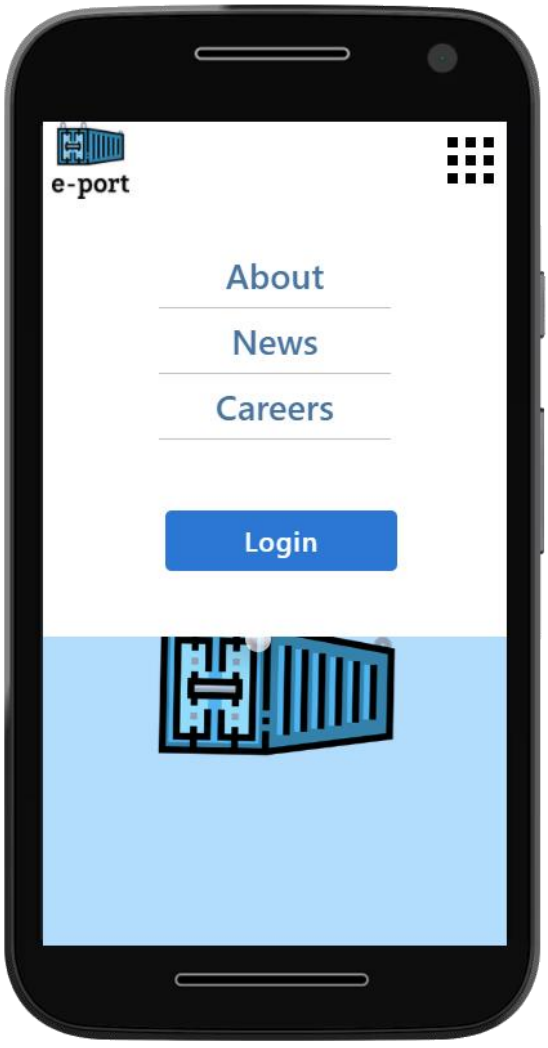
```
@media only screen and (min-width: /*width in pixels*/) { /*some code*/ }
```

που ενεργοποιεί τον CSS κώδικα όταν το μέγεθος της οθόνης του χρήστη είναι μέσα στο εύρος των pixel που ορίσαμε διαφοροποιούμε την σχεδίαση έτσι ώστε να είναι συμβατό και με μεγαλύτερες συσκευές.
Στην παρακάτω εικόνα μπορούμε να δούμε πως η ίδια σελίδα ανταποκρίνεται σε δύο διαφορετικές συσκευές.


e-port
About
News
Careers
Login
e-Port
Lorem, ipsum dolor.
More

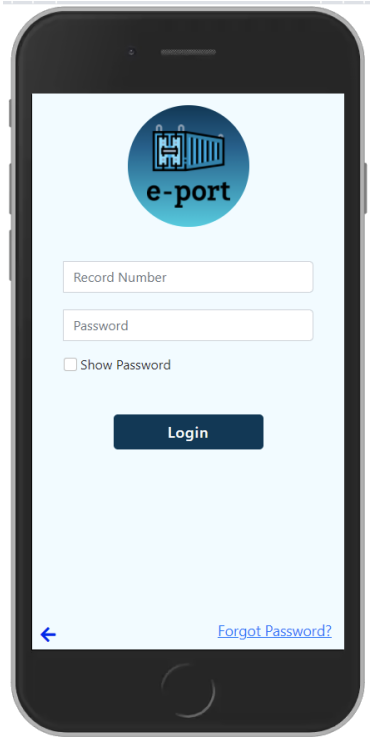
8.Desktop


e-port
e-Port
More

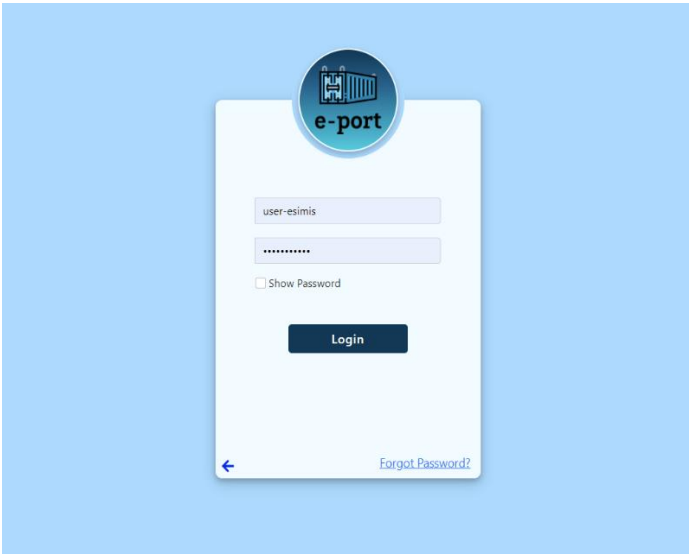

e-port
About
News
Careers
Login

Login

Σχεδιαση



iPhone 8



Desktop

Διαδικασία και Λειτουργικότητα

Σε αυτό το στάδιο ο χρήστης καλείται να επαληθεύσει την ταυτότητα του για να αποκτήσει πρόσβαση στο πρόγραμμα εργασίας του και τις λοιπές λειτουργίες του portal μέσω μιας φόρμας Login. Το σύστημα θα συγκρίνει τα στοιχεία που παρέχονται από τον χρήστη με αυτά που περιέχονται στην βάση δεδομένων και θα επιτρέψει ή θα απορρίψει την πρόσβαση του στο σύστημα. Αυτή την σύγκριση αναλαμβάνει το service, εμείς από τον client θα πρέπει απλά να στείλουμε ένα HTTP request που θα περιέχει τα στοιχεία που δόθηκαν. Για τα requests επιλέξαμε την χρήστη του **async / await** από τις " **.then** " συναρτήσεις καθώς και την fetch(), native συνάρτηση από τις εξωτερικές βιβλιοθήκες όπως το "**Axios**".
Η συνάρτηση fetch() περιέχει δύο (2) ορίσματα, αρχικά την διεύθυνση με την οποία επιθυμούμε να επικοινωνήσουμε και στην συνέχεια ένα object που περιέχει τις πληροφορίες σχετικά με το request που επιθυμούμε να πραγματοποίησουμε.
Στην περίπτωση μας επιθυμούμε να στείλουμε στο service τα στοιχεία του χρήστη έτσι ώστε να επιβεβαιωθεί η ταυτότητα του. Αρχικά λοιπόν κατασκευάζουμε το object που θα περιέχει αυτά τα στοιχεία καθώς και θα ενημερώνει το service για το τι τύπου δεδομένα περιέχονται.

```
let options = {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify({un,ps})
            }
```

Όπως βλέπουμε το request θα είναι τύπου "POST" , ο τύπος δεδομένων είναι "application/json" ενώ στο body του request περιέχονται το username & password που έδωσε ο χρήστης. Στην συνέχεια υλοποιούμε την συνάρτηση που θα καλέσει την fetch() κάνοντας τον απαραίτητο έλεγχο στο promise που επιστρέφεται έτσι ώστε σε περίπτωση αποτυχίας να αναγνωρίσουμε το error μέσω του StatusText που επιστρέφεται.

```
    async _login(un,ps) {
        let request = "https://thesis-api.azurewebsites.net/Thesis/Login"
        try {
            let options = {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify({un,ps}) //username & password
            }
            let response = await fetch(request, options);
            if (response.ok) {  // Έλεγχος Promise
                let serviceData = await response.json();
                return serviceData; // Επιστροφή ID
            }
            else
                throw Error(response.statusText);
        } catch (e) {
            this.showLoader(false);
            alert(e.message);
        }
    }
```

Σε περίπτωση επιτυχίας αποθηκεύουμε το username, password και ID σε observable μεταβλητές σε ένα αρχείο (**Helper.js**) που παρακολουθείται από όποιο component επιλέξουμε μέσω import έτσι ώστε για κάθε χρήση του service να υπάρχει πρόσβαση στα στοιχεία που δόθηκαν κατά το login. Αυτό μας εξασφαλίζει πως μπορούμε να επιβεβαιώνουμε την εγκυρότητα του χρήστη μέσω του header στα επόμενα request μας όπως ορίζει η μέθοδος του Basic Authentication αλλά και να έχουμε διαθέσιμο το ID του συνδεδεμένου χρήστη που λειτουργεί σαν κλειδί στην βάση δεδομένων.

```
    this.showLoader(true);
    this.emp_ID= await this._login(this.username(),this.password());
    if (this.emp_ID!=null){
        config.user_id=this.username(); //Απόθήκευση Username
        config.user_pwd=this.password(); //Απόθήκευση Password
        config.employee_id=this.emp_ID; //Απόθήκευση ID
        this.navigateDashboard(); //Αλλαγή component
    }
}
```

Τέλος αλλάζουμε το περιεχόμενο του **data-bind** που φιλοξενεί το component του login και μεταφερόμαστε στο Dashboard.

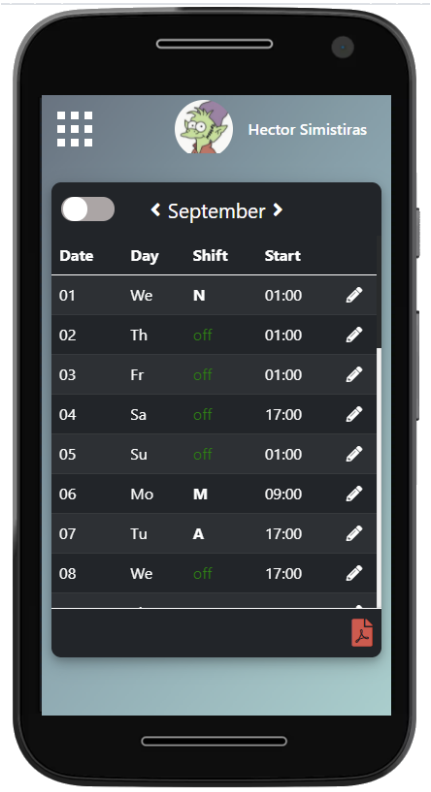
```
navigateDashboard(){
    this.showLoader(false);
    config.dash_body();
    this._PageRoot.currentPage('host-dashboard');
}
```

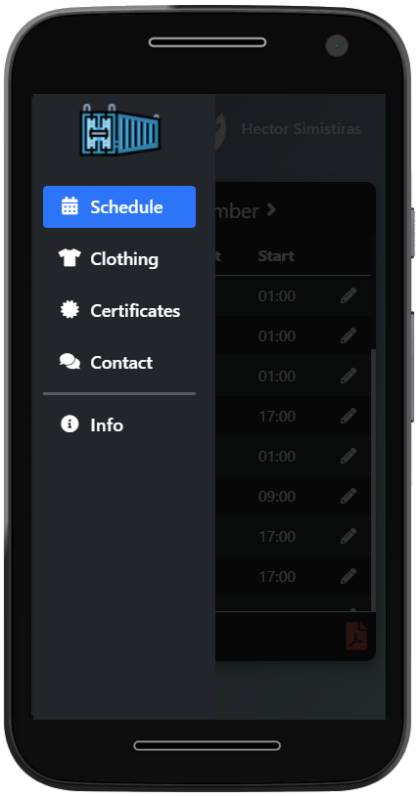
Θα πρέπει να σημειωθεί πως η μέθοδος του basic authentication σε συνδυασμό με το HTTP πρωτόκολλο δεν παρέχει επαρκή ασφάλεια δεδομένων διότι τα δεδομένα μεταφέρονται στο σύρμα χωρίς κρυπτογράφηση. Αυτό σημαίνει πως οποιοσδήποτε επιχειρήσει να «ακούσει» την πόρτα που χρησιμοποιούμε για να επικοινωνήσουμε με το service μπορεί να αποκτήσει πρόσβαση στα δεδομένα που μεταφέρουμε.

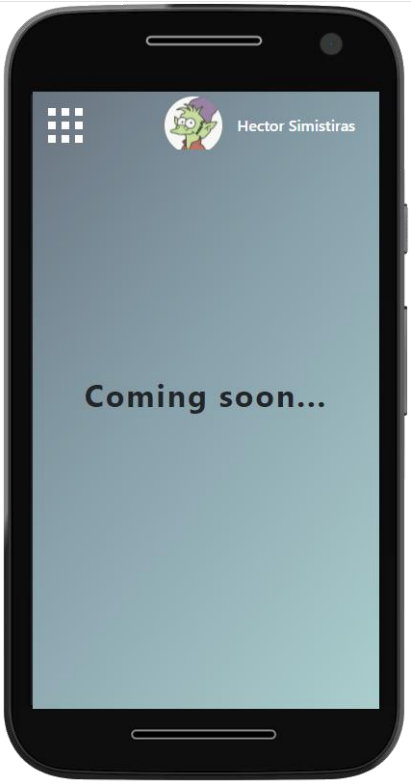
Info Portal

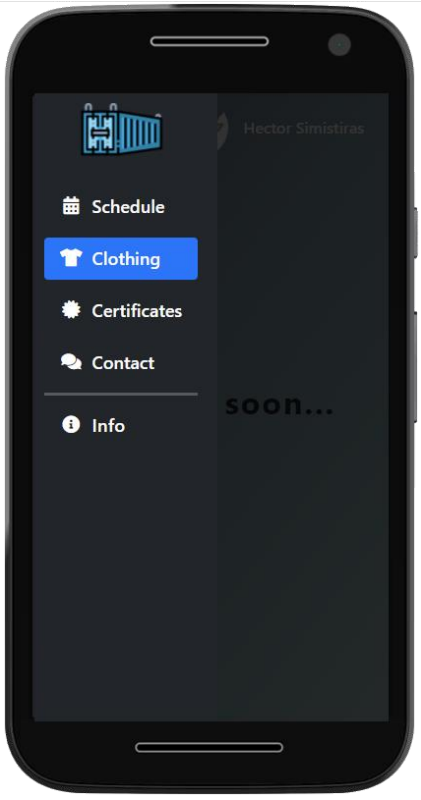
Σχεδίαση

Το Info Portal πρόκειται για το component που προβάλλει τις πληροφορίες ή/και λειτουργίες που χρειάζεται ο εκάστοτε εργαζόμενος. Στο app μας έχει υλοποιηθεί η προβολή προγράμματος εργασίας του τρέχον και επόμενου μήνα καθώς και ένας μηχανισμός που επιτρέπει στον εργαζόμενο να αλλάζει το πρόγραμμα εργασίας του κατά βούληση.
Το Portal φυσικά όπως και τα υπόλοιπα components προσαρμόζεται σε διαφορετικά μεγέθη συσκευών όπως βλέπουμε παρακάτω


Hector Simistiras
September
Date Day Shift Start
01 We N 01:00
02 Th off 01:00
03 Fr off 01:00
04 Sa off 17:00
05 Su off 01:00
06 Mo M 09:00
07 Tu A 17:00
08 We off 17:00


Schedule
Clothing
Certificates
Contact
Info


Hector Simistiras
Coming soon...


Schedule
Clothing
Certificates
Contact
Info
soon...

MOTO G4

1.Desktop View

Πατώντας την ένδειξη το εικονίδιο του «μολυβιού» στα δεξιά κάθε ημέρας ο χρήστης μπορεί να αιτηθεί την αλλαγή στο πρόγραμμα εργασίας του ενώ επίσης δίνεται η δυνατότητα εκτύπωσης.

Δεδομένα

Τα δεδομένα που θα καλέσουμε βρίσκονται στην βάση ως ένα array από objects, όπου κάθε object αντιπροσωπεύει μία ημέρα του μήνα.

```
[
    {"Date": "2021-09-01", "z":2, "t":0 },
    {"Date": "2021-09-02", "z":3, "t":1 },
    {...},
    {...}
 ]
```

Η εγγραφή **“z”** αντιπροσωπεύει την «ζώνη» εργασίας για την συγκεκριμένη ημέρα και μπορεί να πάρει τις εξής τιμές:

- **1** Για πρωινή ζώνη
- **2** Για απογευματινή ζώνη
- **3** Για Βραδινή ζώνη

Η εγγραφή **“t”** μπορεί να πάρει τις τιμές 0 ή 1 και υποδεικνύει το εάν εργάζεται ή όχι ο χρήστης την συγκεκριμένη ημέρα
Η ημερομηνία **“Date”** φυσικά θα χρησιμεύσει στο να προσεγγίσουμε την μέρα το μήνα και το έτος στο οποίο αναφέρεται το πρόγραμμα εργασίας.

Επεξεργασία Δεδομένων

Τα δεδομένα θα πρέπει να προετοιμαστούν κατάλληλα έτσι ώστε ο τελικός χρήστης να μπορεί να δει στην οθόνη του τα εξής στοιχεία :

- Τον Μήνα στον οποίο αναφέρεται το πρόγραμμα
- Την ημέρα αριθμητικά (01, 02, …)
- Την ημέρα ολογράφως (Μόνο τα δύο(2) πρώτα γράμματα)
- Την βάρδια που εργάζεται ή αν έχει ρεπό
- Την ώρα που ξεκινάει η βάρδια του

Για την υλοποίηση αυτού έχουν δημιουργηθεί δύο κλάσεις

- Η **WorkDay**

  Το οποία περιέχει τα απαραίτητα «εργαλεία» που θα τροποποιήσουν τα δεδομένα που θα λάβει στον constructor.
- Η **Work_Collection**

  Η οποία αναλαμβάνει να καλέσει, να δημιουργήσει τα instance της WorkDay και με την βοήθεια της knockout.js να κάνει τα τροποποιημένα δεδομένα ορατά στο σύστημα.

Ο Constructor της κλάσης **WorkDay** θα απομονώσει για μια ημέρα τις εγγραφές για **Date**, **“z”** , **“t”** και στην συνέχεια με τις κατάλληλες συναρτήσεις θα χρησιμοποιήσει μια μεταβλητή για να αποθηκεύσει:

Τα νούμερα της ημέρας

```
this.day_num = this._day_num();
```

Τα δύο πρώτα γράμματα της ημέρας

```
this.day_string = this._day_string();
```

Την βάρδια στην οποία εργάζεται ή την ένδειξη “off” για ρεπό

```
this.shift = this._shift();
```

Την ώρα εκκίνησης για την βάρδια

```
this.shift_start = this._shift_start();
```

Όταν δημιουργηθεί ένα instance για την **Work_Collection** ο constructor της θα πάρει σαν όρισμα ολόκληρο το ακατέργαστο μηνιαίο πρόγραμμα και θα κάνει τα εξής

Πρώτον :

Μέσω της συνάρτησης

```
get_month() {
    let month = new WorkDay(this.monthly_schedule[0]);
    month = month._month();
    this.cur_month(month);
}
```

Θα δημιουργήσει ένα μοναδικό instance της **WorkDay** έτσι ώστε να πάρει τον μήνα στον οποίο αναφέρεται το πρόγραμμα και θα τον κάνει ορατό στο σύστημα

```
this.cur_month = ko.observable();
```

Δεύτερον :

Μέσω της συνάρτησης

```
    build_Days() {
        let entries = [];
        for (let i = 0; i < this.monthly_schedule.length; i++) {
            entries.push( new WorkDay(this.monthly_schedule[i]));
        }
        this.work_list(entries);
    }
```

Θα δημιουργήσει τόσα instance της **WorkDay** όσα και τα object που περιέχει το array που πήρε σαν είσοδο, δηλαδή όσες μέρες περιέχει ο συγκεκριμένος μήνας και στην συνέχεια κάθε instance θα γίνει ένα object ενός καινούργιου array που θα είναι επίσης ορατό από το σύστημα

```
this.work_list = ko.observableArray([]);
```

Πλέον τα δεδομένα μας είναι έτοιμα για προβολή.
Υποσημείωση:
Α) Χρησιμοποιούμε τις observable μεταβλητές της knockout γιατί τα δεδομένα παρουσιάζονται δυναμικά και ενδέχεται να αλλάξουν ζωντανά.
Β) Το array με τα instance της **WorkDay** φυσικά έχει πολύ περιττή πληροφορία αλλά δεν χρειάζεται να το επεξεργαστούμε περαιτέρω μιας και η knockout όταν κληθεί στην HTML θα επιλέξει μόνο τα ορίσματα που θα ζητήσουμε χωρίς να χρειαστεί να κάνει προσπέλαση όλων των δεδομένων.

Παρουσίαση Δεδομένων

Για την παρουσίαση των δεδομένων θα αναφερθούμε μόνο στο ουσιώδες κομμάτι του προγράμματος εργασίας. Στο προηγούμενο κεφάλαιο μιλήσαμε για την επεξεργασία των δεδομένων στο Model-View κομμάτι του συγκεκριμένου component, η παρουσίαση φυσικά γίνεται στην HTML που είναι το View κομμάτι του component. Έχουμε δημιουργήσει έναν πίνακα και φτιάχνοντας ένα μοναδικό row μπορούμε να δημιουργήσουμε δυναμικά όσα rows χρειαζόμαστε για την παρουσίαση του προγράμματος. Ο κώδικας που θα το υλοποιήσει είναι ο εξής:

```
            <tbody>
                <!-- ko if:$component.workCollection -->
                <!-- ko foreach:$component.workCollection().work_list -->
                <tr>
                    <td data-bind="text: day_num"></td>
                    <td data-bind="text: day_string"></td>
                    <td data-bind="text: shift, style:dayNumStyle"></td>
                     <td data-bind="text: shift_start"></td>
                </tr>
                <!-- /ko -->
                <!-- /ko -->
            </tbody>
```

Βλέπουμε λοιπόν πως εντός του body του πίνακα η knockout ενεργοποιείται μέσα από τα σχόλια της HTML.

```
                <!-- ko if:$component.workCollection -->
```

Η **“ko if”** εξετάζει αν συνθήκη που παραθέτουμε είναι αληθής και μόνο τότε υλοποιεί τον κώδικα τον οποίο περικλείει έως το closing tag

```
                <!-- /ko -->
```

Αυτό μας εξασφαλίζει πως το περιεχόμενο του πίνακα θα φορτωθεί μόνο όταν η observable μεταβλητή που περιέχει τα δεδομένα του προγράμματος εργασίας δεν είναι κενή, αποφεύγοντας έτσι ανεπιθύμητες συμπεριφορές στην παρουσίαση. Αμέσως μετά τον έλεγχο χρησιμοποιούμε την

```
<!-- ko foreach:$component.workCollection().work_list -->
```

Η **“ko foreach:”** χρησιμοποιείται για να προσπελάσουμε arrays και θα εκτελέσει τον κώδικα μέχρι το closing tag της για κάθε ένα όρισμα που περιέχει το array, στην δική μας περίπτωση θα τον εκτελέσει για όσες ημέρες περιέχει ο μήνας στον οποίο αναφερόμαστε.
Βλέπουμε παρακάτω πως για κάθε κελί υπάρχει ένα **“data-bind”** το οποίο αναφέρεται σαν **text** και δείχνει ένα όρισμα μέσα στο object για το οποίο εκείνη την στιγμή γίνεται προσπέλαση από την knockout. Έτσι επιτυχώς για κάθε ημέρα θα φορτωθούν τα ανάλογα δεδομένα σε κάθε κελί του πίνακα.

Λήψη Δεδομένων

Μιλήσαμε για την επεξεργασία και παρουσίαση των δεδομένων που αφορούν το πρόγραμμα εργασίας αλλά δεν έχουμε αναφερθεί στο πως φτάνουν αυτά τα δεδομένα στο πρόγραμμα μας από το database.
Την μεταφορά των δεδομένων αναλαμβάνει το service, δεν θα ασχοληθούμε όμως με το πως γίνεται το «σερβίρισμα» αλλά με πως υλοποιούμε την κλήση του από τον client.
Για να κληθεί το πρόγραμμα έχουμε δημιουργήσει την συνάρτηση

```
async _fetchProgram(IsNext) {...}
```

Λόγο όγκου κώδικα θα περιοριστούμε στο να μιλήσουμε για τα βήματα που εκτελέστηκαν αντί να κάνουμε λεπτομερή ανάλυση του κώδικα. Η **_fetchProgram()** θα εκτελεστεί με το που περάσουμε από το Login στο Dashboard ενώ το **IsNext** πρόκειται για μια Boolean μεταβλητή και σκοπός της είναι να επιλέξει ποιόν μήνα θα ζητήσει από το service. Μετά την είσοδο στην συνάρτηση θα γίνει έλεγχος, για τιμές false, undefined ή true και θα ζητηθεί το πρόγραμμα για τον τρέχον ή τον επόμενο μήνα.
Το επόμενο βήμα είναι να δημιουργηθεί το endpoint, η διεύθυνση δηλαδή την οποία θα καλέσει η fetch την συγκεκριμένη λειτουργία του service.
Στην δική μας περίπτωση το endpoint θα είναι της μορφής

```
https://thesis-api.azurewebsites.net/Thesis/GetProgram/ID/Year/Month
```

Παρατηρούμε ότι το ID που αποθηκεύσαμε για τον χρήστη κατα την διάρκεια του Login θα το εισάγουμε εδώ έτσι το service να βρει τον χρήστη που αναζητάμε.
Σαν **“Year”** πάντοτε θα εισάγουμε το τωρινό έτος, ενώ το **Month** θα συμπληρώνεται ανάλογα με την τιμή της **IsNext** μεταβλητής.
Τώρα λοιπόν που έχουμε κατασκευάσει το endpoint που θα χρησιμοποιήσουμε μπορούμε να χρησιμοποιήσουμε την fetch,

```
    async _callFetchService(endpoint){
        try {
            let options = {
                method: "GET",
                credentials: "include",
                headers: {
                    Authoriza-
tion: "Basic " + btoa(config.user_id + ":" + config.user_pwd)
                }
            }

            let response = await fetch(request, options);
            if (response.ok) {
```

```
                if (isJSON){
                    let serviceData = await response.json();
                    this.curtain_table(false);
                    return serviceData;

                                }
                            }
            else
                throw Error(response.statusText);
        }
        catch (e) {
            this.curtain_table(false);
            console.log(e.message);
        }
    }
```

Παρατηρούμε επίσης πως το request μας έχει παραλήπτη ένα διαφορετικό domain από αυτό που βρίσκεται η εφαρμογή, κάτι που δεν επιτρέπεται όταν μιλάμε για δεδομένα και όχι για εικόνες σε ένα HTTP request. Εκεί έρχεται να παίξει τον ρόλο του η πολιτική του **CORS** (Cross Origin Resource Sharing) που υπό συνθήκες επιτρέπει την λειτουργία αυτή. Έτσι λοιπόν για να αποκτήσουμε πρόσβαση στα δεδομένα που ζητάμε και σύμφωνα με το Basic authentication απαιτείται να συμπεριλάβουμε ένα header στο HTTP request της μορφής:

```
Authorization: Basic YWRtaW46cEBzNXcwcmQ=
```

Δηλαδή την λέξη **Basic** και την **btoa()** κωδικοποίηση (64bit) του username+password.
Τελικά το service επιστρέφει το πρόγραμμα εργασίας σε μορφή string , και αφού μετατρέψουμε το response σε μορφή JSON μπορούμε να προσπελάσουμε εύκολα τα δεδομένα και να εκκινήσουμε την διαδικασία επεξεργασίας και προβολής τους δημιουργώντας ένα instance της κλάσης WorkCollection.

Αλλαγές Προγράμματος

Ο χρήστης εκτός από την προβολή του προγράμματος εργασίας του έχει το δικαίωμα να αιτηθεί τις εξής αλλαγές:

- Από ρεπό σε βάρδια
- Από βάρδια σε διαφορετική βάρδια
- Από βάρδια σε ρεπό

Σε οποιοδήποτε πραγματικό σενάριο οποιοδήποτε αίτημα αλλαγής θα έπρεπε να εγκριθεί από τον υπεύθυνο προσωπικού της επιχείρησης, στην δική μας όμως περίπτωση καθαρά για λόγους ευκολίας οι αιτήσεις δεν χρειάζονται έγκριση και δεν υπάρχει κανένας περιορισμός στις αλλαγές εκτός από έναν υποτυπώδη έλεγχο για παρελθοντικές ημερομηνίες. Ο χρήστης πατώντας το τελευταίο κελί του πίνακα -μολύβι- για κάθε ημέρα καλεί την συνάρτηση

```
    grab_day(day_obj) {
        this.day_obj(day_obj);
          let date_current=new Date();

        if (Date.parse(date_current)<Date.parse(day_obj.date)) {

            let temp = day_obj.date.split("-");
            this.selected_day(temp[2]);
```

```
            this.selected_workday(day_obj);
            this.working(day_obj.t);
            this.selected_shift(day_obj.z);
            this.curtain_lift();
        }
        else {
            config._triggerPast();
        }
    }
```

η οποία με την βοήθεια της Knockout δέχεται σαν όρισμα τα data που χρησιμοποιήθηκαν για να χτιστεί η συγκεκριμένη «γραμμή» του πίνακα, δηλαδή το instance της κλάσης WorkDay που αντιστοιχεί στην ημέρα του μήνα που επιλέχθηκε. Έπειτα κάνει έλεγχο για το αν ο χρήστης επέλεξε παρελθοντική ημερομηνία ή όχι. Αν η ημερομηνία που επιλέχθηκε είναι παρελθοντική το σύστημα εμφανίζει το αντίστοιχο alert και δεν εκκινεί καμία διαδικασία αλλαγής. Εάν η ημερομηνία είναι έγκυρη (μελλοντική) τότε απομονώνονται τα δεδομένα που αναφέρονται στην συγκεκριμένη ημέρα και καλείται η συνάρτηση

```
curtain_lift(){...}
```

που εμφανίζει το παράθυρο που επιλέγονται οι αλλαγές με τα στοιχεία που χρειαζόμαστε (πχ σε περίπτωση που ο εργαζόμενος έχει ρεπό την μέρα που επέλεξε να τροποποιήσει το πρόγραμμα του το κουμπί για αίτημα ρεπό δεν θα εμφανιστεί).
Εάν ο χρήστης επιθυμεί να αλλάξει την βάρδια του για την συγκεκριμένη ημέρα μέσω ένος select θα επιλέξει την βάρδια την οποία επιθυμεί, κάνοντας Submit θα καλέσει την συνάρτηση

```
request_schedule_change(condition){…}
```

η οποία θα αποθηκεύσει την καινούργια βάρδια στην ίδια μεταβλητή που αποθήκευσε την βάρδια στην οποία εργάζεται κατά την εκκίνηση της διαδικασίας και θα σχηματίσει το endpoint που θα χρειαστούμε για να κληθεί η μέθοδος του service που αναλαμβάνει να αλλάξει την εγγραφή στην βάση δεδομένων και πρέπει να έχει την μορφή

```
"https://thesis-api.azurewebsites.net/Thesis/ID/ChangeDayProgram/Date/z/num"
```

Όπου
“ID” το μοναδικό ID του χρήστη που έχουμε αποθηκεύσει κατά το login
“Date” είναι η ημερομηνία στην οποία αναφερόμαστε σε **ISO 8601** μορφή
“z” η καινούργια βάρδια που επιλέχθηκε
“num” παίρνει 2 τιμές , 0 ή 1 που προκύπτουν από την “condition” Boolean που περάστηκε στην συνάρτηση request_schedule_change(condition){…} κατά την αρχική της κλήση, η 0 αναφέρεται σε ρεπό και η 1 σε ημέρα εργασίας.
Αφού έχει σχηματιστεί το endpoint που χρειαζόμαστε για την κλήση ενεργοποιείται η

```
async _callFetchService(request) {...}
```

η οποία καλεί το service με τον ίδιο τρόπο που αναλύσαμε στην προηγούμενη ενότητα αλλά με το καινούργιο endpoint και το service ανανεώνει την τη βάση δεδομένων με τα καινούργια στοιχεία.
Τέλος καλείται η -πλέον γνωστή μας-

```
_fetchProgram(boolean);
```

Αφού επιλεγεί η Boolean μεταβλητή για τον μήνα στον οποίο αιτήθηκε η αλλαγή, το επικαιροποιημένο πρόγραμμα εμφανίζεται στην οθόνη του χρήστη και τέλος γίνεται trigger το notification που ενημερώνει για την επιτυχημένη αλλαγή και η διαδικασία ολοκληρώνεται
Πανομοιότυπη διαδικασία επιλέγεται και στην περίπτωση που ο χρήστης ζήτησε ρεπό μέ μοναδική διαφορά την οτι η Boolean condition που περάσαμε στην συνάρτηση

```
request_schedule_change(condition){…}
```

μεταφράζεται σε 0 στο endpoint.

Εκτύπωση Προγράμματος

Ένα πολύ χρήσιμο feature σε κάθε εφαρμογή που προβάλει δεδομένα είναι η δυνατότητα του χρήστη να έχει πρόσβαση σε αυτά ακόμη και εάν βρίσκεται εκτός σύνδεσης, αυτό επιτυγχάνεται με το να παρέχεται η δυνατότητα εκτύπωσης ή αποθήκευσης των δεδομένων σε μορφή pdf.
Παρότι υπάρχουν αρκετές open source βιβλιοθήκες που μας δίνουν την δυνατότητα να επιλέξουμε τα δεδομένα για τα οποία θέλουμε να δώσουμε την δυνατότητα στον τελικό χρήστη να αποθηκεύσει/εκτυπώσει αποφασίσαμε να εκμεταλλευτούμε την τεχνική του SPA και με την βοήθεια της **knockout** να δημιουργήσουμε μια μέθοδο που θα μας επιτρέπει να απομονώνουμε την πληροφορία που θέλουμε ανεξάρτητα από το που βρίσκεται στο πρόγραμμα και να την εκτυπώσουμε ή μετατρέψουμε σε αρχείο .**PDF**
Η μέθοδος του **Window.print()** δίνει προς εκτύπωση ή λήψη το HTML που προβάλλεται την δεδομένη στιγμή στον browser. Για να απομονώσουμε τα δεδομένα που χρειαζόμαστε χρησιμοποιήθηκε η εξής τεχνική. Η «είσοδος» για την παρουσίαση του προγράμματος είναι η index.html όπου εντός του body υπάρχει ένα division που έχουμε ονομάσει **Page Host** το οποίο φιλοξενεί το root component. Για να μπορούμε να απομονώνουμε δεδομένα δημιουργήσαμε ένα δεύτερο, άδειο by default και μη ορατό division στο οποίο θα αναφερθούμε ως **Printing Host**. Πρώτο βήμα είναι να «τυλίξουμε» τον κώδικα που θέλουμε να προωθήσουμε προς εκτύπωση σε ένα division και με την βοήθεια της Knockout και ενός πολύ απλού function να τα αποθηκεύσουμε σε μια μεταβλητή στον constructor της κλάσης που ανήκει το template

```
<div data-
bind="template: {afterRender: function(){$component.startWatching($element)}}">
    <!-- Data προς εκτύπωση / λήψη-->
</div>
```

```
startWatching(element){
    this.printDiv=element;
}
```

```
this.printDiv=null;
```

Το κουμπί/εικονίδιο/link που θα επιλέξουμε για την εκτύπωση

```
<a data-bind="click:print">print something!</a>
```

1. Θα καλεί μια συνάρτηση η οποία θα επιλέγει το division του **Printing Host**
2. Θα αδειάζει το περιεχόμενό του (μπορεί να περιέχει data από προηγούμενη εκτύπωση)
3. Θα παίρνει το template που αποθηκεύσαμε νωρίτερα, θα το τοποθετεί στον **Printing Host**
4. Θα καλεί την **window.print**

```
print() {
    let $ph = $("#printingHost");
    $ph.html("");
    let _html = $(this.printDiv).html();
    $ph.html(_html);
    setTimeout(window.print, 333);
}
```

Με την βοήθεια της CSS ορίζουμε πότε θέλουμε να γίνεται ορατό το **Printing Host** division και πότε το **Page Host**

```
@media screen {
  #PageHost {
```

```
    display: block; }
  #printingHost {
    display: none; } }

@media print {
  #PageHost {
    display: none !important; }
  #printingHost {
    display: block; } }
```

Συμπεράσματα και Βελτίωση Συστήματος

Συμπεράσματα

Στην παρούσα εργασία αναπτύχθηκε ένα application προβολής και αλλαγής προγράμματος εργασίας για εργαζόμενους σε βάρδιες εμπνευσμένο από συστήματα roster που κυκλοφορούν στην διεθνή αγορά. Για την υλοποίηση του προγράμματος χρησιμοποιήθηκαν οι γλώσσες προγραμματισμού, frameworks και βιβλιοθήκες:

- HTML
- Javascript (JQUERY & Knockout.js)
- CSS (Bootstrap)

Επιπρόσθετα χρησιμοποιήθηκε ο bundler “Webpack” καθώς και ο διαχειριστής πακέτων “NPM”, ενώ για την ανάπτυξη του project αξιοποιήθηκαν οι αρχές του αντικειμενοστραφούς προγραμματισμού ενώ η δομή του σχηματίστηκε με γνώμονα στην τεχνική του SPA (Single Page Application).

Σε επίπεδο προγραμματισμού η ολοκλήρωση της εργασίας βοήθησε στην κατανόηση της πολυπλοκότητας αλλά και της δομής που θα πρέπει να έχει ένα σύνθετο application, της επικοινωνίας Client / Service / Server, την λειτουργία του HTTP πρωτοκόλλου για την μεταφορά δεδομένων, την ανάγκη για ασφάλεια δεδομένων και τον σημαντικό ρόλο του προ-σχεδιασμού και ελέγχου ενός σύνθετου έργου λογισμικού. Επίσης κατά την διάρκεια της ανάπτυξης έγινε

κατανοητή η ευελιξία και η δυνατότητα συνεργασίας που παρέχεται στους προγραμματιστές μέσα από το εργαλείο “GIT” .
Σε επίπεδο σχεδίασης ενός roster συστήματος συμπεραίνουμε πως ο στόχος είναι η οργάνωση υπάρχοντος ανθρώπινου δυναμικού ως προς την εργασία σε δεδομένο επιχειρησιακό περιβάλλον. Λαμβάνοντας υπόψη ότι τα επιχειρησιακά περιβάλλοντα διαφοροποιούνται σημαντικά μεταξύ τους σε μια σειρά παραγόντων, έχουν ανάγκη να εφαρμόζουν διαφορετικούς τρόπους οργάνωσης του ανθρώπινου δυναμικού η βασική επιλογή που πρέπει να γίνει είναι ο τύπος προγράμματος που θέλουμε να δημιουργήσουμε (Low End – High End) ανάλογα με τον τύπο της επιχείρησης που εξυπηρετούμε.. Έτσι, τελικώς θα προκύψει ένα σύστημα που θα έχει την δυνατότητα να εξυπηρετεί ένα σταθερό εργασιακό περιβάλλον ή ένα σύστημα που θα έχει στόχο την δυναμική προσαρμογή του ανθρώπινου δυναμικού σε μια διαρκώς μεταβαλλόμενη ζήτηση. Τελικά για να προσδιοριστεί ο βαθμός πολυπλοκότητας και συνθετότητας του προγράμματος που υλοποιούμε θα πρέπει να αναγνωρίσουμε ποια είναι τα χαρακτηριστικά που απαιτούνται να συμπεριληφθούν στο σύστημα.

Βελτίωση

Το application που υλοποιήθηκε δεν πρόκειται για ένα ολοκληρωμένο σύστημα roster, αλλά μια εκδοχή προβολής και αλλαγής προγράμματος εργασίας εμπνευσμένη από συστήματα roster υιοθετώντας την λογική της εργασίας σε βάρδιες. Λαμβάνοντας υπόψη τα παραπάνω δεν κρίνεται απαραίτητο να εισηγηθούμε σε κάποιου τύπου βελτίωση όσον αφορά την πληροφορία που λαμβάνουμε και την οργάνωση της αλλά θα στοχεύσουμε κυρίως στο κομμάτι του προγραμματισμού.

Σχεδίαση

- Παρότι δημιουργήσαμε ένα πλήρως responsive περιβάλλον χρησιμοποιώντας pure CSS/SASS με το framework της Bootstrap καθώς και διαδραστικότητα με την βοήθεια της Javascript (JQUERY, Knockout) η εμφάνιση της εφαρμογής δεν ανταποκρίνεται στα στάνταρ που απαιτούνται για μια εφαρμογή που βγαίνει σε παραγωγή.

Βέλτιστες Τεχνικές

- Το συγκεκριμένο project είναι το πρώτο που δημιουργήσαμε εφαρμόζοντας την τεχνική του **SPA**, κατά την διάρκεια της υλοποίησης του διαπιστώθηκαν λογικά λάθη τα οποία αναγκαστήκαμε να διορθώσουμε χρησιμοποιώντας ανορθόδοξες τεχνικές που κάνουν την δουλειά μας αναίτια δυσκολότερη και δημιουργούν προβλήματα κατανόησης εάν κάποιος άλλος χρειαστεί να δουλέψει πάνω στο project μας. Πιο συγκεκριμένα κάποια παραδείγματα:
    1. Ο Page Host που είναι το division που αρχικοποιεί τα components που δέχεται στο σύστημα, δέχεται ένα παντελώς κενό component το οποίο περιέχει όλα τα υπόλοιπα, λόγω έλλειψης εμπειρίας και λάθος προσχεδιασμού
    2. Παρότι η εφαρμογή υιοθετεί τις αρχές του αντικειμενοστραφούς προγραμματισμού δεν αξιοποιήθηκε η κληρονομικότητα με αποτέλεσμα να μην υπάρχει άμεση πρόσβαση στο root page από όλα τα components που αποτελούν το σύστημα, το πρόβλημα λύσαμε δημιουργώντας ένα pointer στο root page το οποίο στην συνέχεια περάσαμε στο αρχείο Helper.js που παρακολουθείται globally από όλα τα components έτσι ώστε να μπορούμε να μετακινούμαστε κατά βούληση σε όποιο component θέλουμε.

Ασφάλεια

- Η ασφάλεια στα πληροφοριακά συστήματα μας εξασφαλίζει το απόρρητο των προσωπικών δεδομένων του εκάστοτε χρήστη και της πληροφορίας που μεταφέρεται. Στην εφαρμογή που αναπτύξαμε για την διαδικασία του Login χρησιμοποιήθηκε η μέθοδος του basic authentication η οποία μεταφέρει στο σύρμα το username και το password του χρήστη για να επιβεβαιώσει την ταυτότητα του. Για να είναι ασφαλής αυτή η μετάδοση απαιτείται κρυπτογράφηση την οποία αναλαμβάνει το HTTPS πρωτόκολλο. Για την χρήση του HTTPS πρωτοκόλλου απαιτείται η χρήση SSL πιστοποιητικού το οποίο δεν είχαμε διαθέσιμο. Για την μεταφορά δεδομένων χρησιμοποιήσαμε το HTTP πρωτόκολλο που δεν κρυπτογραφεί τα δεδομένα αλλά τα περνάει από το σύρμα σε μια Base-64 κωδικοποιημένη μορφή. Αν οποιοσδήποτε επιχειρήσει να «ακούσει» την πόρτα η οποία επικοινωνεί με το service μπορεί να υποκλέψει τα δεδομένα και να αντιστρέψει την κωδικοποίηση ακόμη και με εργαλεία που υπάρχουν online όπως το https://www.base64decode.org/.

Βιβλιογραφία

1) https://www.humanity.com/ (Πρόσβαση 23 Οκτωβρίου 2021)

2) https://www.kronos.com/products/kronos-virtual-roster (Πρόσβαση 23 Οκτωβρίου 2021)

3) http://www.e-addon.gr/ (Πρόσβαση 23 Οκτωβρίου 2021)

4) https://connecteam.com/ (Πρόσβαση 23 Οκτωβρίου 2021)

5) https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication (Πρόσβαση 23 Οκτωβρίου 2021)

6) https://devdocs.io/knockout (Πρόσβαση 23 Οκτωβρίου 2021)