2015

Web - Mobile Application with Grid Systems Services

ΑΝΑΦΟΡΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

ΔΗΜΗΤΡΗΣ ΜΠΙΚΑΣ

TUC | Technical university of Crete

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ

ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κατασκευή Web - Mobile Application Παροχής Υπηρεσιών του Grid
Web - Mobile Application with Grid Systems Services

Μπίκας Δημήτρης

Επιβλέπων καθηγητής: Σαμολαδάς Βασίλης

Χανιά 2015

Εξεταστική επιτροπή

Σαμολαδάς Βασίλης - Επίκουρος Καθηγητής

Χαλκιαδάκης Γεώργιος - Αναπληρωτής Καθηγητής

Πετράκης Ευριπίδης - Καθηγητής

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΚΟΝΕΣ

ΚΕΦΑΛΑΙΟ 1 - Εισαγωγή

Μία συστάδα υπολογιστών – cluster, αποτελείται από ένα σύνολο διασυνδεδεμένων υπολογιστών, οι οποίοι εργάζονται μαζί με τέτοιο τρόπο ώστε να θεωρούνται ένα ενιαίο σύστημα. Τα εξαρτήματα μίας συστάδας συνήθως είναι συνδεδεμένα μεταξύ τους μέσω γρήγορων τοπικών δικτύων - LAN, με κάθε κόμβο να έχει το δικό του στιγμιότυπο ενός λειτουργικού συστήματος. Στις περισσότερες περιπτώσεις όλοι οι κόμβοι χρησιμοποιούν το ίδιο υλικό - Hardware και το ίδιο λειτουργικό σύστημα.
Οι συστάδες συνήθως αναπτύσσονται για να βελτιώσουν την επίδοση και τη διαθεσιμότητα, με χαμηλό κόστος σε σχέση με μοναδιαίους υπολογιστές με την ίδια επίδοση και διαθεσιμότητα.
Οι συστάδες υπολογιστών δημιουργήθηκαν εξαιτίας της τάσης που υπήρχε στο χώρο των υπολογιστών για χαμηλού κόστους μικροεπεξεργαστές, υψηλής ταχύτητας δίκτυα και λογισμικό για κατανεμημένο υπολογισμό υψηλής απόδοσης. Εφαρμόζονται μία μεγάλη γκάμα συστημάτων που κυμαίνεται από μικρές συστάδες επιχειρήσεων με λίγους κόμβους έως κάποιους από τους γρηγορότερους υπέρ-υπολογιστές στον κόσμο όπως είναι ο Sequoia της IBM. Οι εφαρμογές που μπορούν να γίνουν όμως είναι περιορισμένες, καθώς το λογισμικό πρέπει να κατασκευασθεί για ένα συγκεκριμένο έργο. Οπότε είναι αδύνατον να χρησιμοποιήσουμε συστάδες υπολογιστών για κανονικά υπολογιστικά έργα.

Πλέγμα υπολογιστών – GRID, είναι μία συλλογή πόρων υπολογιστών από πολλαπλές τοποθεσίες για την επίτευξη ενός κοινού στόχου. Το πλέγμα μπορεί να θεωρηθεί ως ένα

κατανεμημένο σύστημα με μη δια-δραστικούς φόρτους εργασίας, οι οποίοι εμπλέκουν ένα μεγάλο αριθμό από αρχεία. Το πλέγμα υπολογιστών διαφέρει από τα συμβατικά υπολογιστικά συστήματα υψηλής απόδοσης όπως η συστάδα - cluster υπολογιστών στο γεγονός ότι το πλέγμα υπολογιστών έχει τον κάθε κόμβο να εκτελεί μία διαφορετική εφαρμογή. Επίσης, το πλέγμα υπολογιστών είναι πιο ετερογενές και γεωγραφικά διάσπαρτο από τη συστάδα υπολογιστών. Αν και ένα μόνο πλέγμα μπορεί να είναι αφιερωμένο σε μία συγκεκριμένη εφαρμογή, συνήθως το πλέγμα χρησιμοποιείται για διαφόρους σκοπούς. Τα πλέγματα συχνά κατασκευάζονται με γενικού σκοπού βιβλιοθήκες λογισμικού. Το μέγεθός τους μπορεί να είναι αρκετά μεγάλο.
Τα πλέγματα είναι ένα είδος κατανεμημένου υπολογισμού όπου ένας εικονικός υπέρ-υπολογιστής αποτελείται από πολλούς χαλαρούς συνδεδεμένους υπολογιστές στο ίδιο δίκτυο, οι οποίοι ενεργούν μαζί για να εκτελέσουν μεγάλα έργα. Για πολλές εφαρμογές τα κατανεμημένα συστήματα ή το πλέγμα υπολογιστών μπορεί να θεωρηθεί ως ένας ειδικός τύπος παράλληλου υπολογισμού όπου βασίζεται σε ολοκληρωμένους υπολογιστές συνδεδεμένους σε ένα δίκτυο υπολογιστών μέσω μίας δικτυακής διεπαφής όπως είναι το Ethernet. Αυτή είναι η αντίθεση στην παραδοσιακή ιδέα ενός υπέρ-υπολογιστή, στον οποίον πολλοί επεξεργαστές είναι συνδεδεμένοι με ένα τοπικό δίαυλο υψηλής ταχύτητας.
Ο όρος πλέγμα υπολογιστών προήλθε στην αρχή του 1990 ως μεταφορά για το πόσο εύκολο ήταν η πρόσβαση της δύναμης των υπολογιστών σε σχέση με ένα πλέγμα ηλεκτρικής ενέργειας. Η συγκεκριμένη μεταφορά έγινε κανονική όταν ο Ian Foster και ο Carl Kesselman δημοσίευσαν τη πρώτη τους δουλειά "The Grid: Blueprint for a new computing infrastructure"(1999).

Για να πραγματοποιηθεί σύνδεση κάποιου χρήση με τον διακομιστή, πρέπει να στείλει και να ταυτοποιήσει τα διαπιστευτήριά του «όνομα χρήστη, κωδικό χρήστη» μέσω της γραμμής εντολών με την χρήση του πρωτοκόλλου SSH.

Ο εντοπισμός και η κατανόηση των δεδομένων του συστήματος από τους χρήστες, πληροφορίες δηλαδή για τις εργασίες του κάθε χρήστη, τις ουρές, τους κόμβους και πληροφορίες του διακομιστή δεν είναι εύχρηστα διότι αναπαρίστανται με μορφή κειμένου.

Η διαχείριση εργασιών και αρχείων του κάθε χρήστη απαιτεί την καλή γνώση εντολών του κελύφους.

Η εξοικείωση των χρηστών με συστήματα UNIX, τις εντολές του TORQUE, αλλά και με την χρήση της κονσόλας – τερματικού για την διαχείριση εργασιών, αρχείων και για την αναπαράσταση των πληροφοριών του συστήματος, είναι χρονοβόρα διαδικασία. Το περιβάλλον γραμμής εντολών απαιτεί γνώση μίας μεγάλης γκάμας εντολών. Η αποτελεσματικότητα και η παραγωγικότητα εξαρτώνται καθαρά από το πόσες τέτοιες εντολές γνωρίζει ο χρήστης. Για να φτάσει ο χρήστης σε ένα υψηλό επίπεδο είναι αρκετά δύσκολο και χρονοβόρο, μεταβλητές που έχουν κυρίαρχο ρόλο στην εκπόνηση ενός project που κάνει χρήση του εργαλείου TORQUE.

Η διπλωματική αυτή εργασία έχει αντικείμενό της την δημιουργία μιας υπηρεσίας και εφαρμογής ιστού, η οποία με την χρήση του πρωτόκολλου REST θα επιτρέπει σε χρήστες να

συνδέονται σε απομακρυσμένο διακομιστή, μέσω κάποιου φυλλομετρητή ιστού και να χρησιμοποιούν το TORQUE PBS μέσα από ένα γραφικό περιβάλλον φιλικό προς το χρήστη. Ο χρήστης θα μπορεί να πραγματοποιεί απομακρυσμένη σύνδεση με τον διακομιστή στον οποίο είναι εγκατεστημένο το TORQUE, θα μπορεί να βλέπει τους διαθέσιμους πόρους του συστήματος, να ανεβάζει/διαγράφει αρχεία προς εκτέλεση και να διαχειρίζεται τις εργασίες του, σε ένα γραφικό περιβάλλον, αλλά και με την χρήση διαγραμμάτων. Με αυτόν τον τρόπο ο χρήστης θα μπορεί να παίρνει πληροφορίες του συστήματος και να διαχειρίζεται τις εργασίες του, οργανωμένα, με πιο παραστατικό τρόπο χωρίς να χρειάζεται να χρησιμοποιεί εντολές κελύφους.

ΚΕΦΑΛΑΙΟ 2 – Σχετική δουλειά

2.1 Portable batch system (PBS)

Λογισμικό που εκτελεί τον προγραμματισμό εργασιών. Ο βασικός του στόχος είναι να κάνει κατανομή υπολογιστικών έργων - task, για παράδειγμα εργασιών δέσμης, ανάμεσα στους υπολογιστικούς πόρους που είναι διαθέσιμοι. Συχνά χρησιμοποιείται σε συνδυασμό με περιβάλλοντα συστάδων UNIX – UNIX cluster environments. Το PBS υποστηρίζεται σαν μηχανισμός προγραμματισμού εργασιών από αρκετούς meta-schedulers συμπεριλαμβανομένων του Moab από την Cluster resources η οποία έγινε Adaptive computing και του Grid resource allocation manager (GRAM) που αποτελεί μέρος του Globus Toolkit.

Το PBS αρχικά δημιουργήθηκε για την NASA στο πλαίσιο ενός σχεδίου σύμβασης το οποίο ξεκίνησε 17 Ιουνίου 1971. Ο κύριος κατασκευαστής, ο οποίος υλοποίησε τον αρχικό κώδικα ήταν η MRJ Technology Solutions. Η MRJ εξαγοράστηκε από την Veridian το 1990. Η Altair engineering απέκτησε τα δικαιώματα της τεχνολογίας PBS και την πνευματική ιδιοκτησία από την Veridian το 2003. Η Altair engineering κατέχει σήμερα και διατηρεί τα πνευματικά δικαιώματα που σχετίζονται με το PBS και έχει προσλάβει την αρχική ομάδα ανάπτυξης από την NASA.

2.2 Terascale Open-source Resource and QUEue Manager (TORQUE)

Είναι ένας κατανεμημένος διαχειριστής πόρων ο οποίος παρέχει έλεγχο σε εργασίες δέσμης και κατανεμημένους υπολογιστικούς κόμβους. Ενώ έχει ενσωματωμένο προγραμματιστή εργασιών - scheduler, τυπικά χρησιμοποιείται μόνον σαν διαχειριστής πόρων σε συνεργασία με κάποιον scheduler ο οποίος του στέλνει αιτήσεις - requests.

Οι διαχειριστές πόρων παρέχουν την χαμηλού επιπέδου λειτουργικότητα που χρειάζεται για να κάνουμε *start, hold, cancel, and monitor* εργασίες δέσμης - jobs. Χωρίς αυτές τις λειτουργίες, ένας scheduler δεν μπορεί να έχει τον έλεγχο σε εργασίες δέσμης.

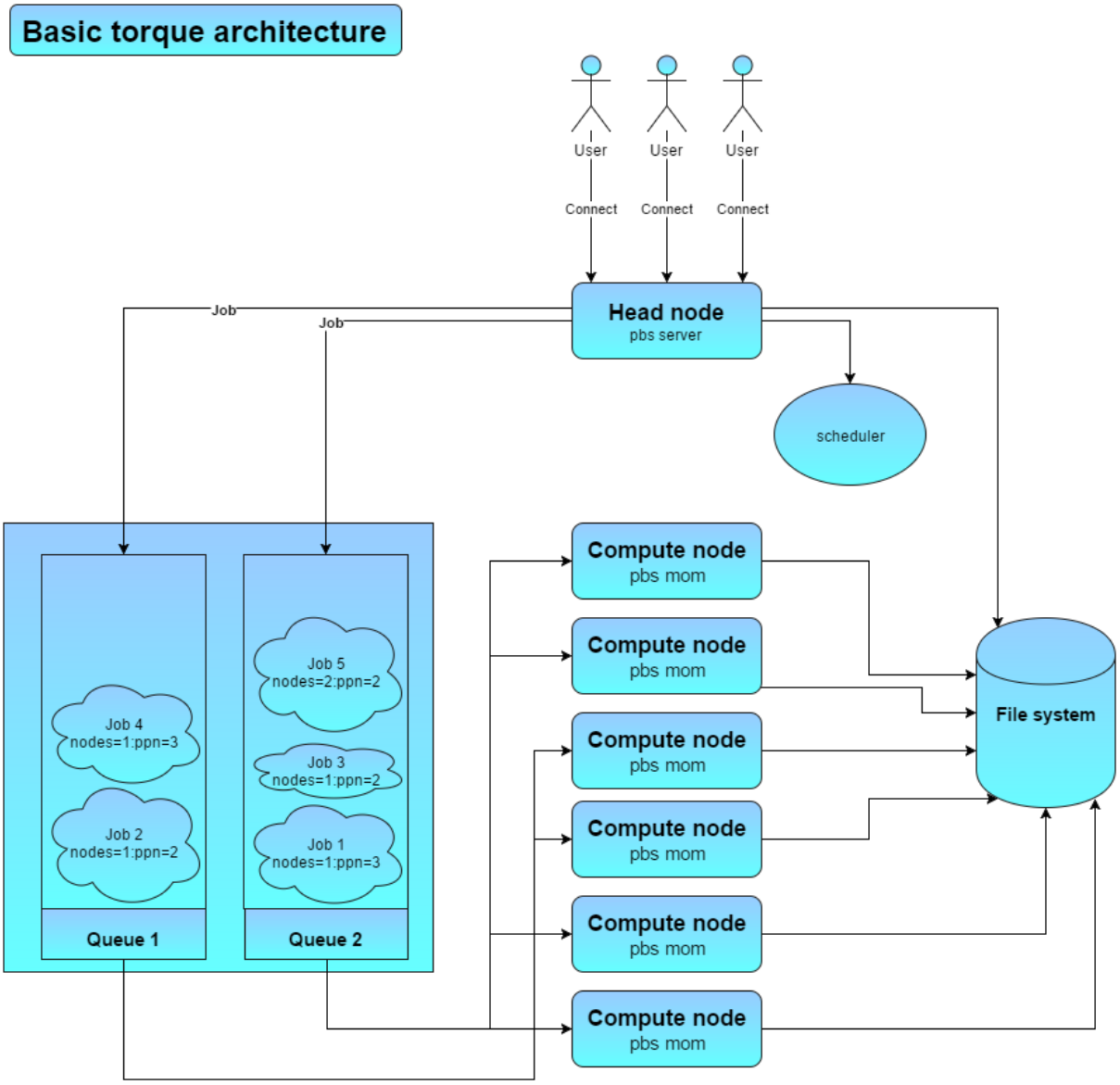
Το TORQUE μπορεί να ενσωματωθεί με το μη εμπορικό Maui Cluster Scheduler ή με το εμπορικό Moab Workload Manager, ώστε να βελτιώσει την συνολική χρήση, τον προγραμματισμό και την διαχείριση σε ένα πλέγμα. Η κοινότητα του TORQUE έχει επεκτείνει το PBS, ώστε να παρατείνει την επεκτασιμότητα, την ανοχή λαθών και λειτουργικότητα. Το TORQUE περιγράφεται από τους δημιουργούς του σαν λογισμικό ανοιχτού κώδικα, χρησιμοποιώντας την άδεια του OpenPBS έκδοση 2.3, και σαν μη ελεύθερο λογισμικό υπό το Debian Free Software Guidelines λόγω προβλημάτων της άδειας.

Το TORQUE παρέχει βελτιώσεις σε σχέση με το πρότυπο OpenPBS όσων αφορά την ανοχή σε σφάλματα, την διασύνδεση προγραμματισμού εργασιών, την επεκτασιμότητα και την χρηστικότητα. Πιο αναλυτικά, ελέγχονται και διαχειρίζονται πρόσθετες συνθήκες αποτυχίας, υπάρχει υποστήριξη ελέγχου της υγείας των κόμβων.

Επίσης έχει γίνει επέκταση, της διεπαφής επερωτήσεων, παρέχοντας τον προγραμματιστή εργασιών με πρόσθετες και πιο ακριβής πληροφορίες, επέκταση της διεπαφής ελέγχου, επιτρέποντας στον προγραμματιστή εργασιών αυξημένο έλεγχο στην συμπεριφορά και τα γνωρίσματα των εργασιών.

Γίνεται συλλογή στατιστικών των ολοκληρωμένων εργασιών, έχει αυξηθεί η ικανότητα διαχείρισης μεγαλύτερων συστάδων υπολογιστών, μεγαλύτερων εργασιών και μεγαλύτερων μηνυμάτων από τον εξυπηρετητή, τα οποία έχουν μια πιο κατανοητή για τον άνθρωπο μορφή.

Εικόνα 1 - Βασική δομή αρχιτεκτονικής του TORQUE



2.2.1 Δομή του TORQUE

- PBS server
  Ο Torque διακομιστής - PBSπεριέχει όλες τις πληροφορίες της συστάδας υπολογιστών. Έχει πληροφορίες για όλους τους MOM κόμβους. Όλες οι εργασίες δέσμης υποβάλλονται με την εντολή qsub στο διακομιστή, ο οποίος διατηρεί μια βάση δεδομένων που περιέχει πληροφορίες για όλες τις εργασίες και τις καταστάσεις τους.
  Ο προγραμματιστές εργασιών, όπως ο Moab Workload Manager, λαμβάνουν πληροφορίες για τις εργασίες, τους κόμβους και τις ουρές, από τον pbs server και υποβάλλουν τις εργασίες σε κατάσταση running.

- PBS mom

Αυτή η εντολή ξεκινάει την λειτουργία του batch Machine Oriented Mini-server, MOM, στον localhost. Μια λειτουργία της είναι να τοποθετήσει τις εργασίες προς εκτέλεση με τον τρόπο που έχει προκαθορίσει ο διακομιστής του TORQUE, να καθιερώσει όρια χρήσης των πόρων του συστήματος, να παρακολουθεί την εργασία και να ειδοποιήσει τον διακομιστή όταν ολοκληρωθεί.
Επίσης ανταποκρίνεται σε αιτήματα που κάνει ο διακομιστής όσον αφορά την παρακολούθηση κάποιας εργασίας.

- Trqauthd
  Είναι μια διεργασία δαίμονας, η οποία χρησιμοποιείται από το TORQUE για να εξουσιοδοτήσει συνδέσεις χρηστών στον PBS διακομιστή. Από την στιγμή που ξεκινήσει παραμένει μόνιμα ενεργή. Οι χρήστες του TORQUE τότε επικοινωνούν μέσω του trqauthd στην πόρτα 15005. Είναι πολυ-νηματική - multi-threaded διεργασία και μπορεί να υποστηρίξει ταυτόχρονα, τεράστιο αριθμό αιτημάτων.

- PBS sched
  Ο προγραμματιστής εργασιών, διαχειρίζεται τις ουρές των εργασιών σε μια συστάδα υπολογιστών. Η βασική του λειτουργία είναι να θέσει ως προς εκτέλεση εργασίες από τις ουρές μόλις ελευθερωθούν πόροι συστήματος.
  Το ποια εργασία θα εκτελεστεί εξαρτάται από τις εξής παραμέτρους:
  - Προτεραιότητα της κάθε εργασίας
  - Υπολογισμός διαθεσιμότητας πόρων συστήματος
  - Χρόνος εκτέλεσης που διατίθεται στον χρήστη
  - Εκτιμώμενος χρόνος εκτέλεσης
  - Χρόνος εκτέλεσης που έχει παρέλθει
  - Διαθεσιμότητα περιφερειακών συσκευών
  - Εξάρτηση των εργασιών και των αρχείων

2.2.2 Χρήση του TORQUE

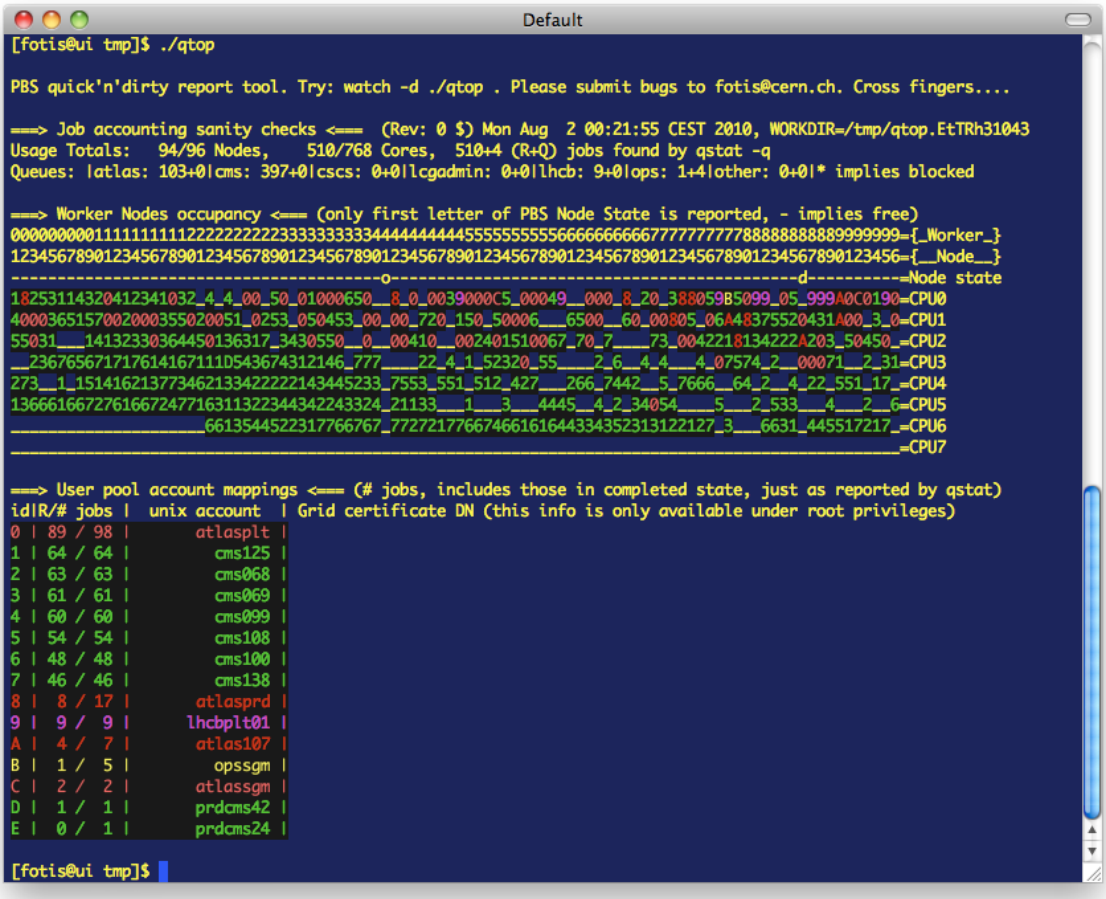
Ο χρήστης συνδέεται με το απομακρυσμένο σύστημα από ένα τερματικό, μέσω πρωτοκόλλου ssh παρέχοντας τα διαπιστευτήριά του.
Μόλις πραγματοποιηθεί η είσοδος του στο σύστημα, αποκτά πρόσβαση στο προσωπικό του κατάλογο αρχείων, στον οποίο μπορεί να διαχειριστεί τα αρχεία του, να χρησιμοποιήσει το TORQUE και να εκτελέσει τις εντολές που υποστηρίζει.

Πίνακας περιγραφής βασικών εντολών του TORQUE

TORQUE εντολές	Περιγραφή
qstat	Παρουσιάζει την κατάσταση των διαθέσιμων ουρών του συστήματος και των εργασιών που σχετίζονται με αυτές. Ανάλογα με τα ορίσματα που δίνει ο χρήστης, επιστρέφει αναλυτικές πληροφορίες για τον διακομιστή, τις εργασίες του συστήματος, τις ουρές κλπ.
qsub	Με αυτήν την εντολή ο χρήστης υποβάλλει μια εργασία στο TORQUE. Εδώ μπορούμε να δηλώσουμε επίσης τους πόρους του συστήματος που χρειάζεται η συγκεκριμένη εργασία, αντί να τους συμπεριλάβουμε στο script, παραδείγματος χάρη τον αριθμό των κόμβων, τον αριθμό επεξεργαστών ανά κόμβο, όνομα της ουράς συστήματος που θα σταλεί η εργασία κλπ.
qhold	Στέλνει αίτημα στον διακομιστή να αλλάξει την κατάσταση μιας η περισσοτέρων εργασιών σε Held. Όταν η κατάσταση της εργασίας είναι Held δεν είναι διαθέσιμη για εκτέλεση. Υπάρχουν 3 διαθέσιμα είδη Hold: χρήστη, χειριστή και διαχειριστή
qrls	Στέλνει ένα αίτημα στον διακομιστή, να αλλάξει την κατάσταση μιας ή περισσοτέρων εργασιών από Held, ώστε να είναι διαθέσιμες για εκτέλεση. Η άρση της κατάστασης Hold της οποιασδήποτε εργασίας, γίνεται με βάση το είδος του Hold.
qdel	Στέλνει ένα αίτημα στον διακομιστή, να διαγράψει κάποια εργασία δέσμης. Όταν μια εργασία διαγραφεί, δεν μπορεί πλέον να διαχειριστεί από υπηρεσίες δέσμης.

TORQUE εντολές	Περιγραφή
qtop	Αυτή η εντολή μας δίνει μια γρήγορη αναπαράσταση της κατάστασης και της χρήσης του συστήματος. Πιο αναλυτικά παρουσιάζει: Γενικές πληροφορίες συστήματος, όπως τους αριθμούς διαθέσιμων και συνολικών κόμβων και εργασιών, αριθμό εργασιών ανά κατάσταση running και queued καθώς και τον αριθμό των ουρών με πληροφορίες για τις εργασίες που βρίσκονται στην κάθε μία. Πίνακας κατανομής εργασιών ανά κόμβο, σε διάταξη επεξεργαστών ανά κόμβο. Εδώ φαίνεται και ο τρόπος με τον οποίο λειτουργεί ο scheduler και κάνει την κατανομή των εργασιών. Υπάρχει αναγνωριστικό για την κάθε κατάσταση του κάθε κόμβου down (d), busy (b), offline (o), job id (id). Πληροφορίες λογαριασμών χρηστών, όπως το αναγνωριστικό και το όνομα λογαριασμού του κάθε χρήστη καθώς και τον αριθμό εργασιών χρήστη που βρίσκονται σε κατάσταση running προς τον συνολικό αριθμό εργασιών χρήστη συμπεριλαμβανομένων και των εργασιών που βρίσκονται σε κατάσταση Completed. Αξίζει να σημειωθεί ότι είναι η μόνη εντολή η οποία αποδίδει κάποια γραφική αναπαράσταση αναλυτικών δεδομένων του συστήματος με την μορφή κειμένου. Αντιπροσωπευτικό παράδειγμα φαίνεται στην Εικόνα 2 – qtop παράδειγμα.

Εικόνα 2 - qtop παράδειγμα



2.3 Python

Είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού η οποία δημιουργήθηκε από τον Ολλανδό Γκβίντο βαν Ρόσσουμ (Guido van Rossum) το 1990. Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της και το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα απ'ότι θα ήταν δυνατόν σε γλώσσες όπως η C++ ή η Java. Διακρίνεται λόγω του ότι έχει πολλές βιβλιοθήκες που διευκολύνουν ιδιαίτερα αρκετές συνηθισμένες εργασίες και για την ταχύτητα εκμάθησης της.

Οι διερμηνευτές της Python είναι διαθέσιμοι για εγκατάσταση σε πολλά λειτουργικά συστήματα, επιτρέποντας στην Python την εκτέλεση κώδικα σε ευρεία γκάμα συστημάτων.

Χρησιμοποιώντας εργαλεία τρίτων, όπως το Py2exe ή το Pyinstaller, ο κώδικας της Python μπορεί να οργανωθεί σε αυτόνομα εκτελέσιμα προγράμματα για μερικά από τα πιο δημοφιλή λειτουργικά συστήματα, επιτρέποντας τη διανομή του βασισμένου σε Python λογισμικού για χρήση σε αυτά τα περιβάλλοντα χωρίς να απαιτείται εγκατάσταση του διερμηνευτή της Python.

Η Python αναπτύσσεται ως ανοιχτό λογισμικό - open source και η διαχείρισή της γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation. Ο κώδικας διανέμεται με την άδεια Python Software Foundation License η οποία είναι συμβατή με την GPL. Το όνομα της γλώσσας προέρχεται από την ομάδα άγγλων κωμικών Μόντυ Πάιθον.

2.4 SQLite

Είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που περιέχεται σε μια C προγραμματιστική βιβλιοθήκη. Σε αντίθεση με άλλα συστήματα διαχείρισης βάσης δεδομένων, το SQLite δεν είναι μια ξεχωριστή διεργασία που προσπελάζεται από μια εφαρμογή πελάτη, αλλά ένα ενσωματωμένο μέρος της. Είναι μια δημοφιλής επιλογή ως ενσωματωμένη βάση δεδομένων για τοπική αποθήκευση ή αποθήκευση πελάτη σε λογισμικό εφαρμογής. Είναι η πιο αναπτυσσόμενη μηχανή βάσης δεδομένων και χρησιμοποιείται σήμερα από πολλούς φυλλομετρητές ευρείας χρήσης , από λειτουργικά συστήματα και από ενσωματωμένα συστήματα.

Το SQLite έχει συνδέσεις με πολλές γλώσσες προγραμματισμού. Αντίθετα με τα συστήματα διαχείρισης βάσεων δεδομένων μορφής Μοντέλο πελάτη-διακομιστή, η μηχανή SQLite δεν έχει αυτόνομες διεργασίες με τις οποίες επικοινωνεί το πρόγραμμα της εφαρμογής, αντίθετα είναι ένα ενσωματωμένο μέρος του προγράμματος εφαρμογής. Η βιβλιοθήκη μπορεί επίσης να κληθεί δυναμικά. Το πρόγραμμα της εφαρμογής χρησιμοποιεί τη λειτουργικότητα του SQLite μέσα από απλές κλήσεις συνάρτησης, που μειώνουν την καθυστέρηση στην πρόσβαση της βάσης δεδομένων καθώς και οι κλήσεις της συνάρτησης μέσα από μια απλή διεργασία είναι πιο αποτελεσματικές από την δια-διεργασιακή επικοινωνία. Το SQLite αποθηκεύει την συνολική βάση δεδομένων ως ένα μοναδικό δια-λειτουργικό αρχείο στη μηχανή ενός οικοδεσπότη. Υλοποιεί αυτόν τον απλό σχεδιασμό με κλείδωμα όλου του αρχείου της βάσης δεδομένων κατά τη διάρκεια της εγγραφής. Τέλος, διαβάζει λειτουργίες που μπορεί να είναι πολύ-λειτουργικές, αν και οι εγγραφές μπορούν να γίνουν μόνο με τη σειρά.

2.5 HyperText Transfer Protocol (HTTP)

Είναι ένα πρωτόκολλο επικοινωνίας. Αποτελεί το κύριο πρωτόκολλο που χρησιμοποιείται στους φυλλομετρητές του Παγκοσμίου Ιστού για να μεταφέρει δεδομένα ανάμεσα σε έναν διακομιστή (server) και έναν πελάτη (client).

Ο όρος υπερκείμενο (hypertext), που περιέχεται στην ονομασία του πρωτοκόλλου, χρησιμοποιήθηκε αρχικά από τον Τεντ Νέλσον το 1965. Η γενική ιδέα του πρωτοκόλλου προτάθηκε, μαζί με τη δημιουργία της γλώσσας HTML, από τον Τιμ Μπέρνερς Λι και την ομάδα του, ώστε, σε συνδυασμό με το ήδη υπάρχον Διαδίκτυο και το πρωτόκολλο TCP, να γίνει εφικτή η δημιουργία του Παγκόσμιου Ιστού (WWW).

Σήμερα το πρωτόκολλο αυτό είναι πλέον καθιερωμένο και διαδεδομένο σε σημείο που σχεδόν όλοι οι φυλλομετρητές να το θεωρούν δεδομένο και να το χρησιμοποιούν σε περίπτωση που ο χρήστης δεν καθορίσει ποιο πρωτόκολλο θέλει να χρησιμοποιήσει. Αν δηλαδή ο χρήστης δεν γράψει: http://my.url αλλά γράψει σκέτο το: my.url σχεδόν όλοι οι φυλλομετρητές θεωρούν σαν δεδομένο το πρωτόκολλο http και όχι κάποιο άλλο (https, ftp κλπ.)

2.6 Representational State Transfer (REST)

Είναι το αρχιτεκτονικό στυλ λογισμικού του Παγκόσμιου Ιστού(World Wide Web). Το REST δίνει ένα συντονισμένο σύνολο από περιορισμούς στο σχεδιασμό των τμημάτων σε ένα κατανεμημένο *hypermedia* σύστημα το οποίο μπορεί να οδηγήσει σε μία αρχιτεκτονική με πολύ υψηλή απόδοση.

Τα συστήματα τα οποία συμμορφώνονται στους περιορισμούς του REST ονομάζονται RESTful. Συνήθως τα συστήματα αυτά επικοινωνούν μέσω HTTP κάνοντας χρήση των ίδιων HTTP verbs(GET, POST, PUT, DELETE) με αυτά των προγραμμάτων περιήγησης ιστού για να ανακτήσουν μία ιστοσελίδα ή να στείλουν δεδομένα σε κάποιο διακομιστή. Το REST αλληλεπιδρά με εξωτερικά συστήματα χρησιμοποιώντας πόρους οι οποίοι προσδιορίζονται μέσω URI, οι οποίοι πόροι μπορούν να επεξεργασθούν μέσω των HTTP verbs, π.χ. DELETE /people/tom

2.7 Application Programming Interface (API)

Γνωστή και ως Διασύνδεση Προγραμματισμού Εφαρμογών (για συντομία διεπαφή ή διασύνδεση), είναι η διεπαφή των προγραμματιστικών διαδικασιών που παρέχει ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή προκειμένου να επιτρέπει να γίνονται προς αυτά αιτήσεις από άλλα προγράμματα ή/και ανταλλαγή δεδομένων. Ένας από τους βασικούς σκοπούς μίας διεπαφής είναι να ορίζει και να διατυπώνει το σύνολο των λειτουργιών-υπηρεσιών που μπορεί να παρέχει μια βιβλιοθήκη ή ένα λειτουργικό σύστημα σε άλλα προγράμματα, χωρίς να επιτρέπει πρόσβαση στον κώδικα που υλοποιεί αυτές τις υπηρεσίες. Η διεπαφή, ένα «συμβόλαιο κλήσης» μεταξύ καλούντος και καλούμενου, διαχωρίζει την προγραμματιστική υλοποίηση κάποιων υπηρεσιών από τη χρήση τους.

2.8 Web Server Gateway Interface (wsgi)

Είναι ένας προσδιορισμός για απλές και γενικές διεπαφές ανάμεσα σε διαδικτυακούς διακομιστές και διαδικτυακές εφαρμογές για την προγραμματιστική γλώσσα Python. Αρχικά προσδιορίστηκε στο PEP 333 με συγγραφέα τον Phillip J. Eby και δημοσιεύτηκε στις 7 Δεκεμβρίου 2003. Έκτοτε, έχει καθιερωθεί ως στάνταρντ πρότυπο για την ανάπτυξη διαδικτυακών εφαρμογών Python. Το wsgi αποτελείται από δύο πλευρές: τη πλευρά του διακομιστή και τη πλευρά της εφαρμογής.

Για την επεξεργασία ενός WSGI αιτήματος, ο διακομιστής εκτελεί την εφαρμογή και παρέχει πληροφορίες περιβάλλοντος και μία συνάρτηση επανάκλησης (callback) στην εφαρμογή. Η εφαρμογή επεξεργάζεται το αίτημα, και επιστρέφει την απάντηση στο διακομιστή μέσω της συνάρτησης επανάκλησης που είχε προμηθευτεί νωρίτερα. Ανάμεσα στο διακομιστή και στην εφαρμογή υπάρχει ένα WSGI ενδιάμεσο λογισμικό(middleware), το οποίο εφαρμόζει και τις δύο πλευρές της διεπαφής προγραμματισμού εφαρμογής(API). Ο διακομιστής λαμβάνει το αίτημα από έναν πελάτη (client) και το προωθεί στο ενδιάμεσο λογισμικό. Μετά την επεξεργασία του στέλνει ένα αίτημα σε μία άλλη εφαρμογή WSGI της οποίας η απάντηση προωθείται στον πελάτη μέσω του ενδιάμεσου λογισμικού και εν τέλει μέσω του διακομιστή. Μπορεί να υπάρχουν πολλά ενδιάμεσα λογισμικά, με αποτέλσεμα να σχηματίζεται μία στοίβα από εφαρμογές συμβατές με WSGI.

2.9 Gunicorn

Είναι ένας Python WSGI HTTP διακομιστής για το λειτουργικό σύστημα Unix. Είναι ένα pre-fork worker μοντέλο που έχει τις ρίζες του στο σχέδιο Unicorn για τη γλώσσα προγραμματισμού Ruby. Ο διακομιστής του Gunicorn είναι συμβατός με πολλούς σκελετούς (frameworks) διαδικτυακών εφαρμογών, με απλή εφαρμογή, χωρίς να καταναλώνει πολλούς πόρους από το διακομιστή και αρκετά γρήγορος.

2.10 Pbs_python

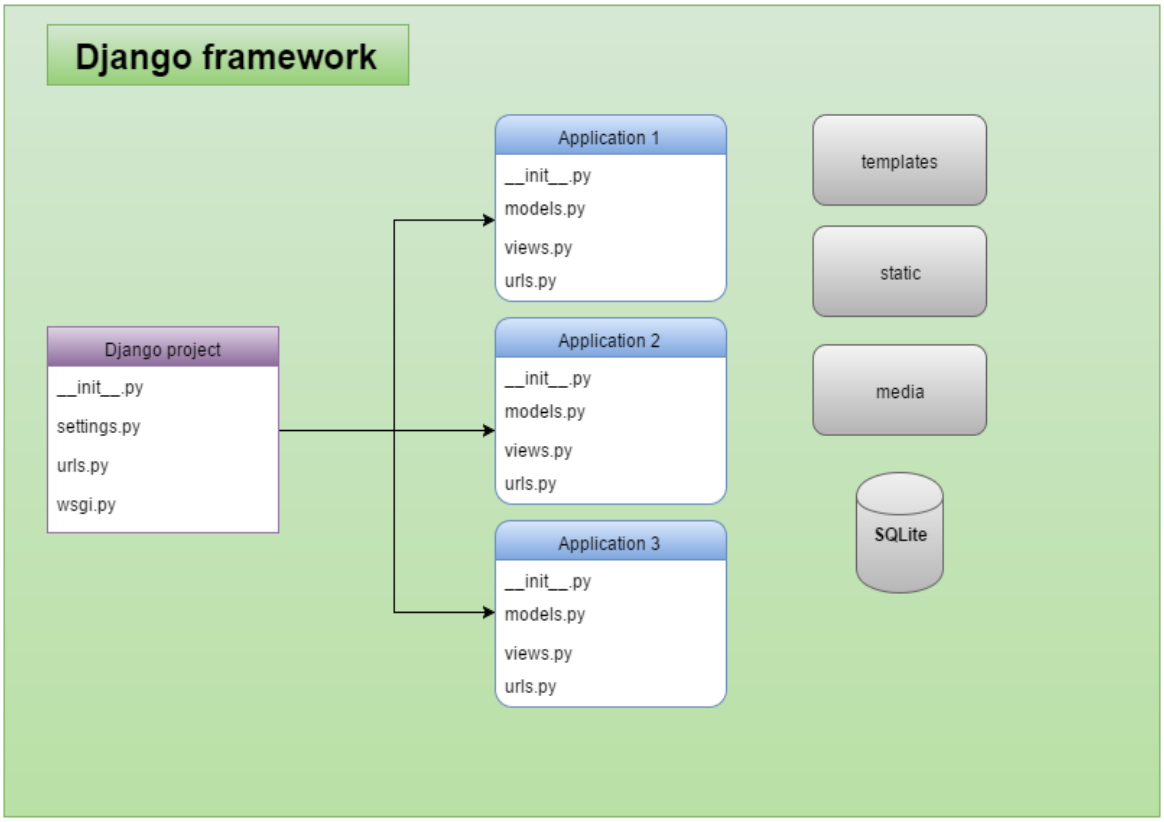
Η pbs_python βιβλιοθήκη είναι μια wrapper class για την C βιβλιοθήκη του Torque. Με την pbs_python μπορούμε να γράψουμε utilities/expansions για το Torque σε Python.

Η PBSQuery συμπεριλαμβάνεται επίσης στην βιβλιοθήκη, η οποία είναι ένα python module βασισμένο στο pbs python module, ώστε να απλοποιεί τα ερωτήματα στον batch server πχ: πόσες jobs τρέχουν στο σύστημα, πόσοι είναι οι ενεργοί nodes του συστήματος.

2.11 Django

Το Django, είναι ένα υψηλού επιπέδου Python Web framework, το οποίο ενθαρρύνει το γρήγορο development και το καθαρό, πραγματιστικό design.

Εικόνα 3 - Δομή του Django project



2.11.1 Model

Αν και μπορούμε να χρησιμοποιήσουμε το Django χωρίς βάση δεδομένων, μας παρέχει έναν object-relational mapper (ORM), ο οποίος μας δίνει την δυνατότητα να φτιάξουμε το layout της βάσης, με κώδικα Python. Μας παρέχει ένα Python API, που υποστηρίζει C.R.U.D. λειτουργίες της βάσης δεδομένων.

2.11.2 Δυναμικό διαχειριστικό περιβάλλον

Από την στιγμή που οριστούν τα μοντέλα, το Django αυτόματα δημιουργεί ένα επαγγελματικό και production ready διαχειριστικό interface - ένα Web site το οποίο επιτρέπει σε πιστοποιημένους χρήστες να προσθέσουν, αλλάξουν και να διαγράψουν αντικείμενα.

Η φιλοσοφία εδώ είναι ότι το site μπορεί να διαχειριστεί από το προσωπικό, κάποιον πελάτη, ή μόνο από τον διαχειριστή – και με αυτόν τον τρόπο δεν χρειάζεται να δημιουργηθούν νέα back end interfaces για απλή διαχείριση περιεχομένων.

Μια τυπική ροή εργασίας στην δημιουργία εφαρμογών με Django, είναι η δημιουργία μοντέλων και το ανέβασμα του admin site το γρηγορότερο δυνατό, ώστε το προσωπικό ή οι πελάτες να αρχίσουν να δημοσιεύουν δεδομένα. Στην συνέχεια γίνεται η ανάπτυξη του τρόπου με τον οποίο τα δεδομένα παρουσιάζονται στο κοινό.

2.11.3 View

Κάθε view είναι υπεύθυνη να κάνει δύο πράγματα:

- Να επιστρέφει ένα HttpResponse object, το οποίο περιέχει τα περιεχόμενα της σελίδας που ζητήθηκε
- Να δημιουργεί μια εξαίρεση Http404

Γενικά οι όψεις ανακτούν δεδομένα σύμφωνα με διάφορες παραμέτρους, κάνουν load ένα template και κάνουν render το template με αυτά τα δεδομένα που έλαβαν.

2.11.4 Url

Ένα καθαρό και κομψό URL σχήμα, αποτελεί σημαντική λεπτομέρεια σε μια υψηλού επιπέδου WEB εφαρμογή. Το Django ενθαρρύνει τον όμορφο σχεδιασμό URL χωρίς κόπο.

2.11.5 Templates

Καθορίζει την εμφάνιση του site, και παρέχει συνδέσεις με child templates. Αυτό κάνει τον επανασχεδιασμό site απλό, σαν να αλλάζουμε ένα μόνον αρχείο. Επίσης, μας δίνει τη δυνατότητα να δημιουργήσουμε πολλαπλές εκδόσεις του ίδιου site κρατώντας μόνο το βασικό template. Παρ'όλ'αυτά δεν είναι απαραίτητο να χρησιμοποιήσουμε το σύστημα των templates, καθώς κάθε μέρος του Django(models, views, templates) δεν είναι αυστηρά συνδεδεμένο με όλα τα υπόλοιπα.

2.12 Django rest framework

Το Django REST framework είναι δυνατό και ευέλικτο εργαλείο, το οποίο χρησιμοποιείται για το εύκολο χτίσιμο Web APIs. Μερικά από τα πλεονεκτήματά του είναι:

- Το Web browsable API.
- Πολιτικές πιστοποίησης συμπεριλαμβανομένων και των πακέτων για OAuth1 και OAuth2.
- Το Serialization που υποστηρίζει ORM (Object relational mapper) και non-ORM πηγές δεδομένων.

- Είναι προσαρμόσιμο σε όλο του το εύρος – μπορούν να χρησιμοποιηθούν και function-based views για λιγότερο ισχυρά χαρακτηριστικά.

Serializers: Οι serializers επιτρέπουν σύνθετα δεδομένα όπως querysets και στιγμιότυπα μοντέλων (Models), να μετατραπούν σε native δεδομένα Python τα οποία εν συνεχεία μπορούν να μετατραπούν σε μορφή JSON ή XML. Επίσης παρέχεται και η αντίστροφη λειτουργικότητα μέσω των deserializers.

Generic Views: Ένα πολύ σημαντικό πλεονέκτημα των class-based views είναι ο τρόπος με τον οποίον μας επιτρέπουν την σύνθεση κομματιών κώδικα με επαναχρησιμοποιήσιμη συμπεριφορά. Το REST framework χρησιμοποιεί αυτό το πλεονέκτημα και παρέχει έναν αριθμό από pre-built views για μοτίβα που χρησιμοποιούνται συχνά. Τα generic views που παρέχονται από το REST framework μας επιτρέπουν να χτίζουμε γρήγορα API views που αντιστοιχούν στα μοντέλα της βάσης δεδομένων μας.

2.13 Extjs

Ext js είναι ένα front-end javascript framework για την δημιουργία feature-rich cross-platform web applications, που στοχεύουν desktop, tablets, και smartphones.

2.13.1 UI – User Intefrace

Sencha Ext JS παρέχει την πιο ολοκληρωμένη συλλογή υψηλών επιδόσεων και προσαρμόσιμων UI widgets. Αυτά τα widgets συμπεριλαμβάνουν HTML5 grids, trees, lists, forms, menus, toolbars, panels, windows και πολλά περισσότερα.

2.13.2 Data

Το εύρωστο πακέτο δεδομένων που συμπεριλαμβάνεται στην Sencha Ext JS αποσυνδέει τα UI widgets από το στρώμα δεδομένων. Το πακέτο δεδομένων επιτρέπει client-side συλλογές δεδομένων χρησιμοποιώντας υψηλού επιπέδου λειτουργικά μοντέλα τα οποία προσφέρουν χαρακτηριστικά όπως ταξινόμηση και φιλτράρισμα. Το πακέτο δεδομένων είναι protocol agnostic και μπορεί να καταναλώσει δεδομένα από οποιαδήποτε backend πηγή. Επίσης συμπεριλαμβάνει ικανότητες session management, οι οποίες επιτρέπουν διάφορες client-side λειτουργίες, ελαχιστοποιώντας τα round-trips στον server.

2.13.3 Layout

Η Sencha Ext JS συμπεριλαμβάνει έναν ευέλικτο layout manager για την οργάνωση της απεικόνισης των δεδομένων και του περιεχομένου σε πολλούς browsers, συσκευές και

μεγέθη οθόνης. Βοηθάει στον έλεγχο της απεικόνισης των components, ακόμα και για τα πιο πολύπλοκα UIs. Ext JS επίσης παρέχει ένα responsive σύστημα διαμόρφωσης, το οποίο επιτρέπει στα components της εφαρμογής να προσαρμόζονται σε συγκεκριμένα orientation της συσκευής (landscape or portrait) ή στο μέγεθος του browser.

2.13.4 Charts – Γραφήματα

Το πακέτο γραφημάτων του Sencha Ext JS επιτρέπει την εικονική αναπαράσταση των δεδομένων μέσα από ένα μεγάλο εύρος γραφημάτων – συμπεριλαμβάνοντας τα line, bar, and pie γραφήματα. Τα γραφήματα χρησιμοποιούν επιφάνειες και sprites τα οποία έχουν δημιουργηθεί χρησιμοποιώντας τεχνολογίες SVG, VML και Canvas. Η ποικιλία των browser, αντιμετωπίζεται αυτόματα, ώστε τα γραφήματα να απεικονίζονται πάντα σωστά. Τα γραφήματα της Ext JS επίσης υποστηρίζουν touch gestures σε συσκευές τηλεφώνων και παρέχουν βελτιωμένα δια δραστικά χαρακτηριστικά, όπως pan, zoom και pinch.

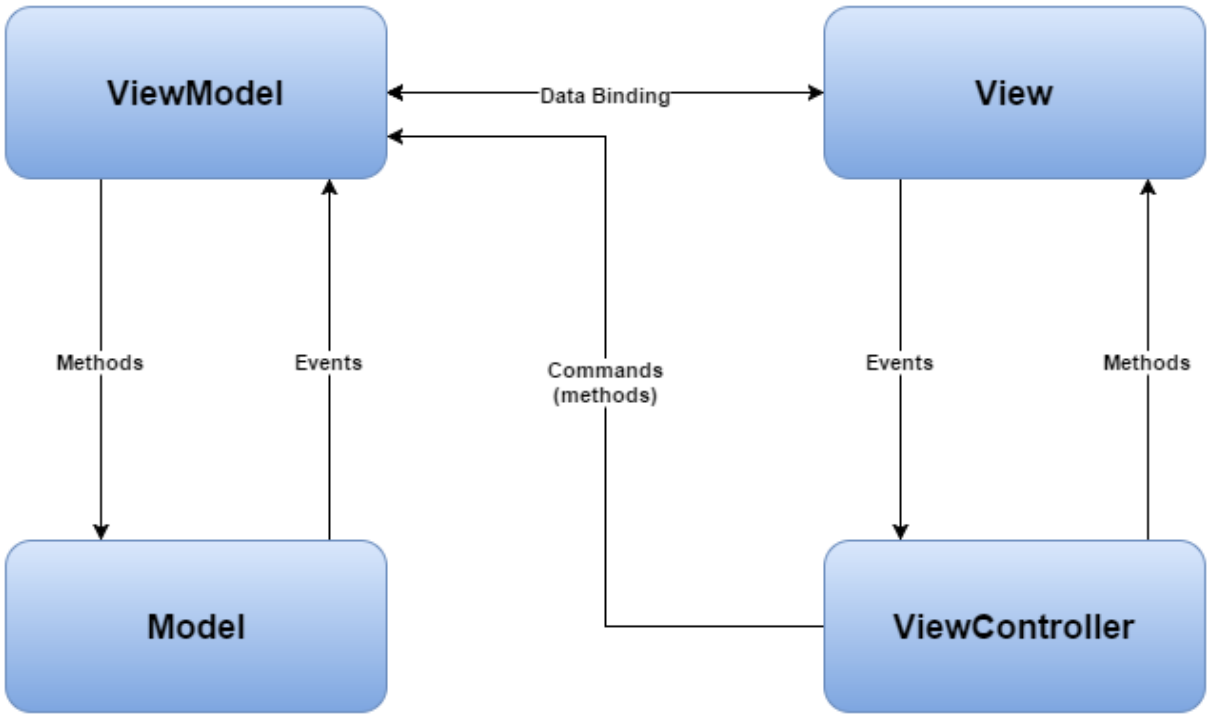
2.13.5 Αρχιτεκτονική

Η Ext JS 5 υποστηρίζει τις αρχιτεκτονικές MVC(Model-View-Controller) και MVVM(Model-View-ViewModel). Και οι δύο αρχιτεκτονικές μοιράζονται την ίδια κεντρική ιδέα και εστιάζονται στον λογικό διαχωρισμό του κώδικα μιας εφαρμογής. Κάθε αρχιτεκτονική προσέγγιση έχει τα δικά της πλεονεκτήματα.

MVC: Στη συγκεκριμένη αρχιτεκτονική όλες οι κλάσεις είναι είτε Models, είτε Views, είτε Controllers. Ο χρήστης αλληλεπιδρά με τα Views, τα οποία απεικονίζουν τα δεδομένα που κρατούνται στα Models. Οι αλληλεπιδράσεις αυτές ελέγχονται από τους Controllers, οι οποίοι απαντούν με στις δράσεις του χρήστη ανανεώνοντας τα Views ή/και τα Models. Επειδή ακριβώς όλη η λογική παρέχεται από τους Controllers, τα Views και τα Models δε γνωρίζει το ένα για την ύπαρξη του άλλου. Ελάχιστη ή καθόλου είναι το μέρος της λογικής που υπάρχει στα Views. Ο στόχος της συγκεκριμένης αρχιτεκτονικής είναι να καθορίσει τις ευθύνες που έχει κάθε κλάση σε μία εφαρμογή, και το γεγονός αυτό κάνει τη κάθε κλάση, στην ουσία, να είναι decoupled από κάθε άλλη κλάση. Αποτέλεσμα είναι μία εφαρμογή η οποία είναι πολύ εύκολη στο testing, στη συντήρηση και με επαναχρησιμοποιήσιμο κώδικα.

MVVM: Η ειδοποιός διαφορά αυτής της αρχιτεκτονικής είναι η εισαγωγή ενός επιπέδου αφαίρεσης στην κλάση View, που ονομάζεται ViewModel. Η συγκεκριμένη κλάση συγχρονίζει τις αλλαγές στα δεδομένα των Models με την αναπαράστασή τους στα Views κάνοντας χρήση μιας τεχνικής που ονομάζεται data-binding. Το αποτέλεσμα είναι τα Models και το framework να εκτελούν την περισσότερη δουλειά και έτσι ελαχιστοποιείται η λογική που εμπεριεχόταν στους Controllers. Η λειτουργικότητα της αρχιτεκτονικής αυτής φαίνεται στο παρακάτω γράφημα.

Εικόνα 4 – Extjs5 αρχιτεκτονική MVVM



2.14 PBS Cluster Viz

Είναι ένα πρόγραμμα με το οποίο απεικονίζονται πληροφορίες, χρήσιμες για τους διαχειριστές και τους χρήστες, για μία συστάδα υπολογιστών από ένα διαχειριστή υπολογιστικών πόρων που είναι συμβατός με ένα φορητό σύστημα δέσμης. Οι πληροφορίες που δίνει αυτό το πρόγραμμα αφορούν το φόρτο και την κατανομή εργασιών. Τέλος, παρέχει δια-δραστική και στατική έξοδο.

2.15 Torque Web Monitor

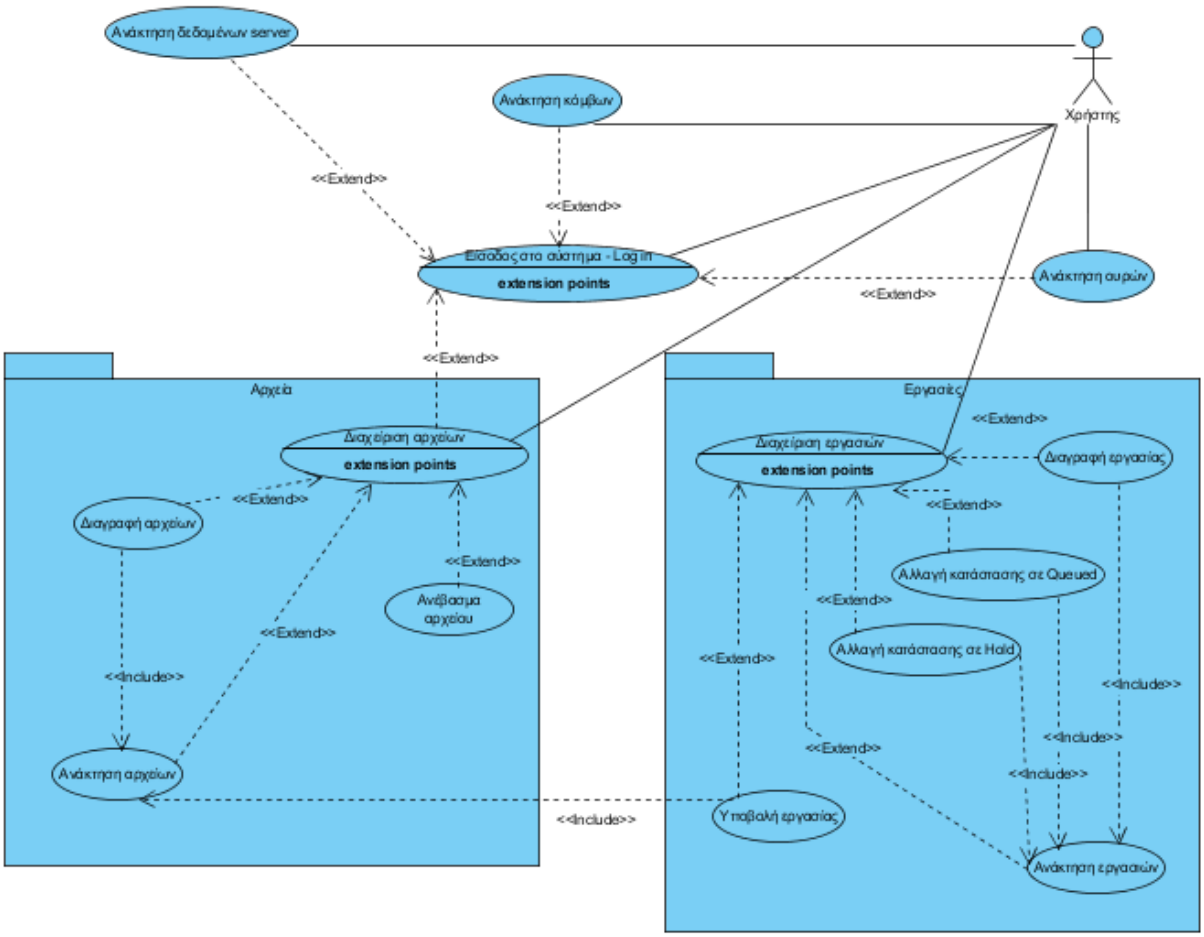
Δια-δραστικός δικτυακός παρατηρητής για PBS/Torque συστήματα δέσμης. Επικεντρώνεται στο να παρέχει στους χρήστες λεπτομερή ανάλυση των εργασιών καθώς και πληροφορίες που σχετίζονται με την ουρά των εργασιών.

2.16 Ganglia Monitoring System

Το Ganglia είναι ένα επεκτάσιμο κατανεμημένο σύστημα εποπτείας για υπολογιστικά συστήματα υψηλής απόδοσης όπως οι συστάδες υπολογιστών και τα πλέγματα. Βασίζεται σε ένα ιεραρχικό σχεδιασμό που στοχεύει σε ομοσπονδίες συστάδων. Κάνει χρήση διαδεδομένων τεχνολογιών όπως XML για την αναπαράσταση δεδομένων, XDR για τη μεταφορά δεδομένων και RRDtool για την αποθήκευση και την απεικόνιση των δεδομένων. Χρησιμοποιεί δομές δεδομένων και αλγόριθμους προσεκτικά σχεδιασμένους για την επίτευξη υψηλού συγχρονισμού. Η εκτέλεση είναι εύρωστη, έχει σχεδιαστεί για ένα μεγάλο σύνολο λειτουργικών συστημάτων και αρχιτεκτονικές υπολογιστών, και χρησιμοποιείται σε χιλιάδες συστάδες υπολογιστών σε όλο τον κόσμο. Έχει χρησιμοποιηθεί για τη σύνδεση

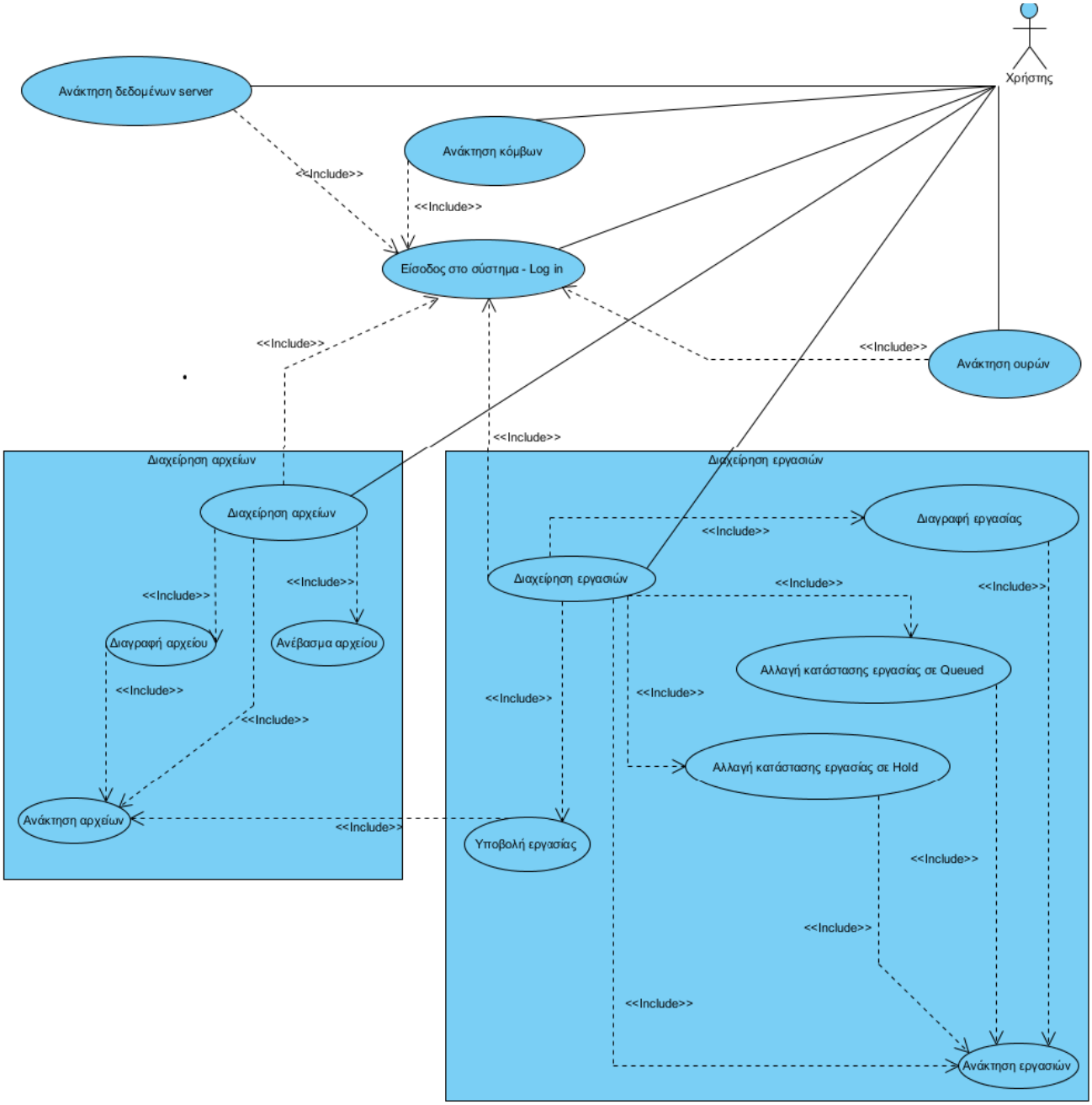
συστάδων υπολογιστών μεταξύ πανεπιστημίων και σε όλο τον κόσμο και μπορεί να ρυθμιστεί με τέτοιο τρόπο ώστε να μπορεί να διαχειριστεί έως και 2000 κόμβους.

ΚΕΦΑΛΑΙΟ 3 – Σχεδιασμός εφαρμογής


Ανάκτηση δεδομένων server
Ανάκτηση κόμβων
Χρήστης
<<Extend>>
<<Extend>>
Είσοδος στο σύστημα - Login
extension points
Ανάκτηση ρυθμών
<<Extend>>
<<Extend>>
Αρχεία
Διαχείριση αρχείων
extension points
Διαγραφή αρχείων
<<Extend>>
<<Extend>>
Ανέβασμα αρχείου
<<Extend>>
<<Include>>
Ανάκτηση αρχείων
<<Include>>
Εργασίες
<<Extend>>
Διαχείριση εργασιών
extension points
Διαγραφή εργασίας
<<Extend>>
Αλλαγή κατάστασης σε Queued
<<Extend>>
Αλλαγή κατάστασης σε Hold
<<Extend>>
<<Include>>
<<Include>>
<<Extend>>
<<Include>>
Υποβολή εργασίας
Ανάκτηση εργασιών

3.1 Διάγραμμα περιπτώσεων χρήσης

Εικόνα 5 - Διάγραμμα περιπτώσεων χρήσης



3.2 Περιγραφή του διαγράμματος περιπτώσεων χρήσης

Ο χρήστης μπορεί να αλληλεπιδράσει με το σύστημα πραγματοποιώντας είσοδο στην εφαρμογή, παρέχοντας τα διαπιστευτήρια «όνομα χρήστη, κωδικό πρόσβασης» του λογαριασμού του. Από την στιγμή που αποκτήσει πρόσβαση στο σύστημα, μπορεί να πραγματοποιήσει ανάκτηση δεδομένων του διακομιστή, πόρων του συστήματος όπως κόμβοι, ουρές και εργασίες, καθώς και να διαχειριστεί τα αχρεία και τις εργασίες του.

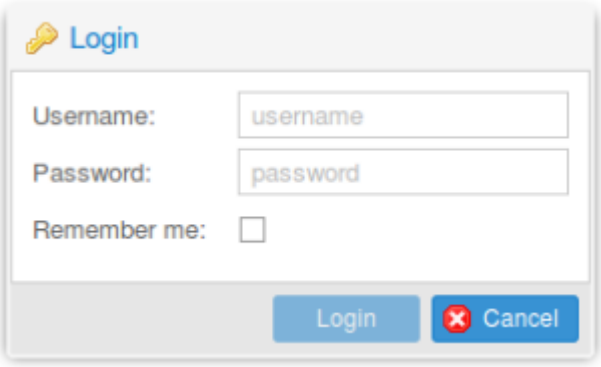
3.3 Περιπτώσεις χρήσεις

3.3.1 Είσοδος στο σύστημα

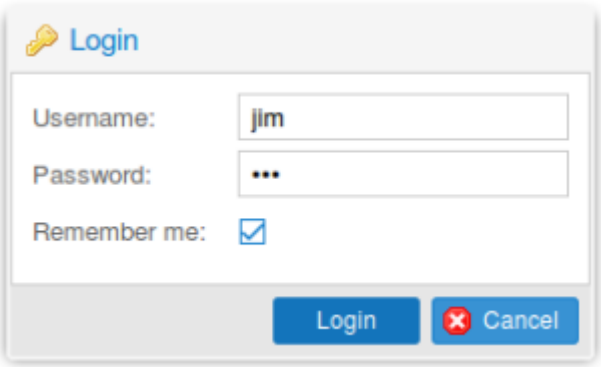
Τίτλος περίπτωσης χρήσης: «Είσοδος στο σύστημα»

Βασική ροή 1:

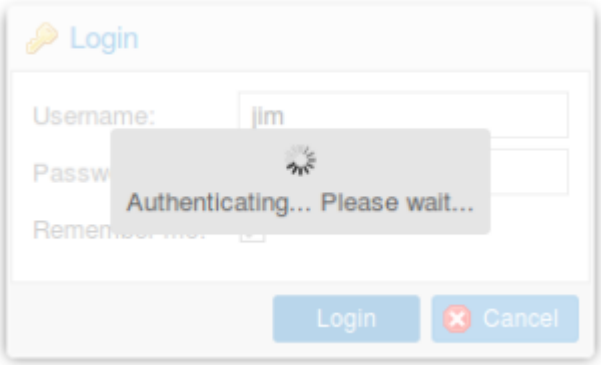
- Ο χρήστης συμπληρώνει τα πεδία username και password



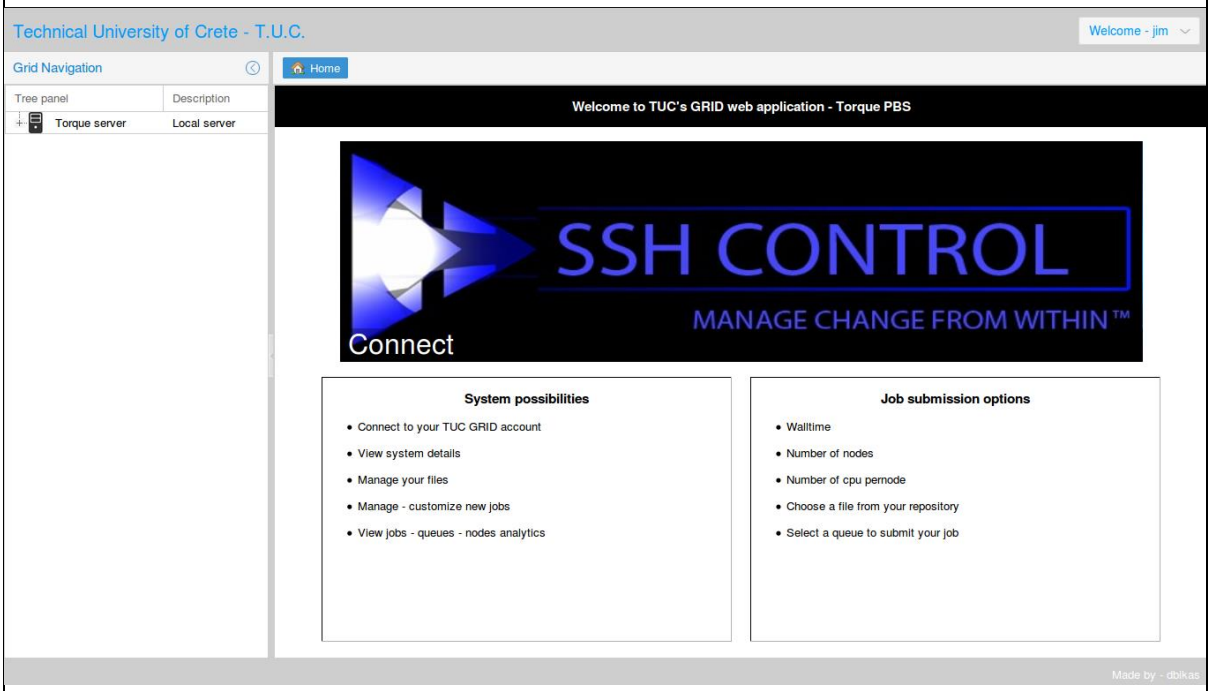
- Ενεργοποιείται το κουμπί Login



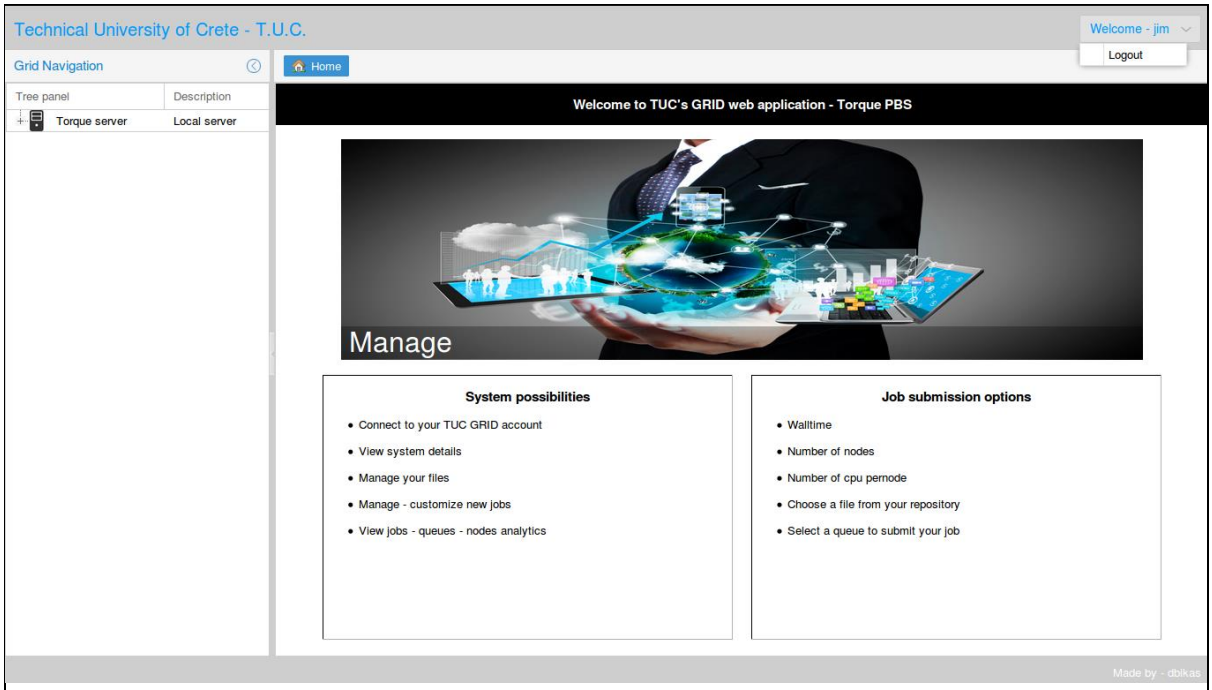
- Το checkbox: remember me είναι επιλεγμένο και ο χρήστης πατάει το κουμπί login



- Τα διαπιστευτήρια του χρήστη υπάρχουν στο απομακρυσμένο σύστημα GRID
- Ο χρήστης έχει λογαριασμό στην βάση δεδομένων της εφαρμογής
- Ο λογαριασμός του χρήστη στην βάση δεδομένων της εφαρμογής είναι ενεργός



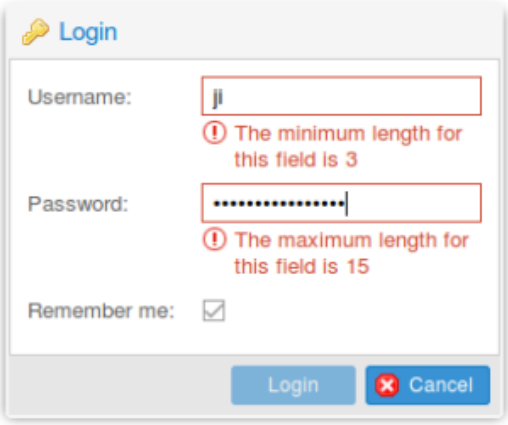
- Ο χρήστης συνδέεται με επιτυχία στο σύστημα βλέπει το dashboard του και γίνεται έξοδος μόνον αν το επιλέξει ο χρήστης.



Εναλλακτική ροή 1:

Το πεδίο username ή password

- είναι άδεια
- έχουν λιγότερους από 3 χαρακτήρες
- έχουν περισσότερους από 25 χαρακτήρες



εμφανίζεται αντίστοιχο μήνυμα λάθους και το κουμπί login παραμένει μη ενεργό.

Εναλλακτική ροή 2:

- Τα διαπιστευτήρια του χρήστη δεν υπάρχουν στο απομακρυσμένο σύστημα GRID

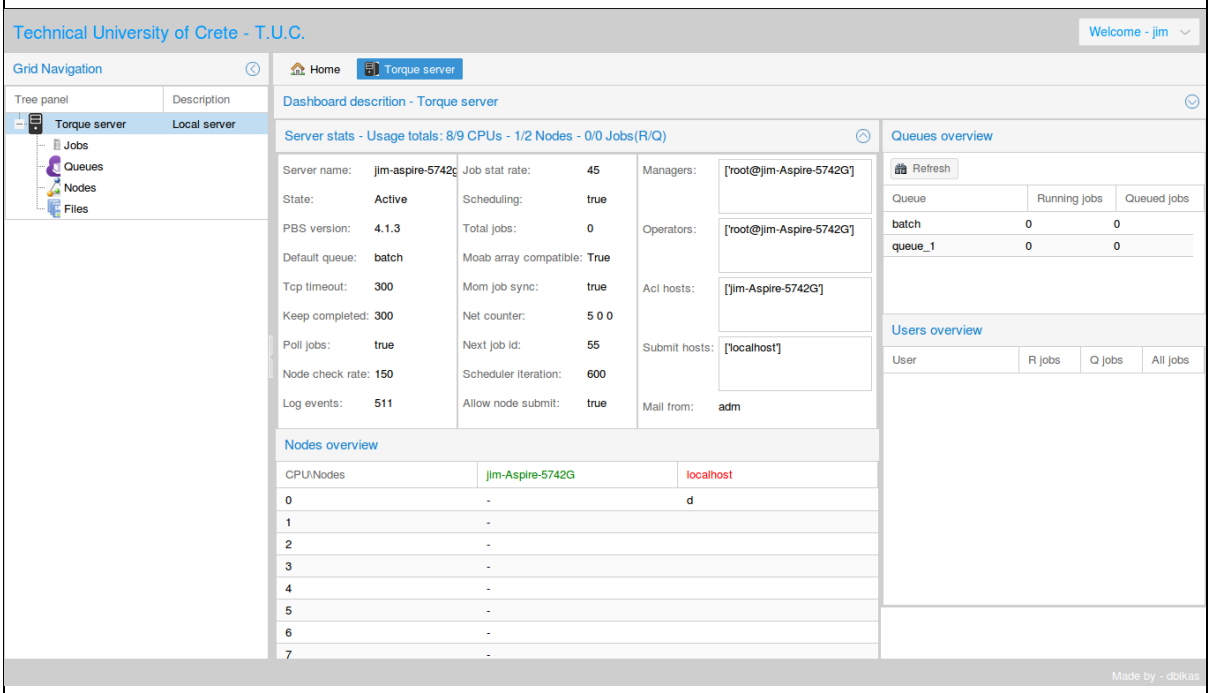
• Εμφανίζεται μήνυμα λάθους το οποίο συμβουλεύει τον χρήστη να επικοινωνήσει με έναν διαχειριστή του συστήματος
Εναλλακτική ροή 3: • Ο χρήστης συνδέεται για πρώτη φορά από την εφαρμογή και δεν διαθέτει λογαριασμό στην βάση δεδομένων της • Δημιουργείται νέος λογαριασμός για τον χρήστη στην βάση δεδομένων • Δημιουργείται νέος κατάλογος αρχείων για τον χρήστη με τα αντίστοιχα δικαιώματα • Ο χρήστης συνδέεται επιτυχώς στην εφαρμογή
Εναλλακτική ροή 4: • Ο λογαριασμός του χρήστη στην βάση δεδομένων της εφαρμογής είναι ανενεργός • Εμφανίζεται αντίστοιχο μήνυμα σφάλματος
Εναλλακτική ροή 5: • Το checkbox: remember me, δεν είναι επιλεγμένο και πραγματοποιείται είσοδος στην εφαρμογή • Αν γίνει ανανέωση του φυλλομετρητή ιστού ή ο φυλλομετρητής ιστού κλείσει, ο χρήστης πρέπει να πραγματοποιήσει είσοδο στο σύστημα πάλι
Εναλλακτική ροή 6: • Ο χρήστης επιλέγει το κουμπί cancel • Καθαρισμός της φόρμας

3.3.2 Ανάκτηση δεδομένων διακομιστή

Τίτλος περίπτωσης χρήσης: «Ανάκτηση δεδομένων διακομιστή»
Βασική ροή 1: • Ο χρήστης πατάει την επιλογή: Torque server από το πλαίσιο δέντρου στα αριστερά

- Ανοίγει ένα νέο πλαίσιο, στο οποίο παρουσιάζονται πληροφορίες σχετικά με τον διακομιστή

Υπάρχουν 4 υπό-πλαίσια τα οποία παρέχουν αναλυτικές πληροφορίες του συστήματος.



- Queues overview – πάνω δεξιά

  Πίνακας που δείχνει τον αριθμό των Running και Queued εργασιών ανά ουρά.

- Users overview – κάτω δεξιά

  Πίνακας που δείχνει τον αριθμό των Running, Queued και All εργασιών ανά χρήστη.

- Server stats – πάνω αριστερά

  Ένα πτυσσόμενο πλαίσιο, το οποίο παρέχει γενικές πληροφορίες για τ TORQUE.

  Στον τίτλο του είναι πάντα ορατός ο αριθμός των:

  Free / all επεξεργαστών, free / all κόμβων, running / queue εργασιών.

- Nodes overview – κάτω αριστερά

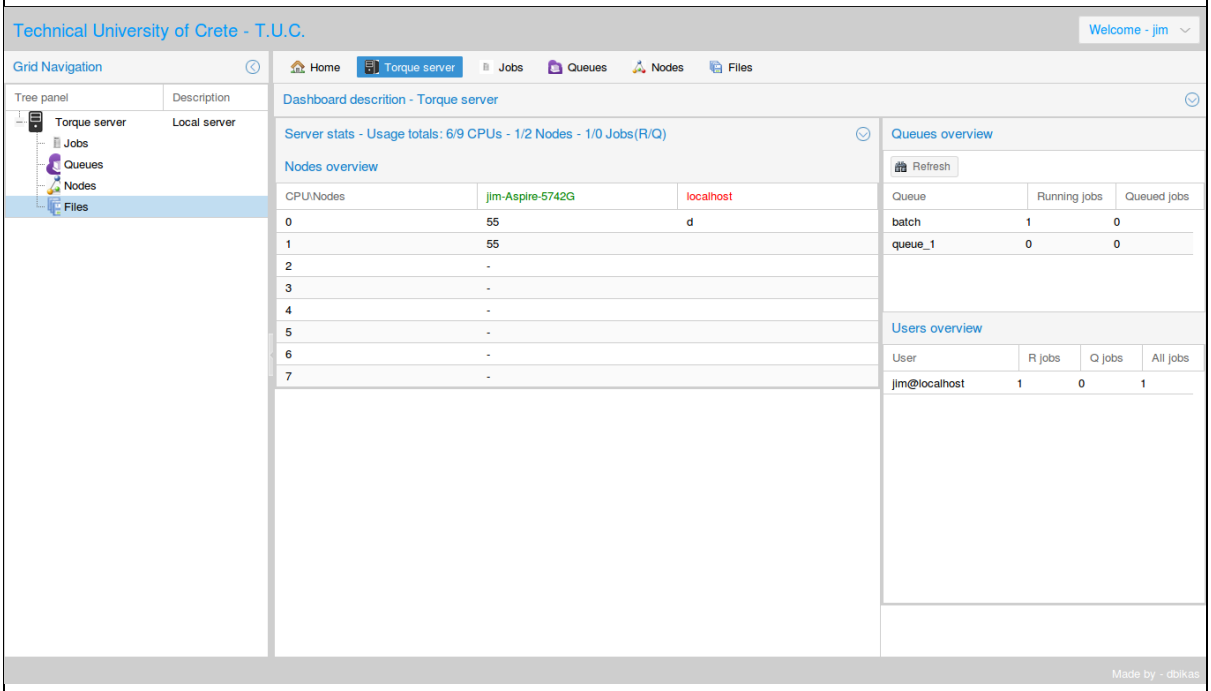
  Είναι ένας πίνακας που δείχνει αναλυτικές πληροφορίες για τους κόμβους του συστήματος.

Πιο αναλυτικά:

- Ψάχνει το σύστημα για να εντοπίσει όλους τους διαθέσιμους κόμβους και τους επεξεργαστές τους
- Δημιουργεί αυτόματα αντίστοιχο αριθμό στηλών με τον συνολικό αριθμό κόμβων του συστήματος
- Δημιουργεί αυτόματα αριθμό γραμμών αντίστοιχο με τον αριθμό επεξεργαστών που διαθέτει ο κόμβος με τους περισσότερους επεξεργαστές.
- Εάν ο κόμβος έχει διαθέσιμους επεξεργαστές και η κατάστασή του είναι running, τότε έχει πράσινο χρώμα, διαφορετικά έχει κόκκινο.

Κάθε επεξεργαστής ανά κόμβο στον πίνακα έχει 3 καταστάσεις:

- Free: - Διαθέσιμο για να εκτελέσει κάποια εργασία
- Down: d Μη διαθέσιμο διότι ο κόμβος είναι ανενεργός
- Occupied: job id Μη διαθέσιμο, διότι ήδη εκτελεί μια εργασία



Στην συγκεκριμένη εικόνα, φαίνεται ότι ο χρήστης jim, έκανε υποβολή μιας εργασίας, η οποία είναι σε κατάσταση Running και αφού είναι και η μοναδική, καταλαβαίνουμε ότι έχει αναγνωριστικό εργασίας: 55, βρίσκεται στην ουρά: batch, εκτελείται στον κόμβο: jim-Aspire-5742G και χρησιμοποιεί 2 επεξεργαστές – τον 0 και τον 1.

Τέλος, ο χρήστης μπορεί να πατήσει το κουμπί Refresh, ώστε να ανανεωθούν τα δεδομένα από τον διακομιστή.

Εναλλακτική ροή 1:

- Η επικοινωνία με το TORQUE, είναι αδύνατη
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

3.3.3 Ανάκτηση κόμβων

Τίτλος περίπτωσης χρήσης: «Ανάκτηση κόμβων»

Βασική ροή 1:

- Ο χρήστης πατάει την επιλογή: nodes από το πλαίσιο δέντρου
- Ανοίγει ένα νέο πλαίσιο, στο οποίο παρουσιάζονται πληροφορίες σχετικά με τους κόμβους το διακομιστή.

Το κύριο πλαίσιο των κόμβων αποτελείται από 5 υπό-πλαίσια.

- **Node list – πάνω δεξιά**

  Ένα πλαίσιο πίνακα το οποίο δείχνει μια λίστα όλων των κόμβων του χρήστη. - (Name, State, Node type, Np)

  Όταν ο χρήστης επιλέξει έναν κόμβο από την λίστα, η εφαρμογή ανανεώνει τα υπόλοιπα πλαίσια, ώστε να δείχνουν τις σωστές πληροφορίες του συγκεκριμένου κόμβου.

- **Node info – κάτω δεξιά**

Αυτό το πλαίσιο δείχνει περισσότερες πληροφορίες για έναν συγκεκριμένο κόμβο. Από προεπιλογή είναι άδειο. Ο χρήστης πρέπει να επιλέξει έναν κόμβο από την λίστα των κόμβων.

- **User's cpu per job – πάνω αριστερά**

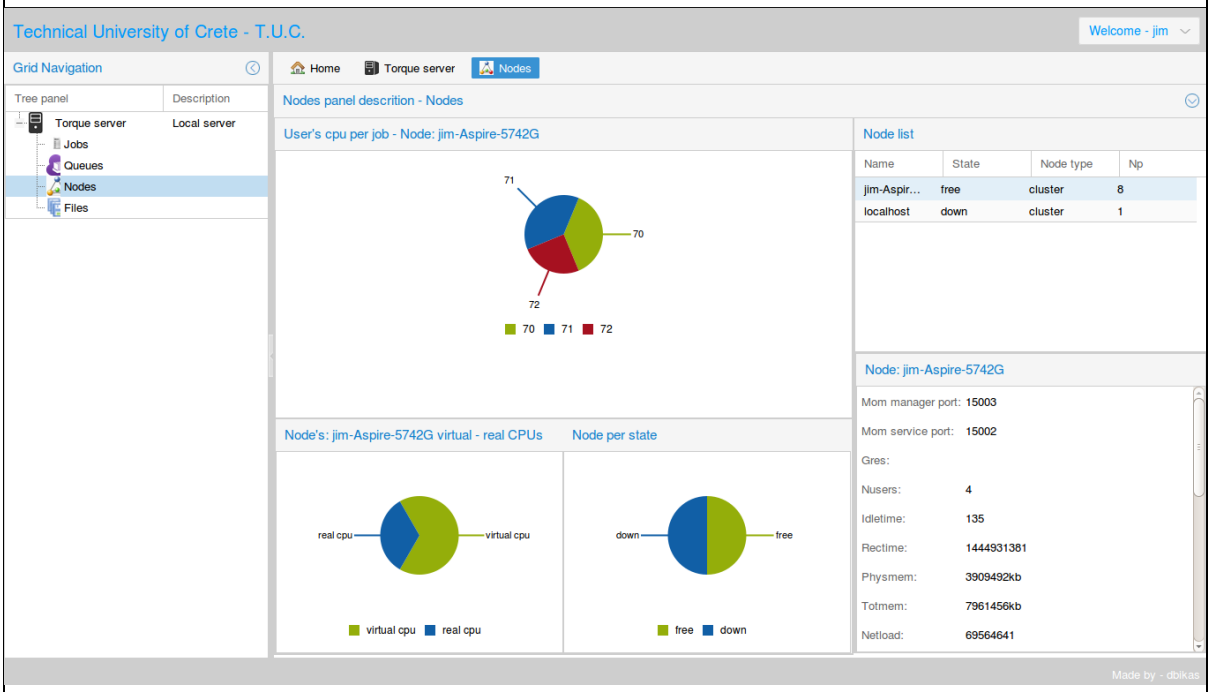
Ένα διάγραμμα πίτας, το οποίο δείχνει τις εργασίες του χρήστη, που εκτελούνται σε έναν κόμβο καθώς και τον αριθμό των επεξεργαστών που χρησιμοποιεί η κάθε εργασία. Τον κόμβο τον επιλέγει ο χρήστης από την λίστα κόμβων πάνω δεξιά και κάθε φορά πραγματοποιείται ανανέωση των δεδομένων.

- **Node's virtual / real cpus**

Ένα διάγραμμα πίτας το οποίο δείχνει πόσοι πραγματικοί και πόσοι εικονικοί επεξεργαστές υπάρχουν σε κάθε κόμβο. Πάλι τον κόμβο τον επιλέγει ο χρήστης από την λίστα κόμβων πάνω δεξιά.

- **Nodes per state**

Ένα διάγραμμα πίτας το οποίο κατηγοριοποιεί τους κόμβους του συστήματος, με βάση την κατάσταση στην οποία βρίσκονται – free, down.



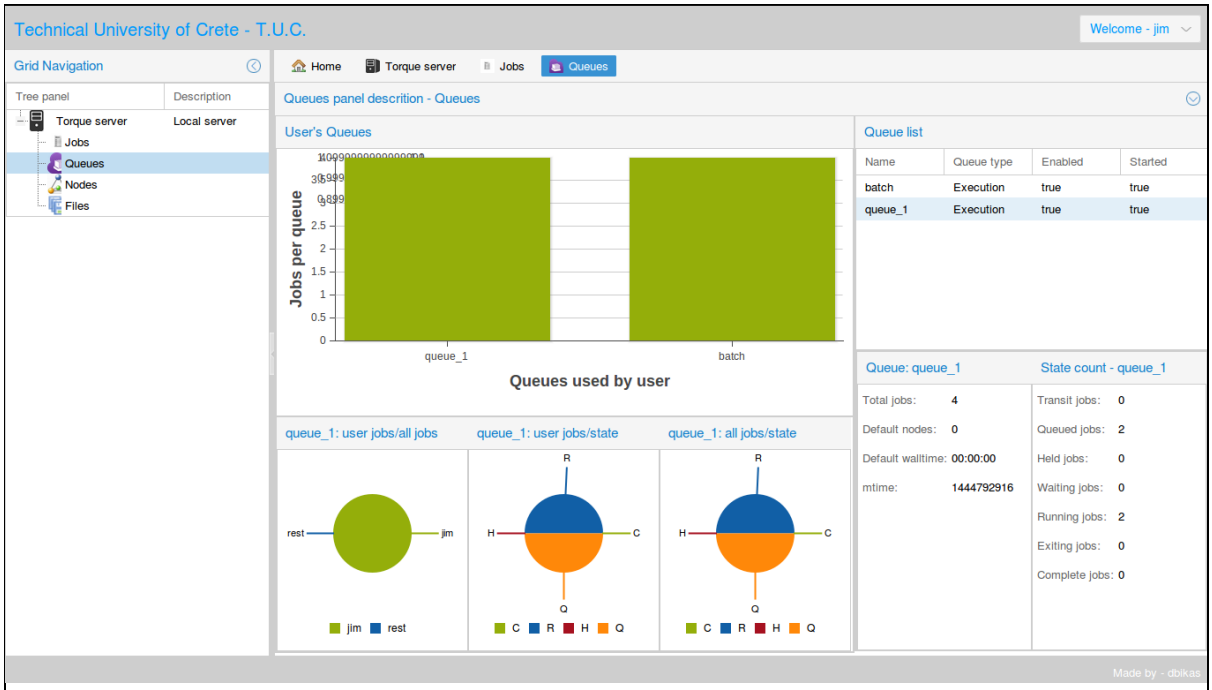
Στην συγκεκριμένη εικόνα φαίνεται ότι ο κόμβος: jim-Aspire-5742G, είναι επιλεγμένος από τον χρήστη. Τα αναλυτικά του στοιχεία φαίνονται στο πλαίσιο κάτω δεξιά, ενώ από τα διαγράμματα βλέπουμε ότι στον συγκεκριμένο κόμβο εκτελούνται 3 εργασίες με αναγνωριστικό: 70, 71, 72 καθώς και ότι οι εργασίες με αναγνωριστικό: 70 και 71 χρησιμοποιούν από 3 επεξεργαστές η κάθε μία, ενώ η εργασία με αναγνωριστικό: 72 χρησιμοποιεί 2 επεξεργαστές. Επίσης στο κάτω αριστερά διάγραμμα φαίνονται οι αριθμοί των πραγματικών και εικονικών επεξεργαστών του συγκεκριμένου κόμβου. Τέλος στο Node per state διάγραμμα φαίνεται ότι το σύστημα έχει 2 κόμβους, έναν free και έναν down, το οποίο μπορούμε να το επαληθεύσουμε και από την λίστα κόμβων στο πάνω δεξιά πλαίσιο.

Εναλλακτική ροή 1:

- Η επικοινωνία με το TORQUE, είναι αδύνατη
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

3.3.4 Ανάκτηση ουρών

Τίτλος περίπτωσης χρήσης: «Ανάκτηση ουρών»
Βασική ροή 1: • Ο χρήστης πατάει την επιλογή: queues από το πλαίσιο δέντρου • Ανοίγει ένα νέο πλαίσιο, στο οποίο παρουσιάζονται πληροφορίες σχετικά με τις ουρές του διακομιστή. Το κύριο πλαίσιο των ουρών αποτελείται από 6 υπό-πλαίσια



- **Queue list – πάνω δεξιά**

Ένα πλαίσιο πίνακα το οποίο δείχνει μια λίστα με όλες τις ουρές του συστήματος - (Name, Queue type, Enabled, Started)

Όταν ο χρήστης επιλέξει μια ουρά από την λίστα, η εφαρμογή ανανεώνει τα υπόλοιπα πλαίσια, ώστε να δείχνουν τις σωστές πληροφορίες της συγκεκριμένης ουράς.

- **Queue info – κάτω δεξιά**

Αυτό το πλαίσιο δείχνει περισσότερες πληροφορίες για μια συγκεκριμένη ουρά. Από προεπιλογή είναι άδειο. Ο χρήστης πρέπει να επιλέξει κάποια ουρά από την λίστα των ουρών. Το πλαίσιο: state count δείχνει αναλυτικά των αριθμό των εργασιών ανά κατάσταση στην οποία βρίσκονται στην συγκεκριμένη ουρά.

- **User's Queues – πάνω αριστερά**

Το συγκεκριμένο διάγραμμα παρουσιάζει την κάθε ουρά του συστήματος με βάση τον αριθμό των εργασιών που βρίσκονται σε αυτήν, (ανεξαρτήτως κατάστασης της κάθε εργασίας) και ανήκουν στον συγκεκριμένο χρήστη.

- **User jobs / all jobs in queue**

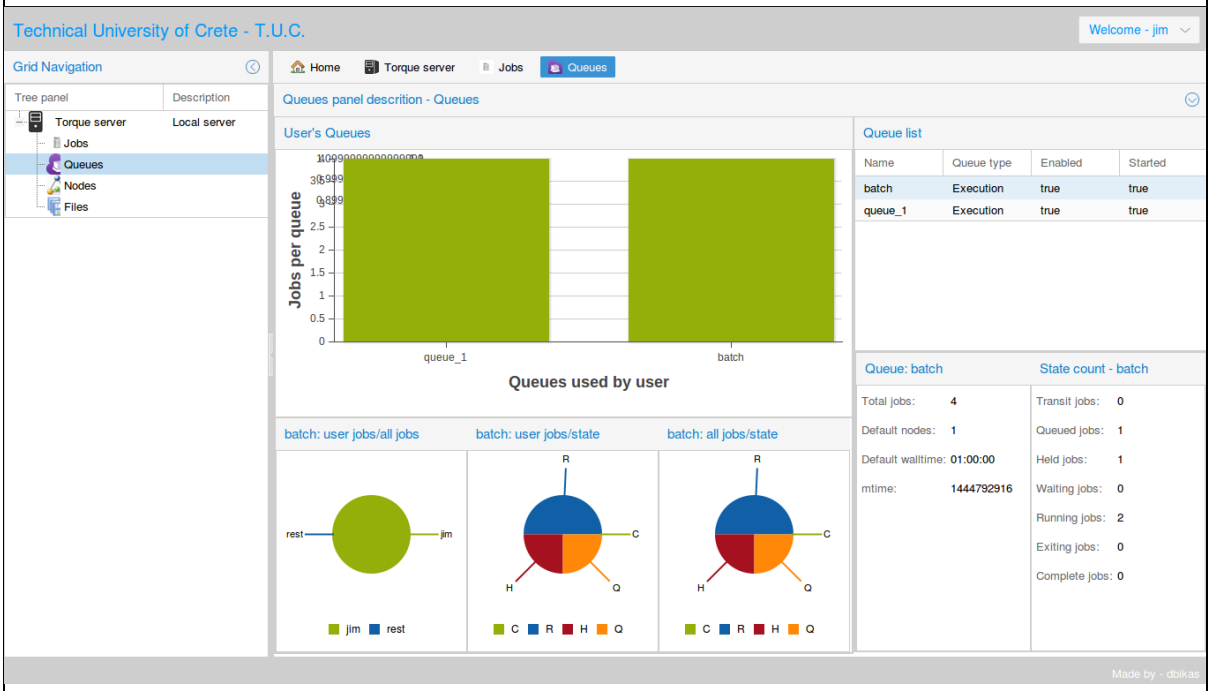
Το διάγραμμα δείχνει τον αριθμό των εργασιών του χρήστη προς τον αριθμό των εργασιών των υπόλοιπων χρηστών, που βρίσκονται στην συγκεκριμένη ουρά. Από προεπιλογή είναι άδειο. Ο χρήστης πρέπει να επιλέξει μια ουρά από την λίστα των ουρών.

- **User jobs per state in queue**

Το διάγραμμα δείχνει τον αριθμό των εργασιών του χρήστη που βρίσκονται στην συγκεκριμένη ουρά, κατηγοριοποιημένες με βάση την κατάστασή τους. Από προεπιλογή είναι άδειο. Ο χρήστης πρέπει να επιλέξει μια ουρά από την λίστα των ουρών.

- **All jobs per state in queue**

  Το διάγραμμα δείχνει τον αριθμό των εργασιών όλων των χρηστών που βρίσκονται στην συγκεκριμένη ουρά, κατηγοριοποιημένες με βάση την κατάστασή τους. Από προεπιλογή είναι άδειο. Ο χρήστης πρέπει να επιλέξει μια ουρά από την λίστα των ουρών.



Στην συγκεκριμένη εικόνα φαίνονται οι ουρές του συστήματος, στην λίστα των ουρών πάνω δεξιά. Είναι επιλεγμένη η ουρά με όνομα batch. Κάτω δεξιά βλέπουμε ότι στην συγκεκριμένη ουρά βρίσκονται 4 εργασίες – 1 σε κατάσταση Queued, 1 Held και 2 Running. Στο user's queues πλαίσιο φαίνονται οι ουρές του συστήματος με βάση τις εργασίες του χρήστη που βρίσκονται στην κάθε μία. Από την στιγμή που τα διαγράμματα και για τις 2 ουρές είναι ίδια, καταλαβαίνουμε ότι και στην ουρά queue_1, ο χρήστης έχει 4 εργασίες. Από το διάγραμμα κάτω αριστερά (user jobs/all jobs) βλέπουμε ότι στην ουρά batch, όλες οι εργασίες ανήκουν στον χρήστη. Στο δίπλα διάγραμμα φαίνονται οι εργασίες του χρήστη που βρίσκονται στην ουρά batch και είναι ίδιο με το γειτονικό του διάγραμμα, διότι ο χρήστης jim είναι ο μοναδικός χρήστης που έχει εργασίες στην συγκεκριμένη ουρά.

Εναλλακτική ροή 1:

- Η επικοινωνία με τα TORQUE, είναι αδύνατη

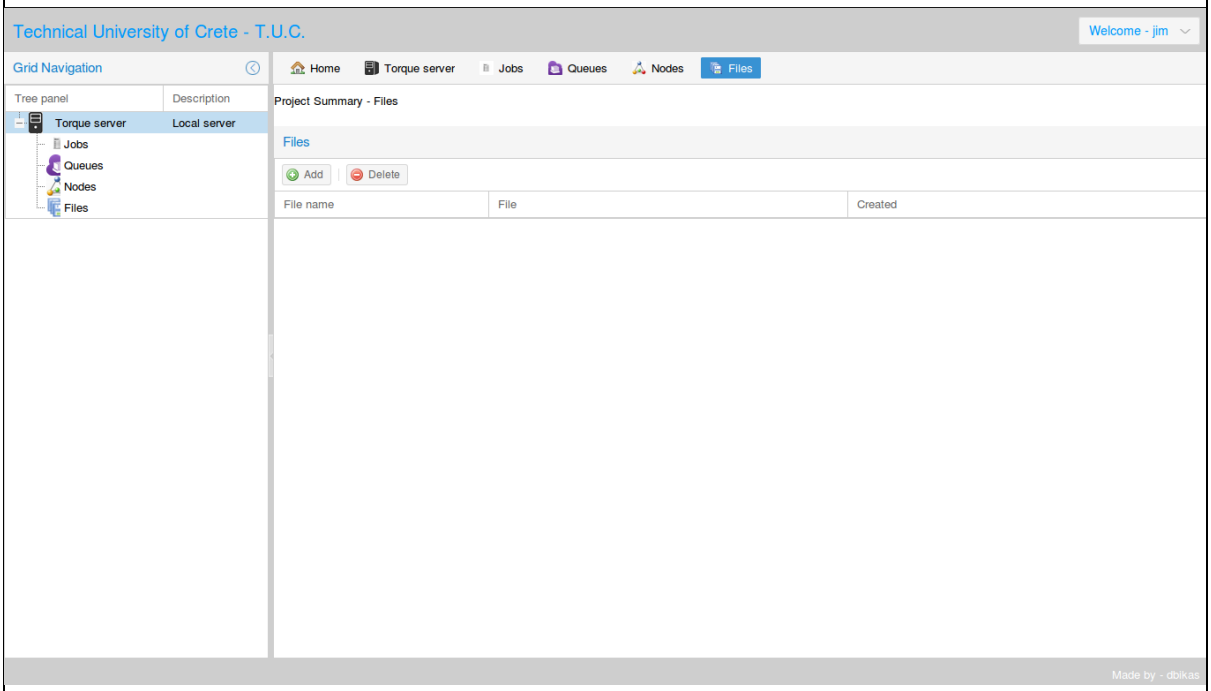
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

3.3.5 Ανάκτηση αρχείων

Τίτλος περίπτωσης χρήσης: «Ανάκτηση αρχείων»

Βασική ροή 1:

- Ο χρήστης πατάει την επιλογή: Files από το πλαίσιο δέντρου
- Ανοίγει ένα νέο πλαίσιο, στο οποίο παρουσιάζεται το σύστημα διαχείρισης αρχείων του χρήστη



- Ένα κύριο πλαίσιο που δείχνει το όνομα του κάθε αρχείου, το μονοπάτι και την ημερομηνία που το ανέβασε ο χρήστης.

Εναλλακτική ροή 1:

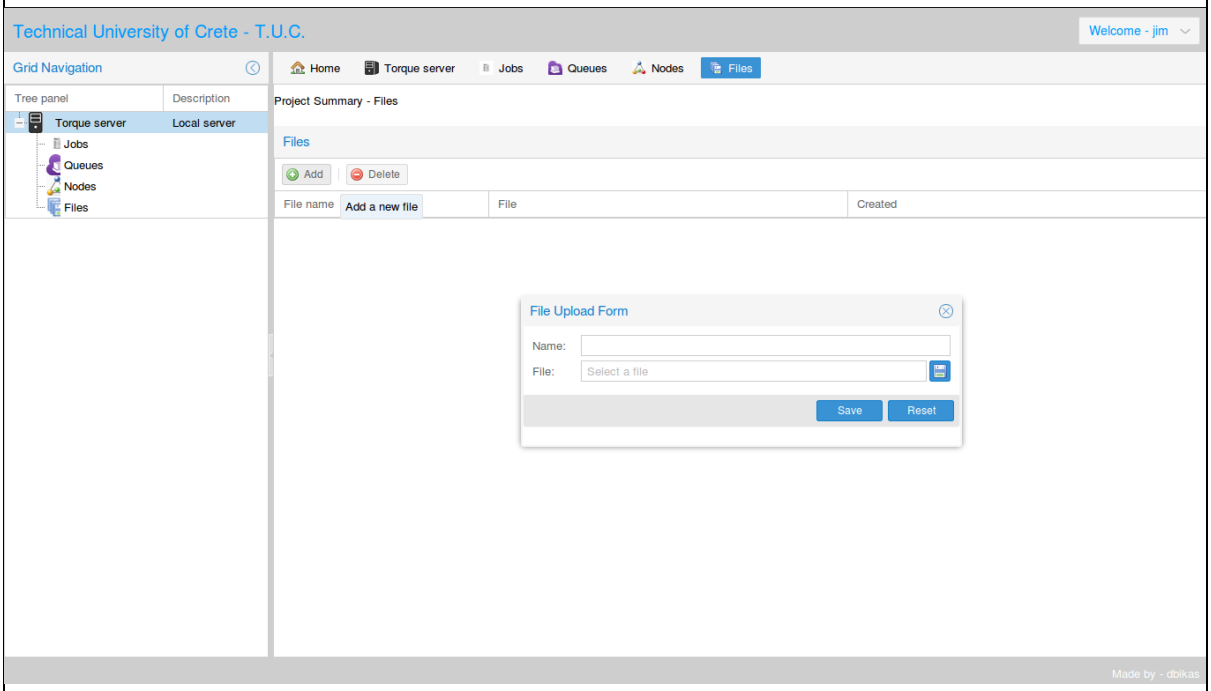
- Η επικοινωνία με το TORQUE, είναι αδύνατη
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

3.3.6 Ανέβασμα αρχείου

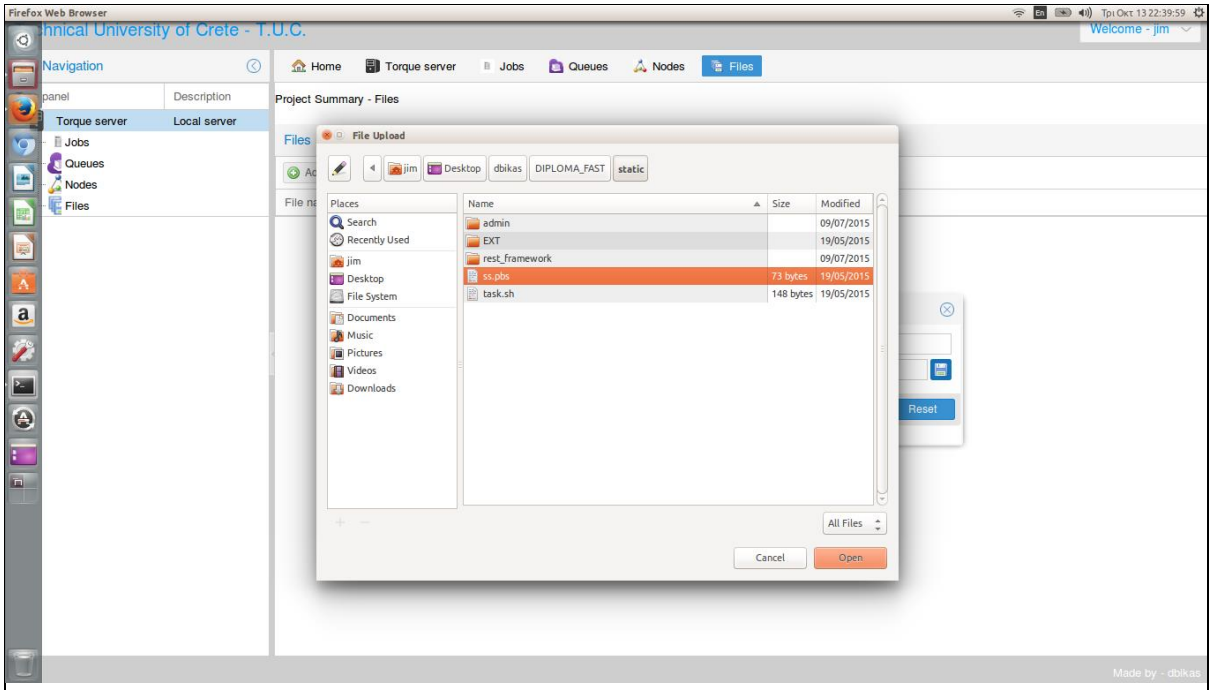
Τίτλος περίπτωσης χρήσης: «Ανέβασμα αρχείου»

Βασική ροή 1:

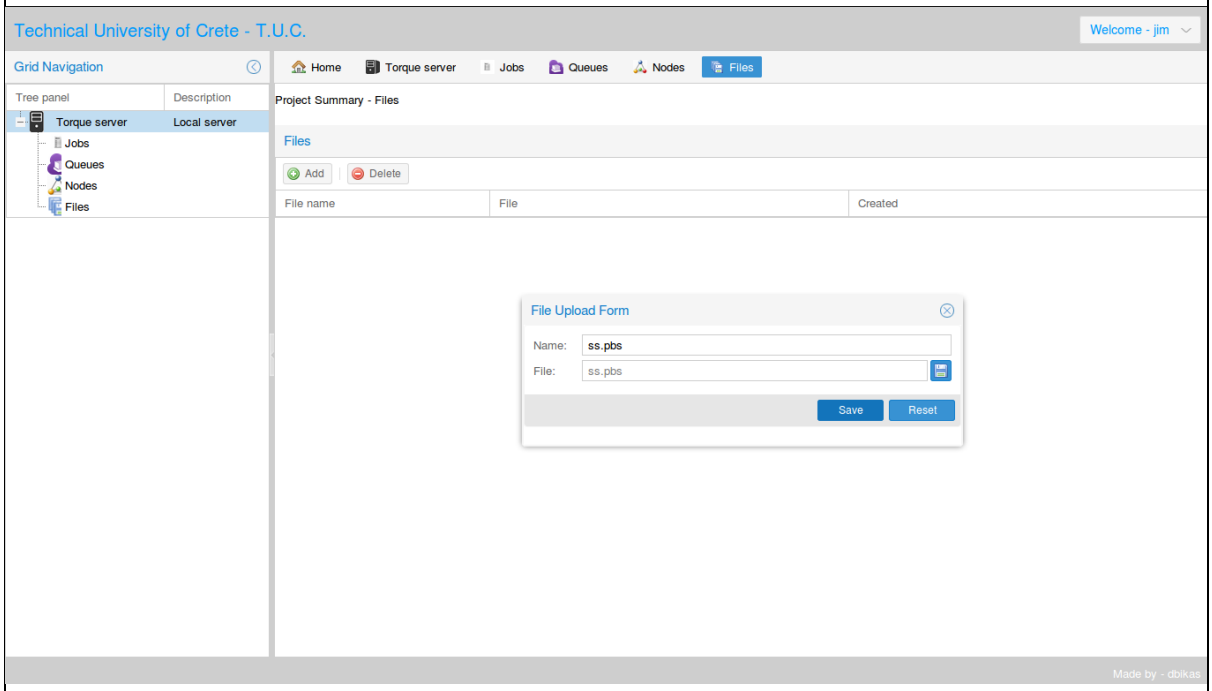
- Ο χρήστης πατάει το κουμπί Add για να εμφανιστεί η φόρμα ανεβάσματος αρχείου.
- Ο χρήστης πατάει το κουμπί επιλογής αρχείου και ανοίγει ο διαχειριστής αρχείων



του συστήματος.

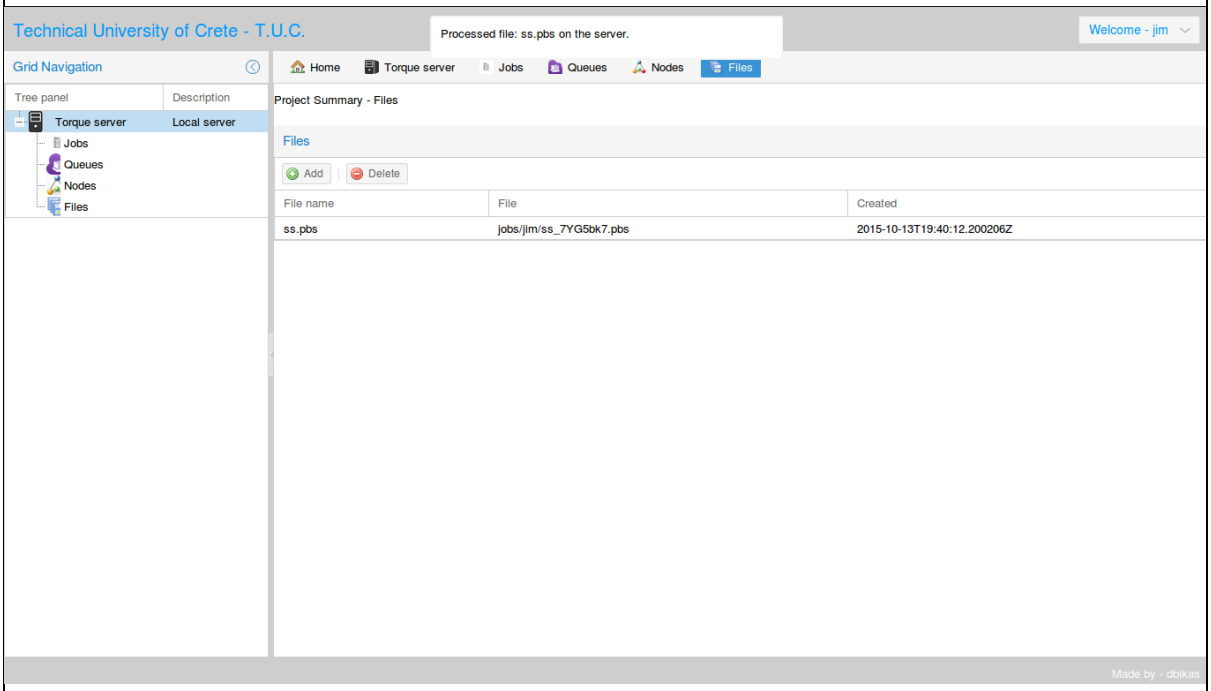


- Ο χρήστης επιλέγει κάποιο αρχείο και πατάει το κουμπί upload. Αν το επιθυμεί



μπορεί να αλλάξει το όνομα του προς ανέβασμα αρχείου.

- Το αρχείο ανεβαίνει στον διακομιστή, εμφανίζεται αντίστοιχο μήνυμα στο πάνω μέρος της εφαρμογής και γίνεται ανανέωση του πλαισίου πίνακα



Εναλλακτική ροή 1:

- Η επικοινωνία με το TORQUE, είναι αδύνατη
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

Εναλλακτική ροή 2:

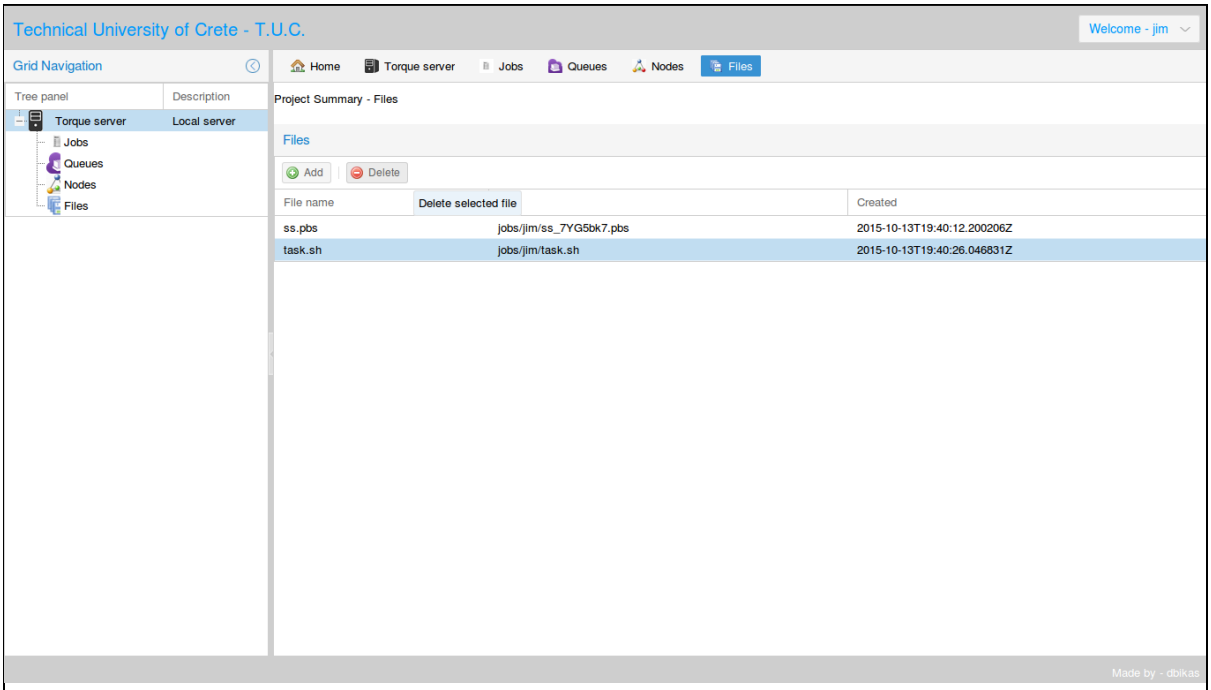
- Παρουσίαση σφάλματος ανεβάσματος του αρχείου στον διακομιστή
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

3.3.7 Διαγραφή αρχείου

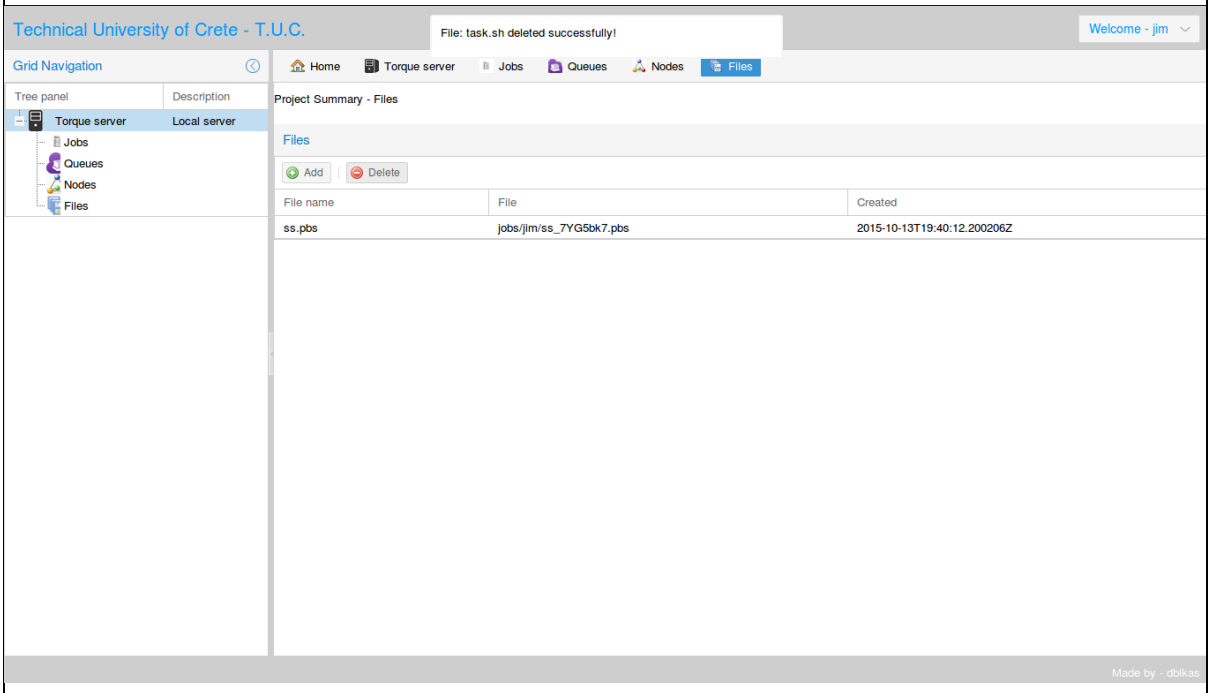
Τίτλος περίπτωσης χρήσης: «Διαγραφή αρχείου»

Βασική ροή 1:

- Ο χρήστης επιλέγει ένα αρχείο από την λίστα και πατάει το κουμπί delete



- Το αρχείο διαγράφεται από τον διακομιστή, εμφανίζεται αντίστοιχο μήνυμα στο πάνω μέρος της εφαρμογής και γίνεται ανανέωση του πλαισίου πίνακα



Εναλλακτική ροή 1:

- Η επικοινωνία με το TORQUE, είναι αδύνατη
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

Εναλλακτική ροή 2:

- Παρουσίαση σφάλματος διαγραφής του αρχείου από τον διακομιστή
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

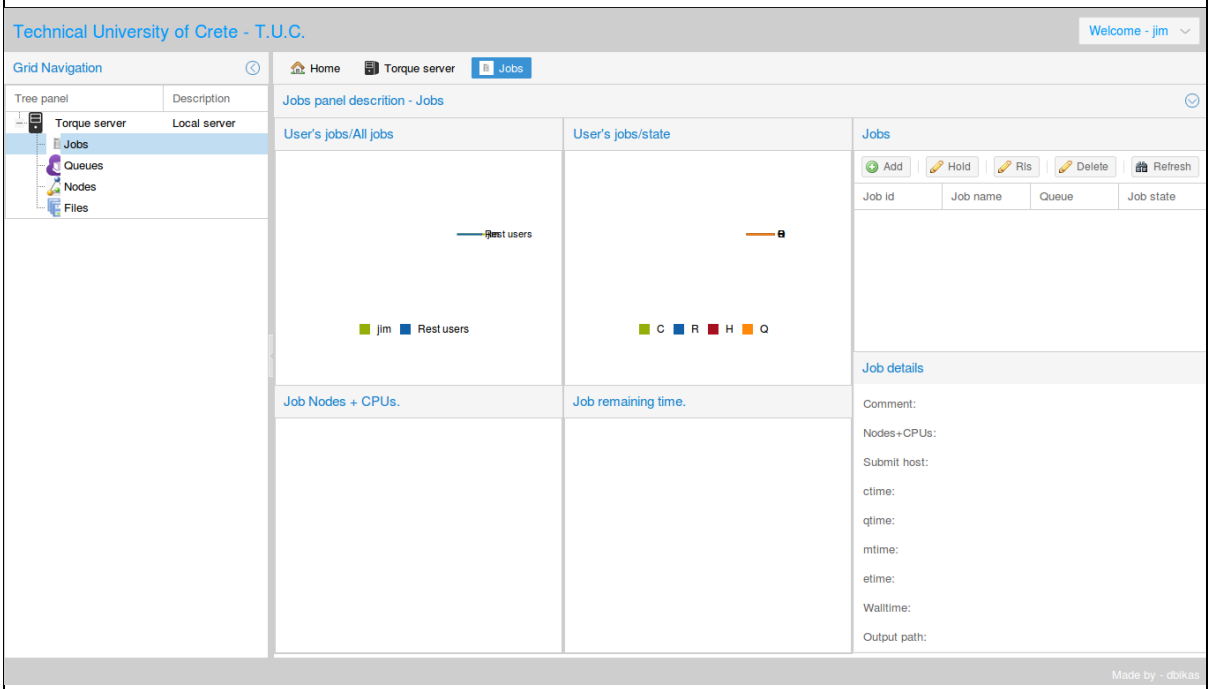
3.3.8 Ανάκτηση εργασιών

Τίτλος περίπτωσης χρήσης: «Ανάκτηση εργασιών»

Βασική ροή 1:

- Ο χρήστης πατάει την επιλογή: Jobs από το πλαίσιο δέντρου
- Ανοίγει ένα νέο πλαίσιο, στο οποίο παρουσιάζεται το σύστημα διαχείρισης εργασιών του χρήστη

Το κύριο πλαίσιο των κόμβων αποτελείται από 6 υπό-πλαίσια.



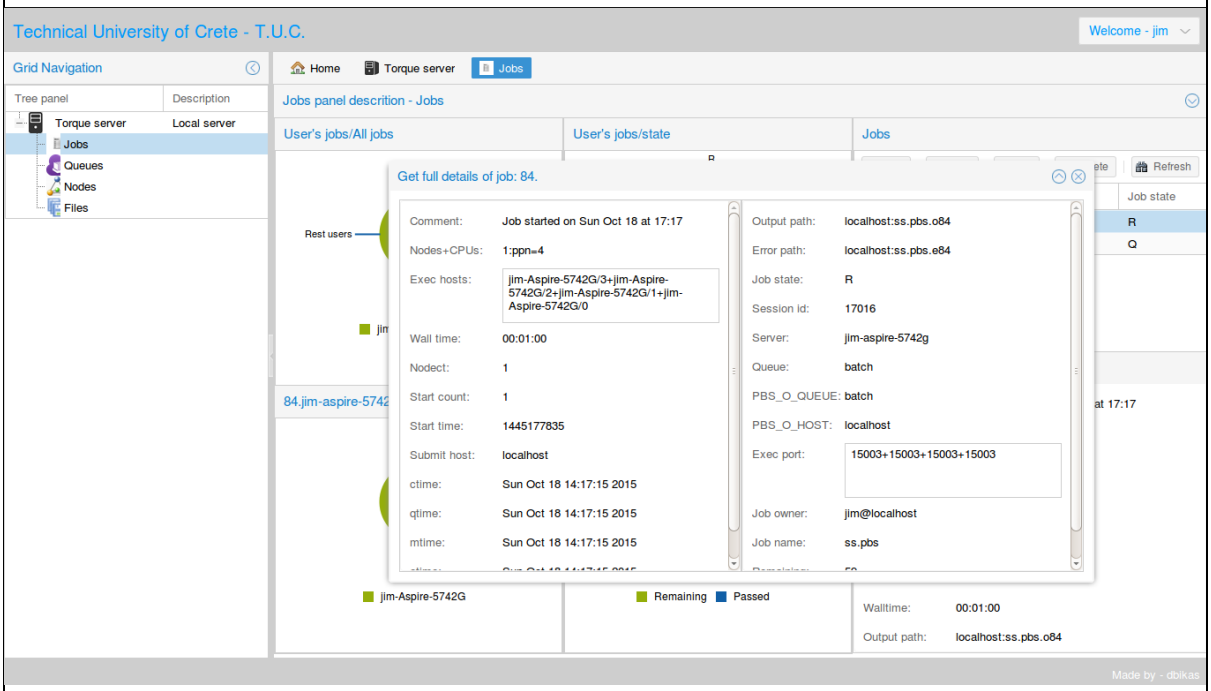
- **Job list - πάνω δεξιά**

Ένα πλαίσιο πίνακα το οποίο δείχνει μια λίστα με όλες τις εργασίες του συστήματος –(Job id, Name, Queue, State)

Όταν ο χρήστης επιλέξει μια εργασία από την λίστα, η εφαρμογή ανανεώνει τα υπόλοιπα πλαίσια, ώστε να δείχνουν τις σωστές πληροφορίες της συγκεκριμένης εργασίας.

Στο πάνω μέρος αυτού του πλαισίου υπάρχει μια γραμμή εργαλείων η οποία δίνει στον χρήστη κάποιον έλεγχο πάνω στις εργασίες του. Ο χρήστης μπορεί να προσθέσει μια νέα εργασία στο σύστημα, να αλλάξει την τρέχων κατάσταση μιας εργασίας σε Hold και να την επαναφέρει όταν εκείνος επιθυμεί, η ακόμα και να διαγράψει μια εργασία. Η αλλαγές στην κατάσταση μιας εργασίας που ο χρήστης έχει την δυνατότητα να κάνει, λειτουργούν πάντα κάτω από προϋποθέσεις οι οποίες θα αναφερθούν στην συνέχεια πιο αναλυτικά. Επίσης με το κουμπί refresh πραγματοποιείται ανανέωση των δεδομένων που αφορούν τις εργασίες του συστήματος.

Τέλος αν ο χρήστης κάνει διπλό κλικ πάνω σε μια εργασία από την λίστα εργασιών, ανοίγει ένα παράθυρο εντός της εφαρμογής και του πλαισίου των εργασιών, στο οποίο παρουσιάζονται όλες οι πληροφορίες που έχει το σύστημα για την συγκεκριμένη εργασία αναλυτικά.



- **Job details – κάτω δεξιά**
  Αυτό το panel δείχνει περισσότερες πληροφορίες για μια συγκεκριμένη εργασία. Από προεπιλογή είναι άδειο. Ο χρήστης πρέπει να επιλέξει κάποια εργασία από την λίστα των εργασιών.

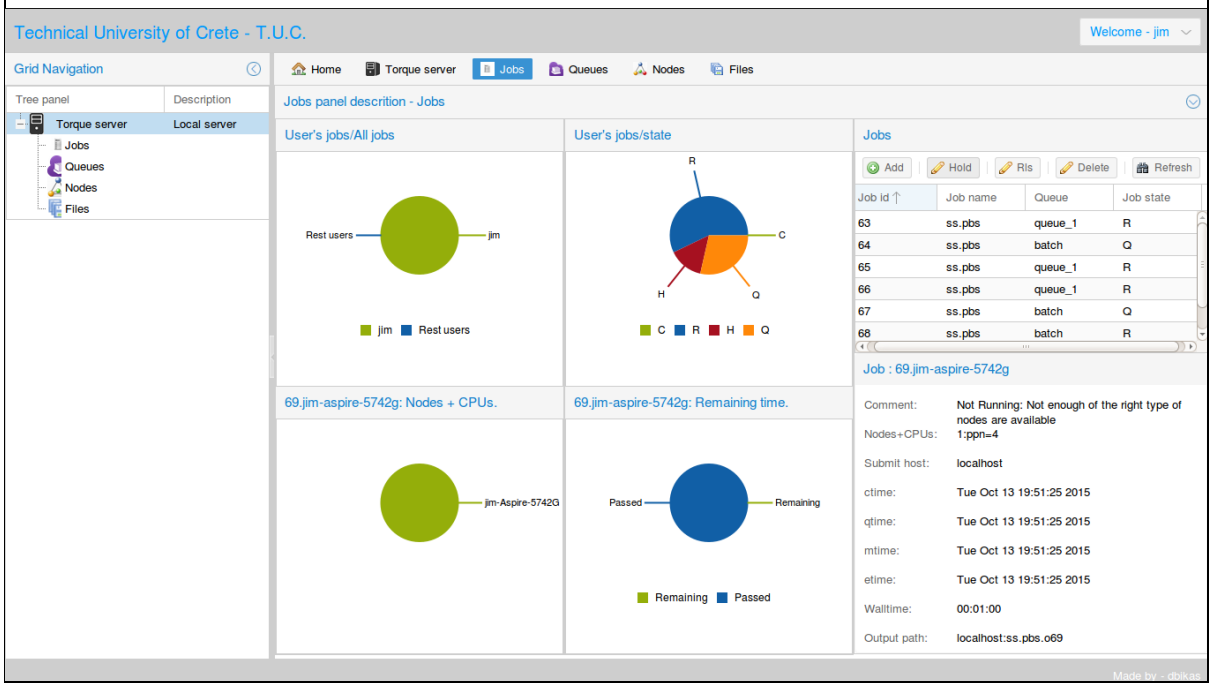
- **User's jobs / all jobs – πάνω αριστερά**

Σε αυτό το διάγραμμα, φαίνεται ο συνολικός αριθμός εργασιών του χρήστη, που υπάρχουν στο σύστημα, προς τον συνολικό αριθμό εργασιών όλων των χρηστών του συστήματος.

- **Job Nodes + cpu – κάτω αριστερά**
  Σε αυτό το διάγραμμα, φαίνονται οι κόμβοι του συστήματος, τους οποίους χρησιμοποιεί η συγκεκριμένη εργασία καθώς και οι επεξεργαστές που χρησιμοποιούνται ανά κόμβο. Από προεπιλογή είναι άδειο. Ο χρήστης πρέπει να επιλέξει κάποια εργασία από την λίστα των εργασιών.

- **User's job per state – πάνω κέντρο**
  Σε αυτό το διάγραμμα, φαίνονται όλες οι εργασίες του χρήστη που υπάρχουν στο σύστημα, ομαδοποιημένες με βάση την κατάστασή τους
  - C - Completed
  - R - Running
  - Q - Queued
  - H - Held

- **Job's remaining time - κάτω κέντρο**
  Σε αυτό το διάγραμμα φαίνεται ο χρόνος που έχει διανύσει μια συγκεκριμένη εργασία με βάση το walltime που της είχε ορίσει ο χρήστης όταν την υπέβαλε στο σύστημα. Από προεπιλογή είναι άδειο. Ο χρήστης πρέπει να επιλέξει κάποια εργασία από την λίστα των εργασιών.



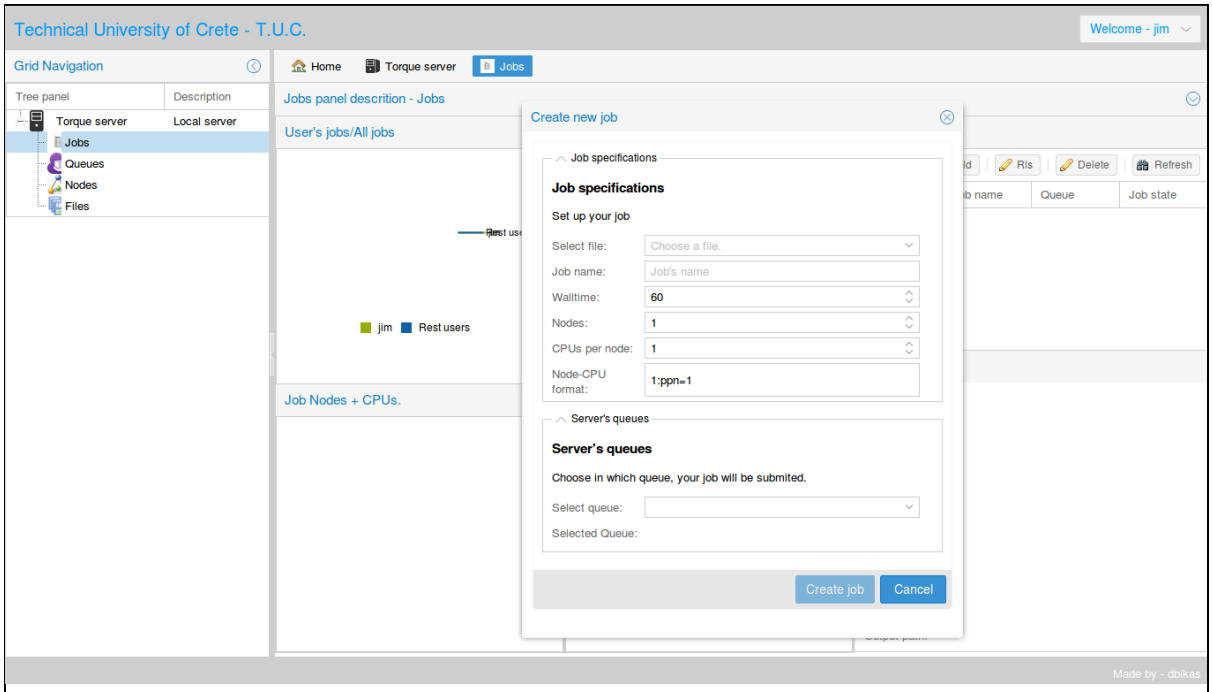
Στην εικόνα φαίνεται το πλαίσιο εργασιών ενός χρήστη, ο οποίος έχει υποβάλει κάποιες εργασίες στο σύστημα. Στο job detail πλαίσιο – κάτω δεξιά παρουσιάζονται οι πληροφορίες της τελευταίας εργασίας που ανέβηκε από τον χρήστη (στο συγκεκριμένο παράδειγμα, αυτή η εργασία βρίσκεται σε κατάσταση queued λόγω έλλειψης πόρων).

Εναλλακτική ροή 1:

- Η επικοινωνία με το TORQUE, είναι αδύνατη
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

3.3.9 Υποβολή εργασιών

Τίτλος περίπτωσης χρήσης: «Υποβολή εργασιών»
Βασική ροή 1: • Ο χρήστης πατάει το κουμπί Add από την γραμμή εργαλείων που βρίσκεται στο Jobs πλαίσιο • Ανοίγει ένα νέο παράθυρο, το οποίο δίνει την δυνατότητα στον χρήστη να υποβάλει μια νέα εργασία στο σύστημα.



Ο χρήστης καλείται να συμπληρώσει καταλλήλως τα εξής πεδία:

Job specifications

- *Select file*

  Ο χρήστης πρέπει να επιλέξει ένα αρχείο από αυτά που έχει ήδη ανεβάσει στην εφαρμογή από το πλαίσιο Files. Το συγκεκριμένο πεδίο είναι ένα select button που παρέχει μια λίστα των αρχείων που έχει ανεβάσει ο χρήστης στην εφαρμογή.

- *Job name*

  Το όνομα της εργασίας. Από προεπιλογή είναι το όνομα του αρχείου που επιλέγει ο χρήστης.

- *Walltime*

  Ο προβλεπόμενος χρόνος εκτέλεσης της εργασίας στο σύστημα. Δέχεται ακέραιους αριθμούς που συμβολίζουν δευτερόλεπτα. Από προεπιλογή είναι 60 δευτερόλεπτα.

- *Nodes*

  Ο αριθμός των κόμβων που χρειάζεται η εργασία. Δέχεται ακέραιους αριθμούς. Από προεπιλογή είναι 1 κόμβος.

- *CPU's per node*

  Ο αριθμός των επεξεργαστών ανά κόμβο που χρειάζεται η εργασία. Δέχεται ακέραιους αριθμούς. Από προεπιλογή είναι ένας επεξεργαστής

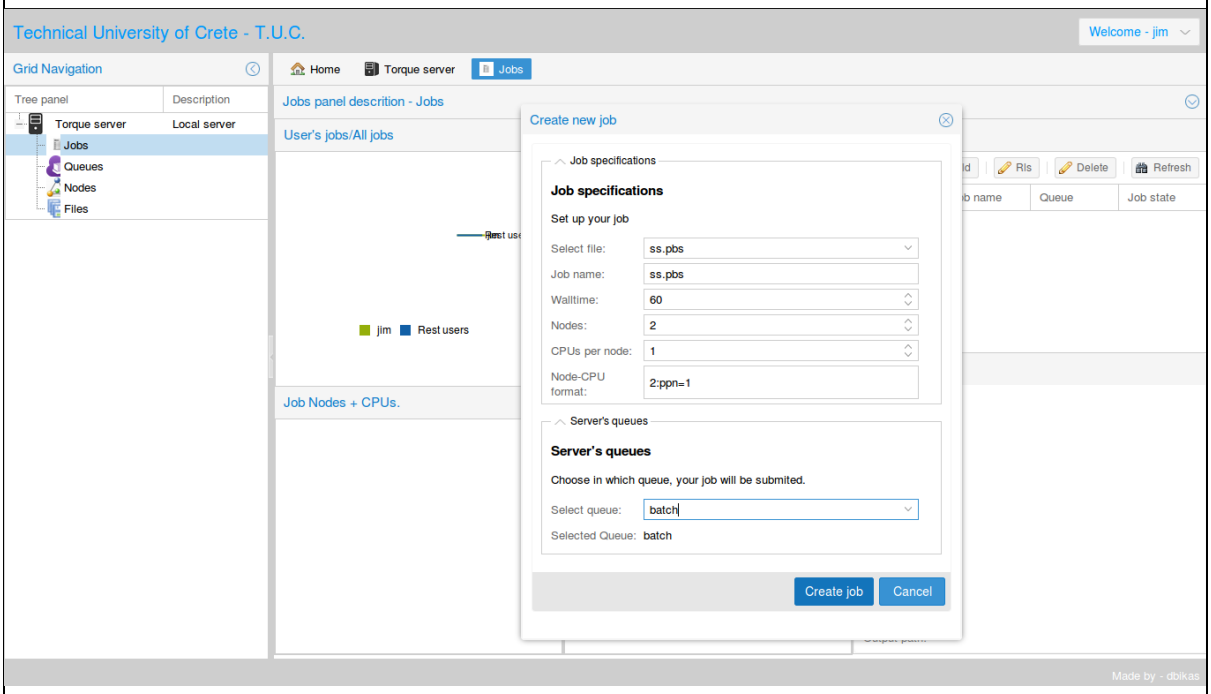
- *Node-CPU format*

  Αυτό το πεδίο διαμορφώνεται αυτόματα από τα ορίσματα που θα δώσει ο χρήστης στα 2 προηγούμενα πεδία. Είναι αλφαριθμητικό πεδίο και αυτό καθορίζει τους κόμβους και τους επεξεργαστές ανά κόμβο της εργασίας. Ο χρήστης μπορεί να συμπληρώσει αυτό το πεδίο, ανάλογα με τους πόρους που χρειάζεται η εργασία του. Για παράδειγμα θα μπορούσε η εργασία του να χρησιμοποιούσε 2 κόμβους με 2 επεξεργαστές ανά κόμβο και έναν κόμβο με 4 επεξεργαστές. Αυτό το ενδεχόμενο ο χρήστης μπορεί να το χειριστεί μόνον από αυτό το πεδίο συμπληρώνοντάς το με τον εξής τρόπο: *2:ppn=2+1:ppn=4*

  Server's queues

- *Select queue*

  Ο χρήστης πρέπει να επιλέξει μια ουρά από τις διαθέσιμες του συστήματος. Το συγκεκριμένο πεδίο είναι ένα drop down button που παρέχει μια λίστα των ουρών του συστήματος.

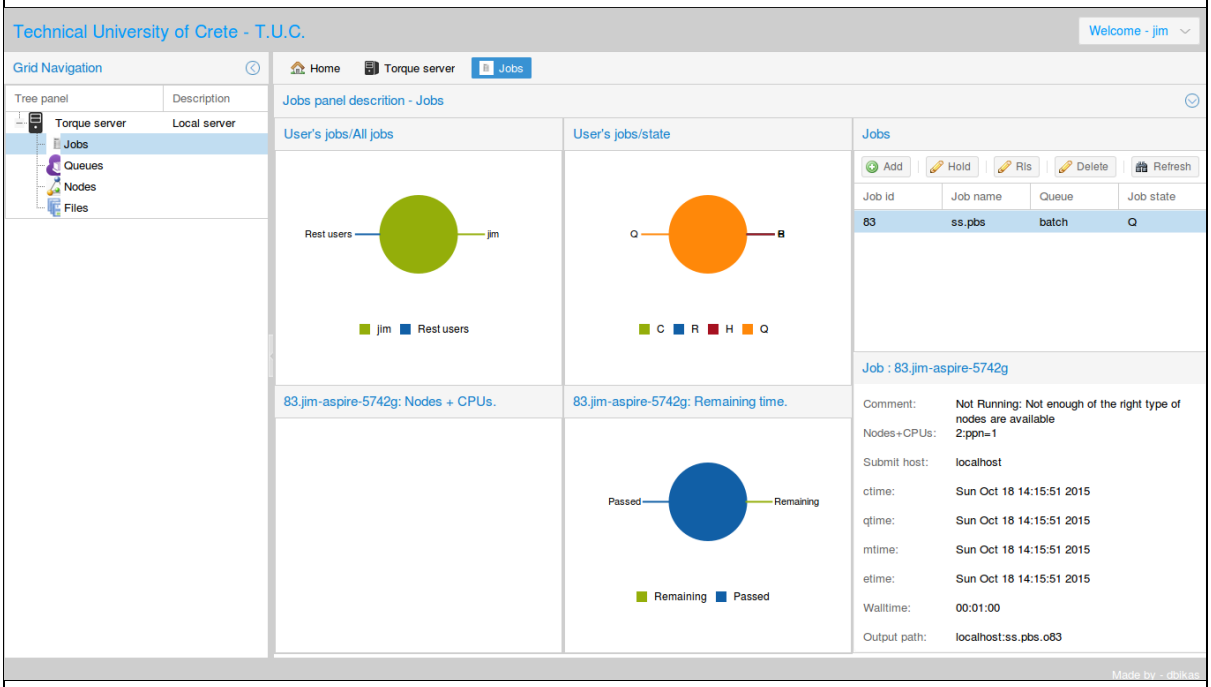


Μόλις τα πεδία συμπληρωθούν καταλλήλως, το create job κουμπί γίνεται ενεργό, ο χρήστης το επιλέγει, όλο το πλαίσιο job ανανεώνει τα δεδομένα από τον διακομιστή, κλείνει το παράθυρο και εμφανίζεται το αντίστοιχο μήνυμα στο πάνω μέρος της εφαρμογής.

Οι πόροι που χρειάζεται η εργασία είναι διαθέσιμοι και η νέα εργασία εμφανίζεται στην λίστα εργασιών πάνω δεξιά σε κατάσταση running.

Εναλλακτική ροή 1:

- Ο χρήστης πατάει το κουμπί για την υποβολή της εργασίας του στο σύστημα, οι πόροι που χρειάζεται η εργασία υπάρχουν στο σύστημα, αλλά την συγκεκριμένη στιγμή δεν είναι διαθέσιμοι. Αυτό συμβαίνει όταν οι κόμβοι του συστήματος χρησιμοποιούνται από άλλες εργασίες ή κάποιοι κόμβοι είναι σε κατάσταση down και δεν μπορούν να χρησιμοποιηθούν.
- Η εργασία υποβάλλεται στο σύστημα, αλλά βρίσκεται σε κατάσταση queued μέχρι να ελευθερωθούν πόροι του συστήματος, δηλαδή μόλις τελειώσει την εκτέλεσή της κάποια εργασία, αν κάποιος κόμβος μεταβεί σε ενεργή κατάσταση από down, ή αν προστεθούν νέοι κόμβοι στο σύστημα.



Εναλλακτική ροή 2:

- Ο χρήστης πατάει το κουμπί για την υποβολή της εργασίας του στο σύστημα και οι πόροι που χρειάζεται η εργασία δεν υπάρχουν στο σύστημα. Παραδείγματος χάρη το σύστημα διαθέτει 2 κόμβους με 4 επεξεργαστές ο κάθε ένας, και ο χρήστης προσπαθήσει να υποβάλλει εργασία η οποία απαιτεί 4 κόμβους με 4 επεξεργαστές ο κάθε ένας. Αυτή η εργασία δεν θα εκτελεστεί ποτέ γιατί οι πόροι που ζητάει δεν είναι εφικτό να υπάρξουν.

- Δεν γίνεται υποβολή της εργασίας στο σύστημα και εμφανίζεται αντίστοιχο μήνυμα στο πάνω μέρος της εφαρμογής.

Εναλλακτική ροή 3:

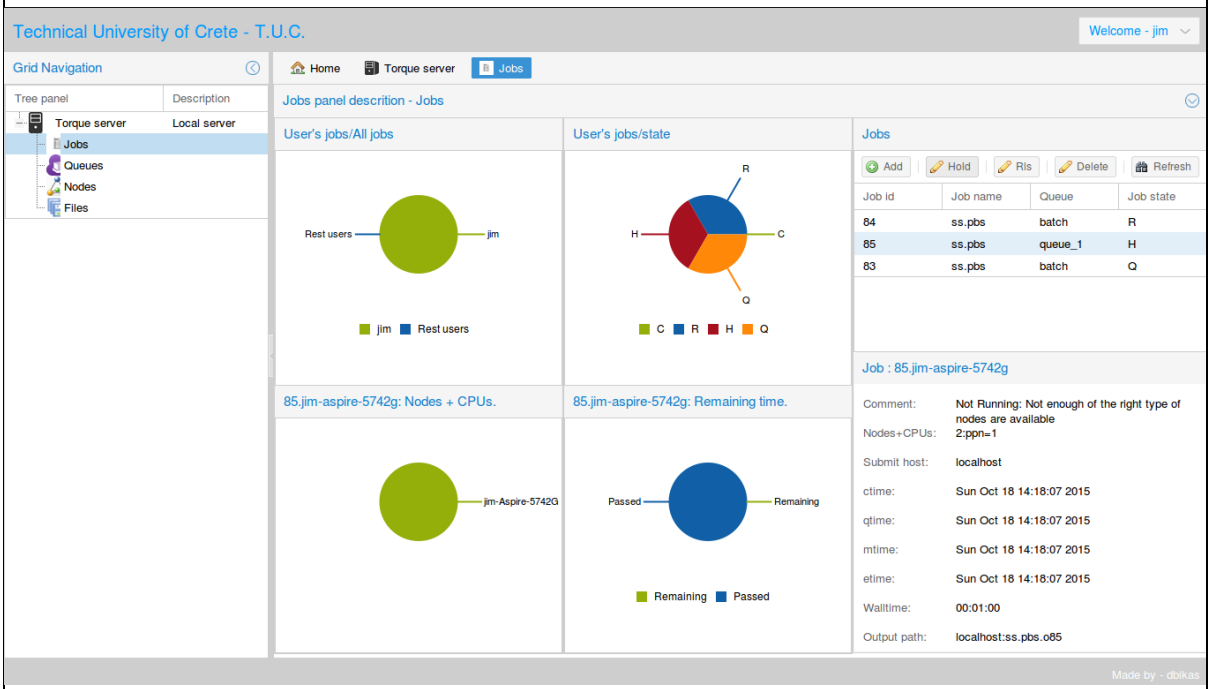
- Η επικοινωνία με το TORQUE, είναι αδύνατη
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

3.3.10 Αλλαγή κατάστασης εργασίας σε Held

Τίτλος περίπτωσης χρήσης: «Αλλαγή κατάστασης εργασίας σε Held»

Βασική ροή 1:

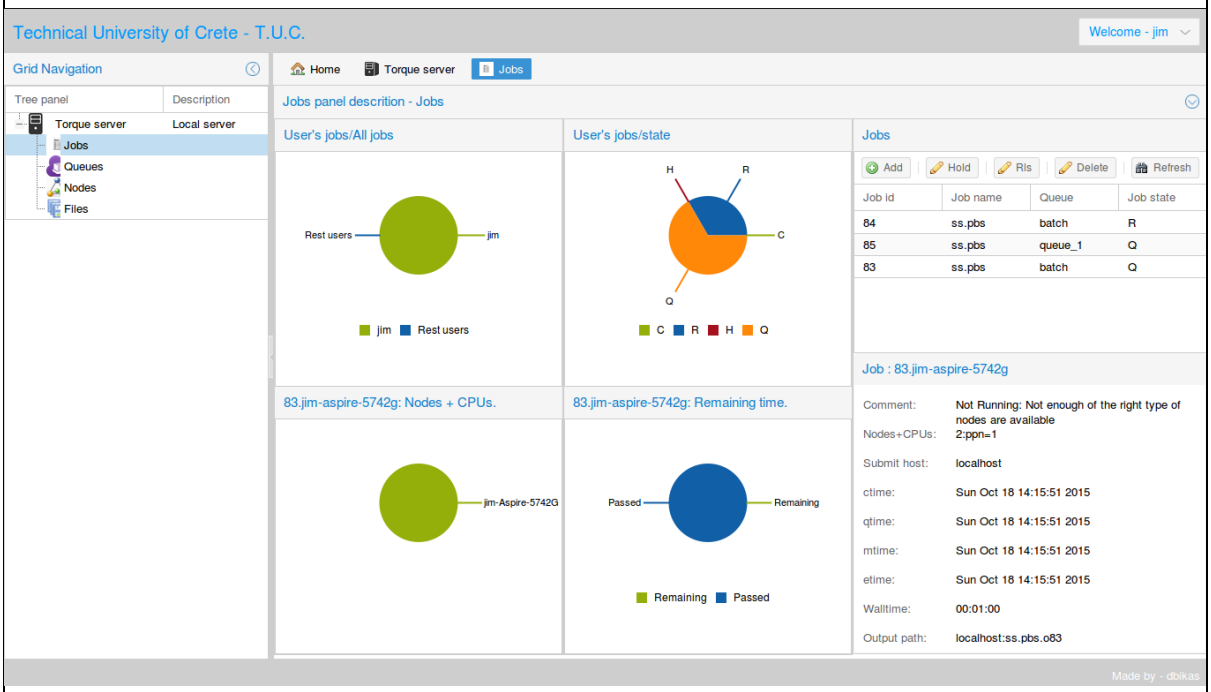
- Ο χρήστης επιλέγει μια εργασία από τον πίνακα εργασιών στο πάνω αριστερά μέρος της εφαρμογής που βρίσκεται σε κατάσταση Q (queued)
- Στην συνέχεια πατάει το κουμπί Hold
- Αλλάζει η κατάσταση της συγκεκριμένης εργασίας σε Held, γίνεται ανανέωση σε όλο το πλαίσιο job και εμφανίζεται το αντίστοιχο μήνυμα στο πάνω μέρος της εφαρμογής.



- Αξίζει να σημειωθεί ότι η συγκεκριμένη εργασία μεταβαίνει σε κατάσταση held κάτω από τα δικαιώματα του συγκεκριμένου χρήστη. Αυτό συνεπάγεται ότι μόνον

ο ίδιος χρήστης η ο διαχειριστής του συστήματος μπορούν να επαναφέρουν την εργασία από την κατάσταση held.
Εναλλακτική ροή 1: • Η επικοινωνία με το TORQUE, είναι αδύνατη • Εμφάνιση αντίστοιχου μηνύματος σφάλματος

3.3.11 Επαναφορά εργασίας από κατάσταση Held

Τίτλος περίπτωσης χρήσης: «Επαναφορά εργασίας από κατάσταση Held»
Βασική ροή 1: • Ο χρήστης επιλέγει μια εργασία από τον πίνακα εργασιών στο πάνω αριστερά μέρος της εφαρμογής που βρίσκεται σε κατάσταση Held (H) • Στην συνέχεια πατάει το κουμπί Rls (release) • Αλλάζει η κατάσταση της συγκεκριμένης εργασίας σε Q (queued) ή R (running) αν υπάρχουν οι πόροι που χρειάζεται, γίνεται ανανέωση σε όλο το πλαίσιο job και    εμφανίζεται το αντίστοιχο μήνυμα στο πάνω μέρος της εφαρμογής. • Αξίζει να σημειωθεί ότι η συγκεκριμένη εργασία μεταβαίνει από την κατάσταση held μόνον αν είχε βρεθεί σε αυτήν την κατάσταση από τον ίδιο χρήστη. Αν για παράδειγμα είχε υποβληθεί σε κατάσταση held από κάποιον διαχειριστή, ο χρήστης δεν θα είχε τα δικαιώματα να την επαναφέρει.
Εναλλακτική ροή 1:

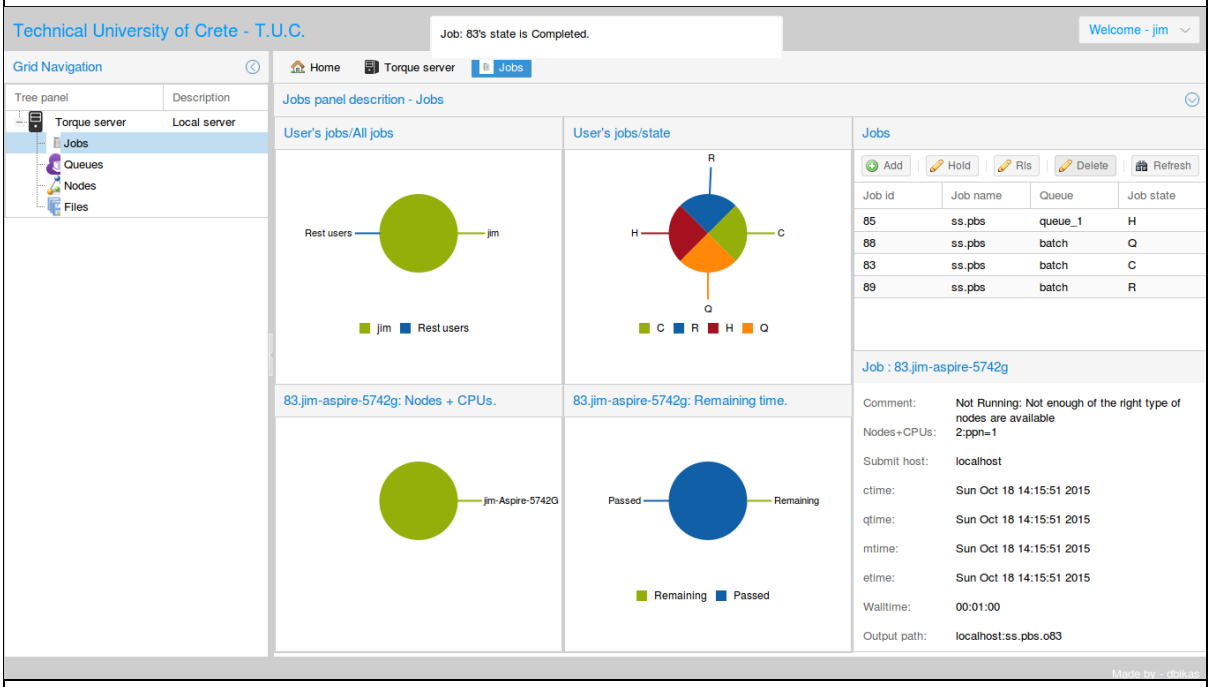
- Η επικοινωνία με το TORQUE, είναι αδύνατη
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

3.3.12 Διαγραφή εργασίας

Τίτλος περίπτωσης χρήσης: «Διαγραφή εργασίας»

Βασική ροή 1:

- Ο χρήστης επιλέγει μια εργασία από τον πίνακα εργασιών στο πάνω αριστερά μέρος της εφαρμογής που βρίσκεται σε κατάσταση Held(H) ή Queued(Q)
- Στην συνέχεια πατάει το κουμπί Delete
- Η εργασία μεταβαίνει στην κατάσταση Completed(C) και μετά από σύντομο χρονικό διάστημα διαγράφεται από το σύστημα, γίνεται ανανέωση σε όλο το πλαίσιο job και εμφανίζεται το αντίστοιχο μήνυμα στο πάνω μέρος της εφαρμογής.



Εναλλακτική ροή 1:

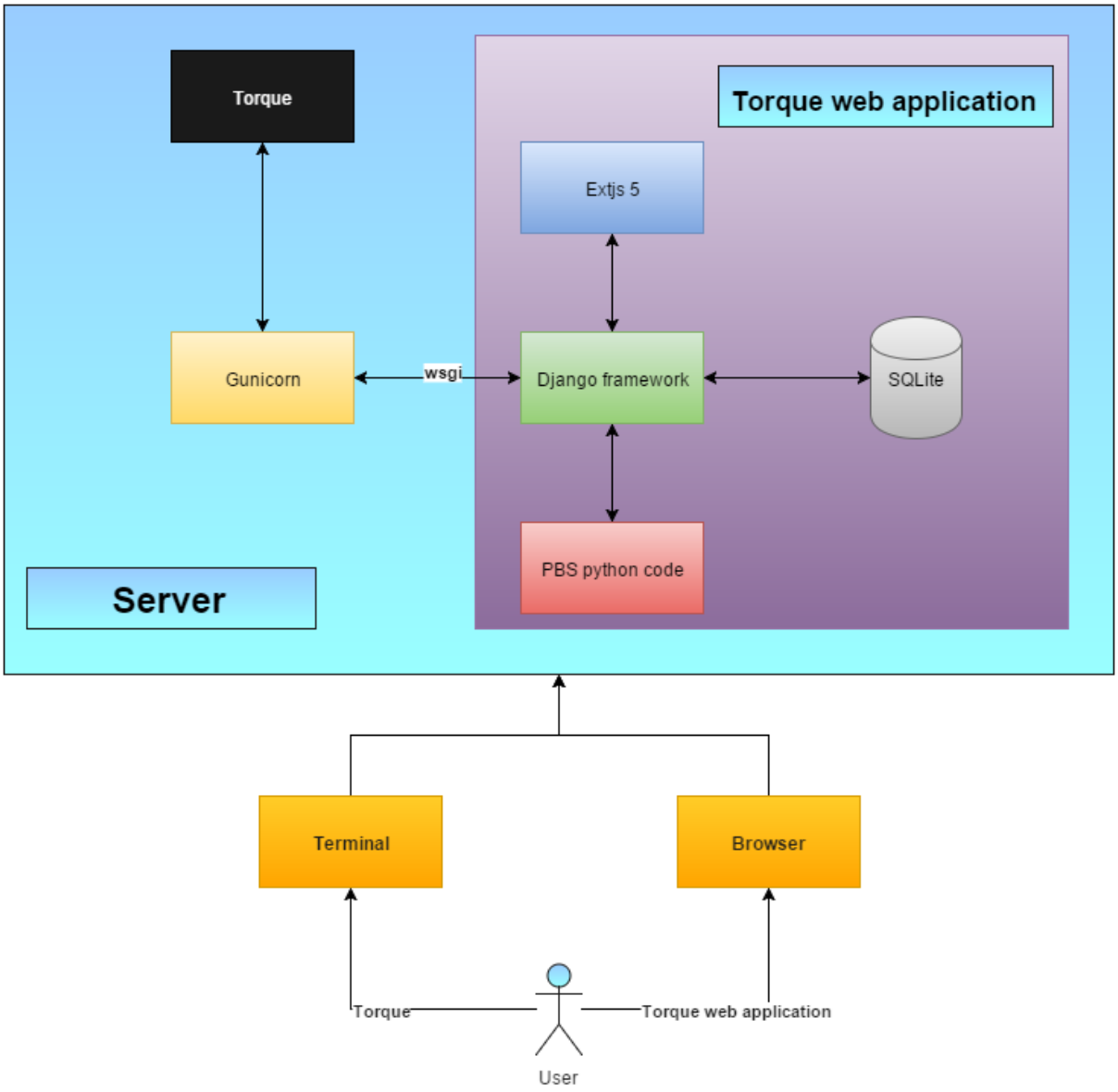
- Η επικοινωνία με το TORQUE, είναι αδύνατη
- Εμφάνιση αντίστοιχου μηνύματος σφάλματος

ΚΕΦΑΛΑΙΟ 4 – Υλοποίηση

Σε αυτό το κεφάλαιο γίνεται η περιγραφή της υλοποίησης της εφαρμογής, ο τρόπος με τον οποίο χρησιμοποιήθηκαν τα εργαλεία και οι τεχνολογίες που έχουν αναφερθεί, καθώς και ο τρόπος με τον οποίο συνδέονται και αλληλεπιδρούν μεταξύ τους, ώστε να έχουμε το επιθυμητό αποτέλεσμα.

4.1 Δομή εγκατάστασης εφαρμογής

Εικόνα 6 - Δομή εγκατάστασης εφαρμογής



Στο συγκεκριμένο διάγραμμα απεικονίζεται ο τρόπος με τον οποίο δομείται το σύστημα του απομακρυσμένο διακομιστή, στον οποίο βρίσκονται εγκατεστημένα το TORQUE και η εφαρμογή, καθώς και ο τρόπος με τον οποίο επικοινωνεί ο χρήστης με το σύστημα.

Αρχικά ο χρήστης μπορεί να συνδεθεί στον διακομιστή με τον κλασικό τρόπο, από το τερματικό, χρησιμοποιώντας το SSH πρωτόκολλο, ώστε να μπορεί να χρησιμοποιεί το TORQUE και να διαχειρίζεται τα αρχεία του.

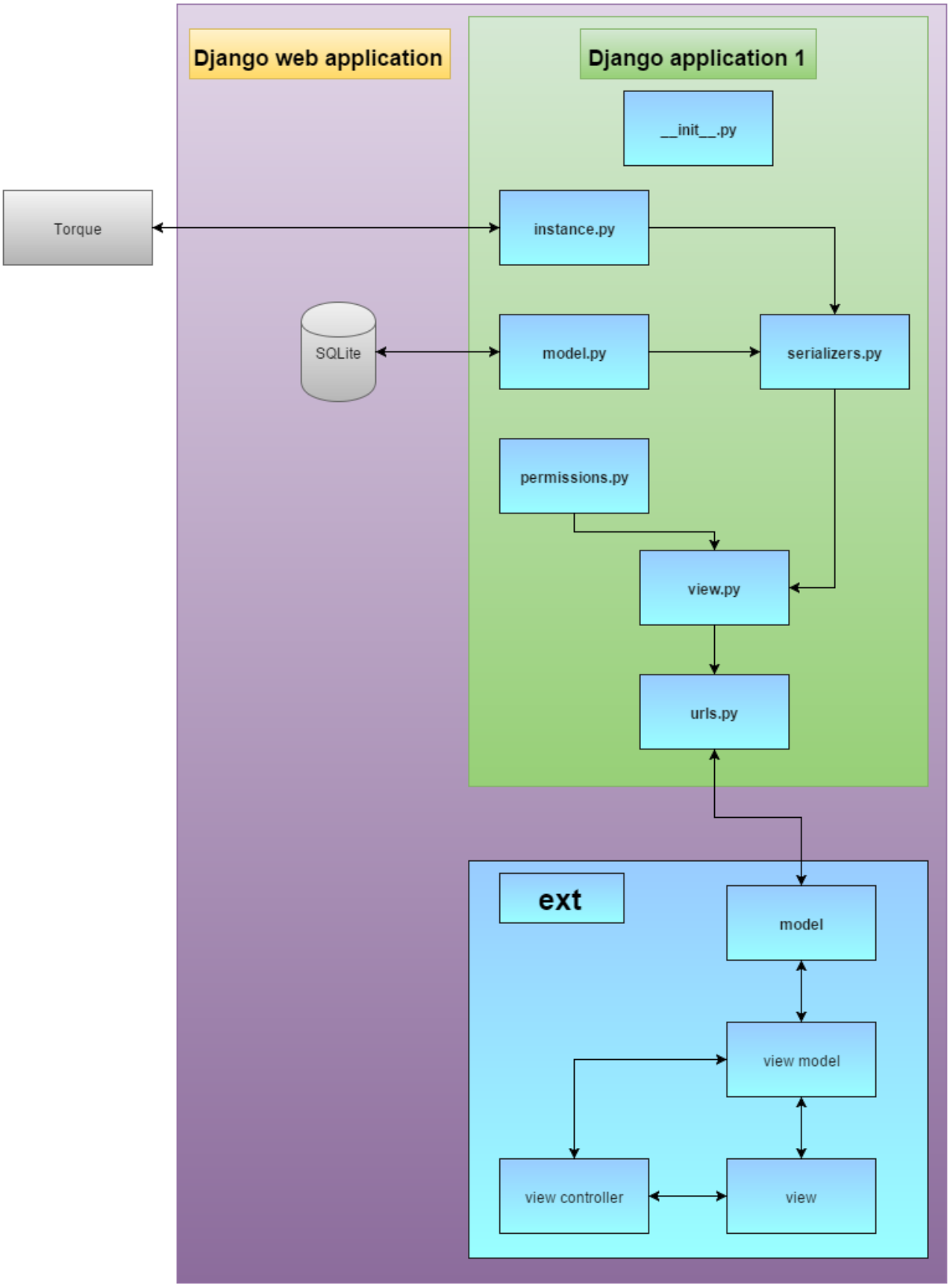
Ο χρήστης έχει την επιλογή να πληκτρολογήσει την διεύθυνση της εφαρμογής σε κάποιον φυλλομετρητή ιστού, να πραγματοποιήσει είσοδο στο σύστημα, να χρησιμοποιεί το TORQUE και να διαχειρίζεται τα αρχεία του μέσα από ένα γραφικό περιβάλλον.

Αυτή η λειτουργία επιτυγχάνεται ως εξής σε ένα σύστημα:

- Εγκατάσταση και ρύθμιση του TORQUE
- Εγκατάσταση της Python2.7 και των python-tools (e.g. virtualenv, pip, setuptools, …)
- Εγκατάσταση της βιβλιοθήκης PBS_python
- Χρήση του virtualenv για την δημιουργία ενός εικονικού περιβάλλοντος στο οποίο θα πραγματοποιηθεί εγκατάσταση και ρύθμιση όλων των εργαλείων που χρειάζεται η εφαρμογή:
    - Django – Το framework της εφαρμογής
    - Django-rest-framework – Βιβλιοθήκη που οργανώνει και διευκολύνει την δημιουργία REST-API
    - PAM – Βιβλιοθήκη της python που μας επιτρέπει να επιβεβαιώσουμε τα credentials του χρήστη στο σύστημα.
    - Gunicorn – Python WSGI HTTP Server for Unix.

4.2 Βασική δομή εφαρμογής

Εικόνα 7 - Δομή εφαρμογής



Το Django project της εφαρμογής αποτελείται από 5 applications οι οποίες έχουν την δομή του Django application 1, που φαίνεται στο διάγραμμα. Οι εφαρμογές api, jobs_api,

nodes_api και queues_api, που αλληλεπιδρούν με το TORQUE, περιέχουν το αρχείο instance.py, ώστε να ανταλλάσουν δεδομένα με το TORQUE, ενώ η εφαρμογή file_api έχει το models.py διότι συνδέεται με την βάση δεδομένων SQLite.

Η κάθε μία εφαρμογή μας παρέχει ένα API, το οποίο «καταναλώνεται» από την Ext JS. Η διαδικασία γίνεται ως εξής:

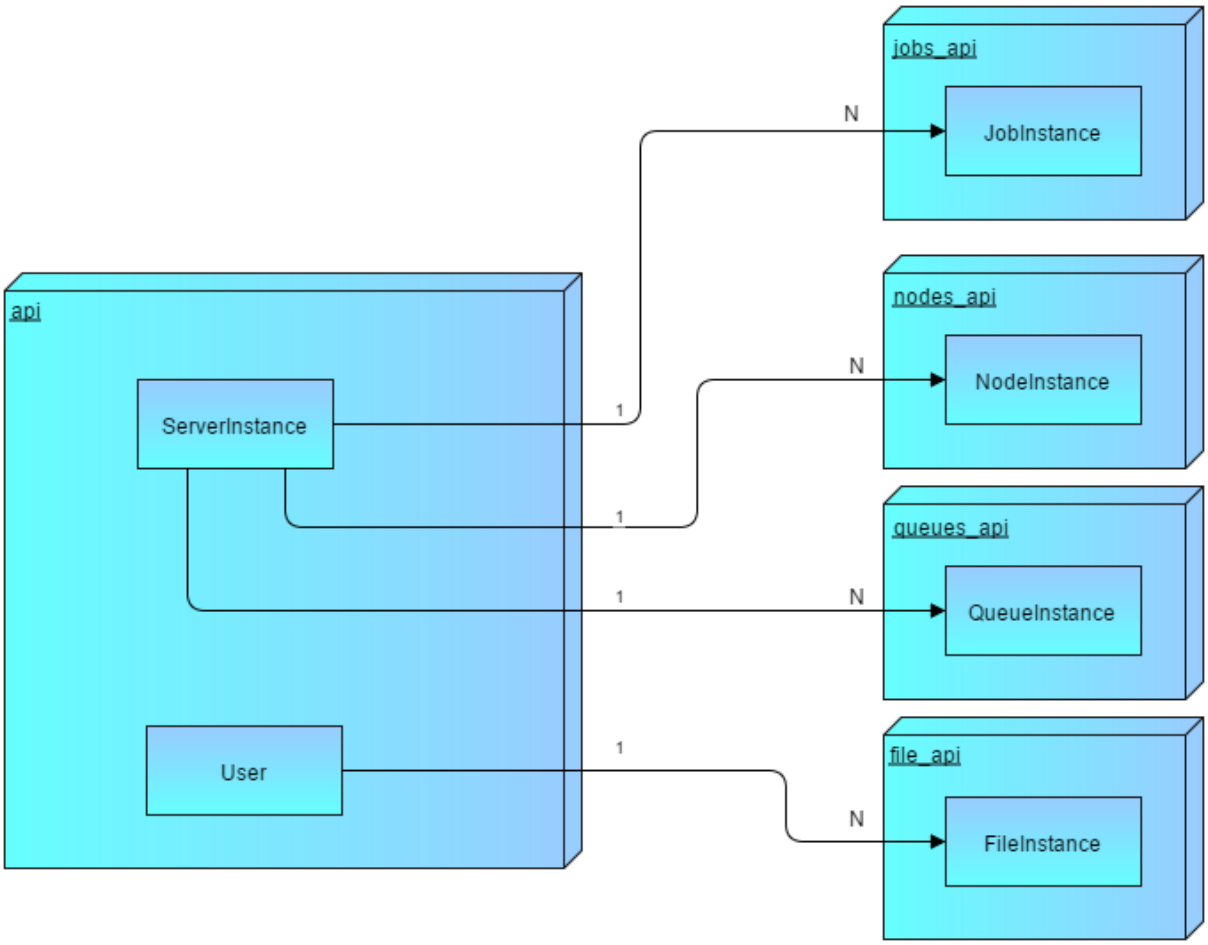
- Η εφαρμογή αποκτά πρόσβαση στα δεδομένα του TORQUE ή της βάσης δεδομένων, με την χρήση της pbs-PBSQuery και των Django-models αντίστοιχα.
- Γίνεται η μορφοποίηση των δεδομένων από το αρχείο serializers.py σε json.
- Τα δεδομένα από τα αρχεία instance.py, models.py και serializers.py εισάγονται στα αρχεία views.py, στα οποία βρίσκεται η λογική της εφαρμογής.
- Τα αρχεία urls.py περιέχουν τα urls που φαίνονται στο REST API και αντιστοιχούνται με τις κατάλληλες συναρτήσεις των αρχείων views.py.

Εν συνεχεία το REST API, καταναλώνεται από το front-end της εφαρμογής, και συγκεκριμένα από την Ext JS.

- Αρχικά τα models της Ext, ορίζουν την δομή των δεδομένων και εκτός από ορισμένες εξαιρέσεις, προσκομίζουν τα δεδομένα από το REST API.
- Τα view-models της Ext, παίρνουν τα δεδομένα από τα models, τα μορφοποιούν κατάλληλα ώστε να χρησιμοποιηθούν από τα views της Ext ή τους view-controllers.
- Τα views αναλαμβάνουν την δημιουργία και την διαμόρφωση των γραφικών της εφαρμογής και της διεπαφής χρήστη - user interface (UI).
- Οι view-controllers, αναλαμβάνουν τον έλεγχο των γεγονότων που συμβαίνουν στα views, καθώς και την προσκόμιση – επεξεργασία δεδομένων από τα view-models.

4.3 Διάγραμμα κλάσεων

Εικόνα 8 - Αφαιρετικό διάγραμμα κλάσεων



4.3.1 api

Οργάνωση

Instance.py, serializers.py, views.py, urls.py

Λειτουργικότητα

Αυτή η application, αλληλεπιδρά με το TORQUE, χρησιμοποιώντας την βιβλιοθήκη PBSQuery, για να πάρει όλα τα δεδομένα από τον TORQUE διακομιστή:

- Βασικές πληροφορίες του TORQUE και του συστήματος
- Δεδομένα των κόμβων
- Δεδομένα των ουρών
- Δεδομένα των εργασιών

τα οποία κατηγοριοποιεί με την βοήθεια - κλήση των jobs_api, nodes_api και queues_api στιγμιοτύπων.

Επίσης με την χρήση της βιβλιοθήκης pbs, επιτρέπει στον χρήστη την δημιουργία - υποβολή εργασιών στο σύστημα, την αλλαγή κατάστασής τους σε Held, την επαναφορά τους στην αρχική, καθώς και την διαγραφή τους.

ServerInstance – αναλυτικό διάγραμμα κλάσης


ServerInstance
+ c: integer
+ p: < PBSQuery object >
+ name: string
+ tcp_timeout: integer
+ managers: dictionary
+ keep_completed: integer
+ poll_jobs: string
+ node_check_rate: integer
+ scheduling: string
+ job_stat_rate: integer
+ total_jobs: integer
+ acl_hosts: dictionary
+ moab_array_compatible: string
+ mom_job_sync: string
+ state_count: dictionary
+ default_queue: string
+ server_state: string
+ operators: dictionary
+ net_counter: string
+ next_job_number: integer
+ submit_hosts: dictionary
+ scheduler_iteration: integer
+ mail_from: string
+ pbs_version: string
+ log_events: string
+ allow_node_submit: string
+ create_job(self, username, Job_Name, queue, nodes, walltime, file): integer
+ hold_rls_job(self, job_id, server, mode, permission): integer
+ del_job(self, job_id, server): integer
+ job_list(self, usernames=None, job_id=None): list of JobInstance
+ node_list(self, node_name=None, username=None): list of NodeInstance
+ queue_list(self, queue_name=None): list of QueueInstance
+ getjobs(self): dictionary
+ getjob(self, id): dictionary
+ getnodes(self): dictionary
+ getnode(self, id): dictionary
+ getqueues(self): dictionary
+ getqueue(self, id): dictionary

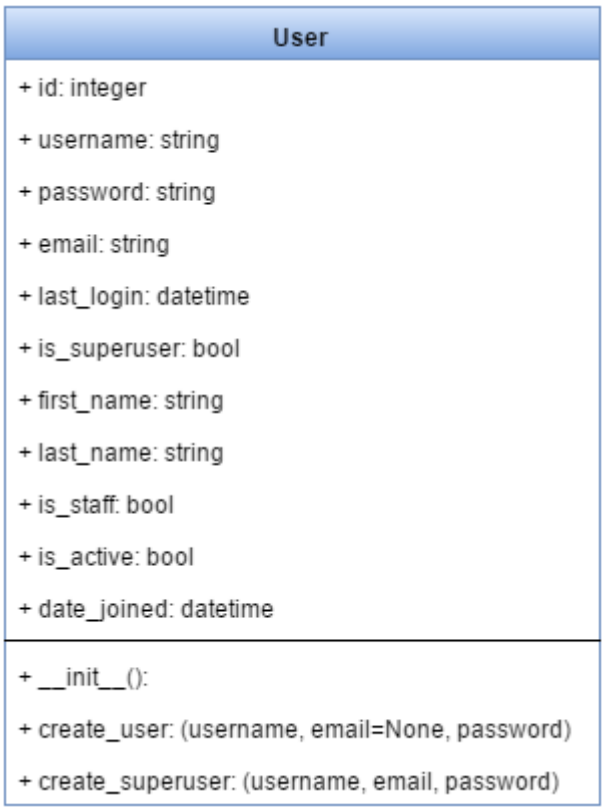
Εδώ φαίνονται οι μεταβλητές – δεδομένα του TORQUE διακομιστή καθώς και οι συναρτήσεις που χρησιμοποιούνται για την οργάνωση αυτών των δεδομένων, αλλά και για τις διαθέσιμες ενέργειες του χρήστη.

Οι μέθοδοι: create_job(), hold_rls_job(), del_job(), υλοποιούν τις αντίστοιχες συναρτήσεις της βιβλιοθήκης pbs.

Οι μέθοδοι: get_jobs(), getjob(), get_nodes(), get_node(), get_queues(), get_queue(), υλοποιούν τις αντίστοιχες συναρτήσεις της PBSQuery.

Οι μέθοδοι: job_list(), node_list(), queue_list(), αρχικοποιούν τα αντίστοιχα αντικείμενα από τα jobs_api, nodes_api και queues_api, παρέχοντάς τους τα κατάλληλα δεδομένα.

User – αναλυτικό διάγραμμα κλάσης



Το μοντέλο χρήστη, το παρέχει αυτόματα το Django από το πακέτο: *django.contrib.auth,* καθώς και τον μηχανισμό του *authentication* και του *login – logout*. Γίνεται εύκολα επέκταση του μοντέλου με περισσότερα πεδία – μεταβλητές, αλλά και μεθόδους μέσω κληρονομικότητας.

REST API

/api/login/

Methods: POST
Δέχεται σαν είσοδο τα διαπιστευτήρια του χρήστη και πραγματοποιεί είσοδο στο σύστημα ή επιστρέφει αντίστοιχο μήνυμα σφάλματος.

/api/logout/

Methods: POST
Πραγματοποιεί την έξοδο του χρήστη από το σύστημα.

/api/user/

Methods: GET, POST

POST: Επιτρέπει την δημιουργία νέου χρήστη με την προϋπόθεση να δοθούν τα απαραίτητα δεδομένα: όνομα χρήστη, κωδικός χρήστη, email.

GET: Επιστρέφει μια λίστα από τους χρήστες της εφαρμογής.

/api/user/< user id >/

Methods: GET, PUT, DELETE

GET: Επιστρέφει τα δεδομένα ενός χρήστη που έχει το αντίστοιχο αναγνωριστικό.

PUT: Κάνει ενημέρωση-αντικατάσταση των δεδομένων του χρήστη με τα νέα δεδομένα που δίνονται.

DELETE: Διαγράφει τον χρήστη με το συγκεκριμένο αναγνωριστικό.

/api/server/

Method: GET
Επιστρέφει μια λίστα των διαθέσιμων διακομιστών TORQUE.

/api/server/< server id >

Method: GET
Επιστρέφει τα δεδομένα του TORQUE διακομιστή με το αντίστοιχο αναγνωριστικό.

/api/rqalljobsperuser/

Method: GET

Επιστρέφει τον αριθμό των running, queued εργασιών, καθώς και τον αριθμό όλων των εργασιών του κάθε χρήστη.

/api/allqueues/

Method: GET
Επιστρέφει τον αριθμό των running και queued εργασιών ανά ουρά.

/api/getnofnodes/

Method: GET
Επιστρέφει τον συνολικό αριθμό κόμβων του συστήματος.

/api/getmaxcpus/

Method: GET
Επιστρέφει τον συνολικό αριθμό επεξεργαστών του συστήματος.

/api/getnodesnamesstate/

Method: GET
Επιστρέφει όλους τους κόμβους του συστήματος και την κατάσταση του καθενός.

/api/cpusummary/

Method: GET
Επιστρέφει τον αριθμό των ελεύθερων επεξεργαστών και τον αριθμό των συνολικών επεξεργαστών.

/api/jobsummary/

Method: GET
Επιστρέφει τον αριθμό των running και queued εργασιών του συστήματος.

/api/usagetotals/

Method: GET
Επιστρέφει τον αριθμό των ελεύθερων κόμβων, τον αριθμό όλων των κόμβων, τον αριθμό των ελεύθερων επεξεργαστών, τον αριθμό όλων των επεξεργαστών, τον αριθμό των running εργασιών και τον αριθμό των queued εργασιών του συστήματος.

/api/qtopnodes/

Method: GET

Επιστρέφει τα δεδομένα των κόμβων του συστήματος, που χρειάζονται στον πίνακα κατανομής εργασιών ανά κόμβο και επεξεργαστών.

4.3.2 jobs_api

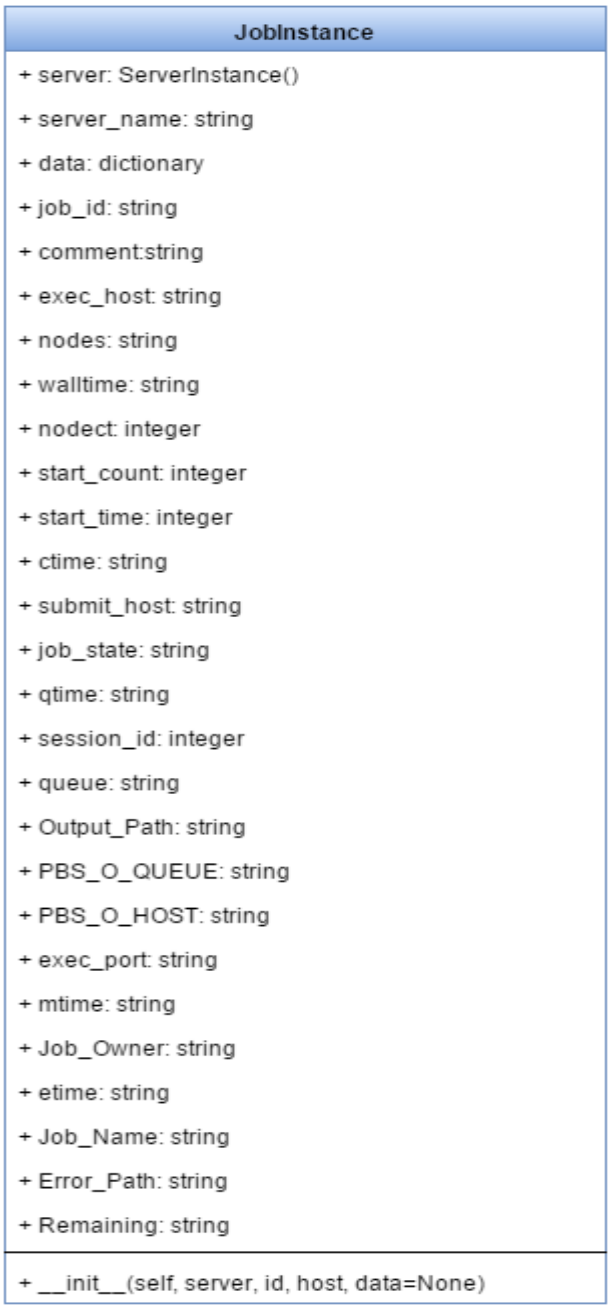
Οργάνωση

Instance.py, serializers.py, views.py, urls.py

Λειτουργικότητα

Αυτή η εφαρμογή, καλείται από το ServerInstance της εφαρμογής api, αρχικοποιείται με τα δεδομένα μιας εργασίας και τα μορφοποιεί κατάλληλα .

JobInstance – αναλυτικό διάγραμμα κλάσης



Εδώ παρουσιάζονται τα δεδομένα που επιστρέφει το σύστημα, ανά εργασία και οργανώνονται στο αντικείμενο JobInstance.

REST API

/jobs_api /job/

Methods: GET, POST

POST: Επιτρέπει την δημιουργία νέας εργασίας με την προϋπόθεση να δοθούν τα απαραίτητα δεδομένα: JobName, queue, nodes-cpus, walltime, file.

Εδώ πρέπει να σημειωθεί ότι για να ανήκει η εργασία στον χρήστη, που την υπέβαλε, πρέπει να γίνε υποβολή, κάτω από τα δικαιώματα του συγκεκριμένου χρήστη. Για αυτόν τον λόγο, δημιουργείται μια υπό-διεργασία, η οποία εκτελείται κάτω από τα δικαιώματα του συγκεκριμένου χρήστη, κάνει υποβολή την εργασία στο σύστημα με τα ίδια δικαιώματα χρήστη και τερματίζει επιστρέφοντας μήνυμα επιτυχίας ή σφάλματος.

GET: Επιστρέφει μια λίστα από τις εργασίες του συγκεκριμένου χρήστη.

/jobs_api/job/< job id >/

Methods: GET, PUT, DELETE

GET: Επιστρέφει τα δεδομένα της εργασίας με το συγκεκριμένο αναγνωριστικό.

PUT: Αλλάζει την κατάσταση της εργασίας με το συγκεκριμένο αναγνωριστικό, από Queued σε Held και από Held σε Queued ή Running. Πρέπει να σημειωθεί εδώ, ότι η αλλαγή της κατάστασης εργασίας γίνεται επιτρεπτή σε όποιον έχει τα κατάλληλα δικαιώματα. Στην συγκεκριμένη περίπτωση θέλουμε να γίνεται κάτω από τα δικαιώματα του χρήστη, οπότε δημιουργείται μια υπό-διεργασία, η οποία εκτελείται κάτω από τα δικαιώματα του συγκεκριμένου χρήστη, κάνει αλλαγή την κατάσταση της εργασίας με τα ίδια δικαιώματα χρήστη και τερματίζει επιστρέφοντας μήνυμα επιτυχίας ή σφάλματος.

DELETE: Διαγράφει την εργασία με το συγκεκριμένο αναγνωριστικό, εφόσον δεν βρίσκεται σε κατάσταση Running. Η διαγραφή μιας εργασίας στο TORQUE, γίνεται μόνον από κάποιον διαχειριστή. Για αυτό το λόγο εδώ δεν χρειάζεται να δημιουργηθεί υπό-διεργασία με τα δικαιώματα του χρήστη, αλλά παραμένουν τα δικαιώματα διαχειριστή.

/jobs_api/alljob/

Methods: GET
Επιστρέφει μια λίστα με όλες τις εργασίες του συστήματος.

/jobs_api/jobs_per_user/

Methods: GET

Επιστρέφει τον αριθμό των εργασιών ανά χρήστη.

/jobs_api/user_jobs_all_jobs/

Methods: GET

Επιστρέφει τον αριθμό των εργασιών του χρήστη και τον αριθμό των εργασιών των υπόλοιπων χρηστών.

/jobs_api/user_jobs_per_state/

Methods: GET

Επιστρέφει τον αριθμό των εργασιών του χρήστη ανά κατάσταση εργασίας: Queued, Running, Held, Completed.

/jobs_api/remaining/< job id >/

Methods: GET

Επιστρέφει τον χρόνο που απομένει για την εκτέλεση της εργασίας με το συγκεκριμένο αναγνωριστικό.

/jobs_api/cpupernode/< job id >/

Methods: GET

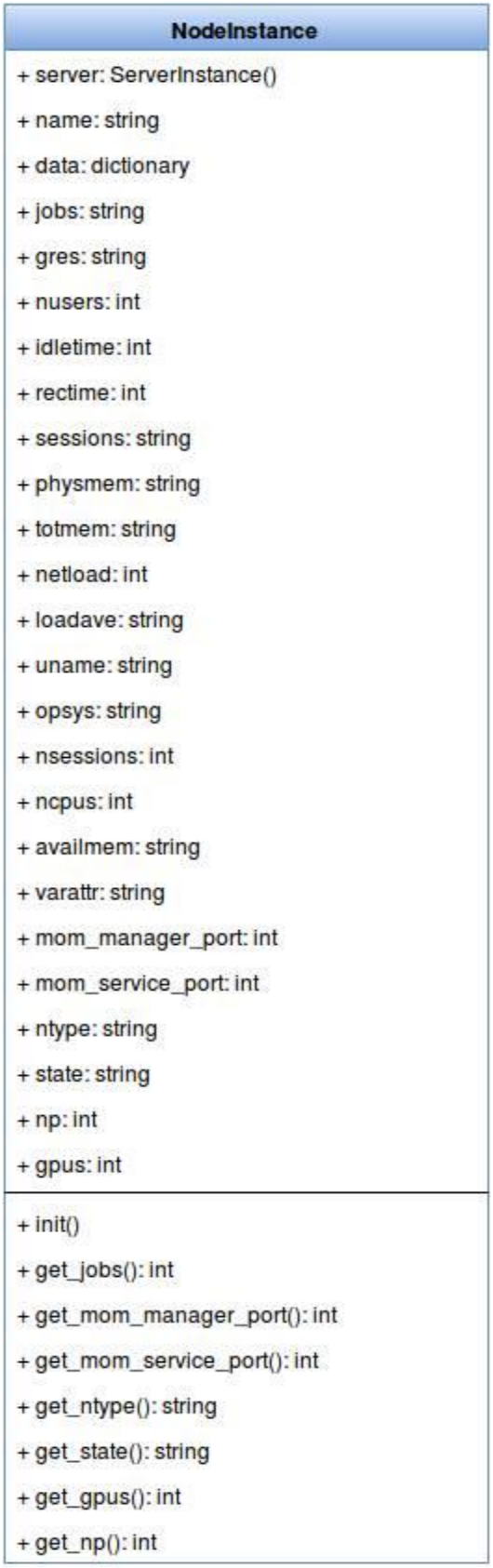
Επιστρέφει τους κόμβους που χρησιμοποιεί η εργασία με το συγκεκριμένο αναγνωριστικό, καθώς και τον αριθμό των επεξεργαστών ανά κόμβο.

4.3.3 nodes_api

Οργάνωση
Instance.py, serializers.py, views.py, urls.py

Λειτουργικότητα
Αυτή η εφαρμογή, καλείται από το ServerInstance της εφαρμογής api, αρχικοποιείται με τα δεδομένα ενός κόμβου και τα μορφοποιεί κατάλληλα .

NodeInstance – αναλυτικό διάγραμμα κλάσης



Εδώ παρουσιάζονται τα δεδομένα που επιστρέφει το σύστημα, ανά κόμβο και οργανώνονται στο αντικείμενο NodeInstance.

REST API

/nodes_api /node/

Methods: GET

Επιστρέφει μια λίστα με τα δεδομένα όλων των κόμβων του συστήματος.

/nodes_api /node/< node id >/

Methods: GET

Επιστρέφει τα δεδομένα του κόμβου με το συγκεκριμένο αναγνωριστικό.

/nodes_api /usercpuperjobspernode/< node id >/

Methods: GET

Επιστρέφει τις εργασίες που εκτελούνται στον κόμβο με το συγκεκριμένο αναγνωριστικό, καθώς και τον αριθμό των επεξεργαστών του κόμβου που χρησιμοποιεί η κάθε εργασία.

/nodes_api /npperncpupernode/< node id >/

Methods: GET

Επιστρέφει τον αριθμό των virtual επεξεργαστών και τον αριθμό των real επεξεργαστών του κόμβου με το συγκεκριμένο αναγνωριστικό.

/nodes_api /nodeperstate/

Methods: GET

Επιστρέφει τους κόμβους του συστήματος ανά την κατάσταση στην οποία βρίσκονται.

4.3.4 queues api

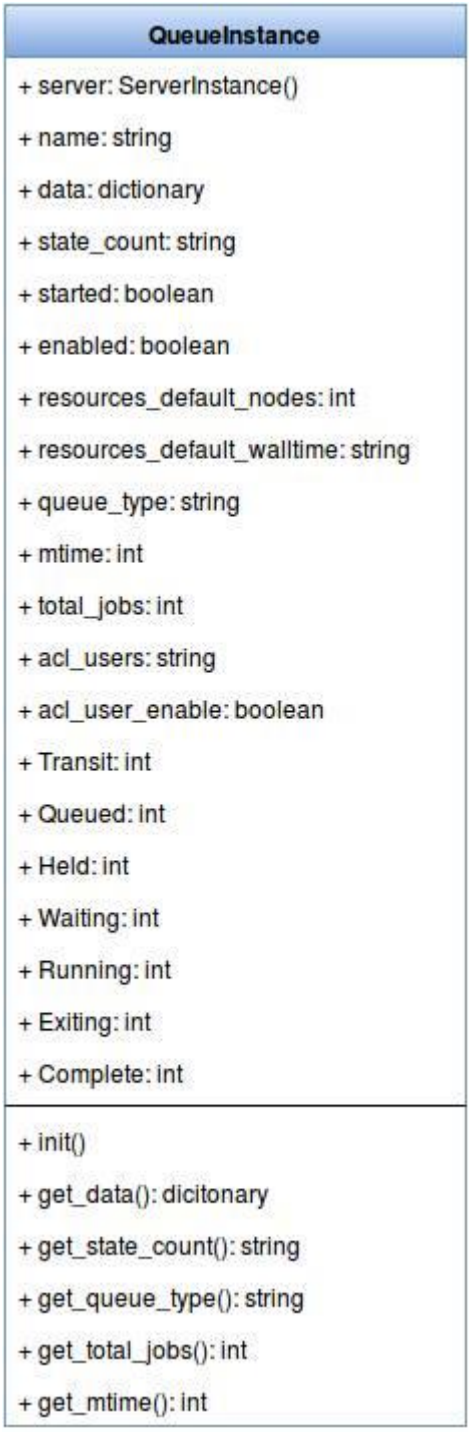
Οργάνωση

Instance.py, serializers.py, views.py, urls.py

Λειτουργικότητα:

Αυτή η εφαρμογή, καλείται από το ServerInstance της εφαρμογή api, αρχικοποιείται με τα δεδομένα μιας ουράς και τα μορφοποιεί κατάλληλα.

QueueInstance - αναλυτικό διάγραμμα κλάσης



Εδώ παρουσιάζονται τα δεδομένα που επιστρέφει το σύστημα, ανά κόμβο και οργανώνονται στο αντικείμενο NodeInstance.

REST API

/api/queues_api /queue/

Methods: GET

Επιστρέφει μια λίστα με τα δεδομένα όλων των ουρών του συστήματος.

/api/queues_api /queue/< queue id >/

Methods: GET

Επιστρέφει τα δεδομένα της ουράς με το συγκεκριμένο αναγνωριστικό.

/api/queues_api /userperalljobsperqueue/< queue id >/

Methods: GET

Επιστρέφει τους χρήστες που διατηρούν εργασίες στην ουρά με το συγκεκριμένο αναγνωριστικό, καθώς και τον αριθμό των εργασιών ανά χρήστη.

/api/queues_api /userjobsperqueueperjobstate/< queue id >/

Methods: GET

Επιστρέφει τις εργασίες του χρήστη που βρίσκονται στην ουρά με το συγκεκριμένο αναγνωριστικό, ανά κατάσταση εργασίας.

/api/queues_api /alljobsperqueueperjobstate/< queue id >/

Methods: GET

Επιστρέφει τις εργασίες όλων των χρηστών, που βρίσκονται στην ουρά με το συγκεκριμένο αναγνωριστικό, ανά κατάσταση εργασίας.

/api/queues_api /userqueue/

Methods: GET

Επιστρέφει τις ουρές στις οποίες ο χρήστης διατηρεί εργασίες, καθώς και τον αριθμό των εργασιών του χρήστη στην κάθε ουρά.

4.3.5 file_api

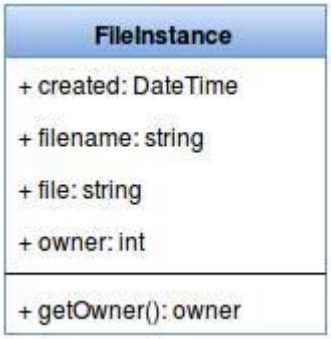
Οργάνωση

models.py, serializers.py, views.py, urls.py

Λειτουργικότητα

Αυτή η εφαρμογή συνδέεται με την βάση δεδομένων της εφαρμογής SQLite και δίνει την δυνατότητα στον χρήστη να ανεβάσει και να διαγράψει αρχεία στον αντίστοιχο κατάλογο αρχείων του, στην εφαρμογή. Ο κάθε χρήστης διατηρεί το δικό του κατάλογο αρχείων, με τα δικά του δικαιώματα, ο οποίος δημιουργείται αυτόματα την στιγμή που ο χρήστης θα πραγματοποιήσει είσοδο για πρώτη φορά στην εφαρμογή.

FileModel - αναλυτικό διάγραμμα κλάσης



Εδώ παρουσιάζεται το μοντέλο του αρχείου. Τα δεδομένα που χρειάζονται για την διαχείριση των αρχείων χρηστών στην εφαρμογή είναι το όνομα του αρχείου, το μονοπάτι στο οποίο βρίσκεται, η ημερομηνία που έγινε ανέβηκε στον διακομιστή και το αναγνωριστικό του ιδιοκτήτη του.

REST API

/file_api/file/

Methods: GET, POST

POST: Επιτρέπει την δημιουργία - ανέβασμα νέου αρχείου στον αντίστοιχο κατάλογο αρχείων του χρήστη, με την προϋπόθεση να δοθούν τα απαραίτητα δεδομένα: όνομα αρχείου, αρχείο.

GET: Επιστρέφει μια λίστα από τα αρχεία που βρίσκονται στον κατάλογο αρχείων του συγκεκριμένου χρήστη.

/file_api/file/< file id >/

Methods: GET, DELETE

GET: Επιστρέφει τα δεδομένα του αρχείου με το συγκεκριμένο αναγνωριστικό, που διατηρούνται στην βάση δεδομένων.

DELETE: Διαγράφει τα δεδομένα του αρχείου με το συγκεκριμένο αναγνωριστικό από την βάση δεδομένων και στην συνέχεια διαγράφει το αρχείο από τον αντίστοιχο κατάλογο αρχείων του χρήστη.

4.4 Είσοδος – έξοδος – δημιουργία χρηστών

Αρχικά πρέπει να επισημανθεί ότι για να μπορέσει να συνδεθεί και να χρησιμοποιήσει το TORQUE κάποιος χρήστης, θα πρέπει να διατηρεί λογαριασμό στο λειτουργικό σύστημα UNIX, στο οποίο είναι εγκατεστημένο το TORQUE.

Ένας τρόπος σύνδεσης στον διακομιστή του TORQUE, όπως έχει προαναφερθεί, γίνεται μέσω τερματικού χρησιμοποιώντας το πρωτόκολλο ssh, εφόσον βέβαια ο χρήστης διατηρεί λογαριασμό στο σύστημα.

Ο τρόπος σύνδεσης του χρήστη στον διακομιστή του TORQUE, χρησιμοποιώντας την εφαρμογή, ακολουθεί την εξής διαδικασία:

Ο χρήστης δίνει τα διαπιστευτήριά του στην εφαρμογή «όνομα χρήστη, κωδικό χρήστη». Η εφαρμογή, χρησιμοποιώντας την βιβλιοθήκη της python: PAM-python, ελέγχει αν ο χρήστης με τα συγκεκριμένα διαπιστευτήρια διατηρεί λογαριασμό στον διακομιστή. Αν όντως ο χρήστης διαθέτει λογαριασμό και πραγματοποιεί είσοδο στο σύστημα για πρώτη φορά από την εφαρμογή, τότε δημιουργείται λογαριασμός χρήστη και στην βάση δεδομένων της εφαρμογής, γεγονός το οποίο μας δίνει μεγαλύτερο έλεγχο στους λογαριασμούς χρηστών και στην συνέχεια πραγματοποιείται σύνδεση, διαφορετικά ο χρήστης λαμβάνει ένα μήνυμα το οποίο τον προτρέπει να επικοινωνήσει με κάποιον διαχειριστή του συστήματος.

4.5 Υποβολή εργασιών κάτω από τα δικαιώματα του κάθε χρήστη

Όταν ο κάθε χρήστης υποβάλει μια εργασία στο Torque μέσο terminal, το σύστημα, αναγνωρίζει τα δικαιώματα του χρήστη (uid – user id) και υποβάλει την εργασία κάτω από αυτά τα δικαιώματα. Με αυτόν τον τρόπο αναγνωρίζει το Torque ποιες εργασίες ανήκουν σε ποιους χρήστες.

Η εφαρμογή, όμως τρέχει κάτω από τα δικαιώματα του διακομιστή ιστού, οπότε είναι αδύνατο από το TORQUE να αναγνωρίσει ποιος χρήστης υποβάλλει εργασία κάθε φορά.

Για να γίνει δυνατός ο διαχωρισμός των εργασιών των χρηστών οι οποίοι τις έκαναν υποβολή στο σύστημα μέσω της εφαρμογής, ήταν να δοθούν στην εφαρμογή δικαιώματα root και όταν ο χρήστης πραγματοποιήσει μια ενέργεια, για την οποία το TORQUE χρειάζεται να γνωρίζει τα δικαιώματα του κάθε χρήστη, τότε δημιουργείται μια υπό-διεργασία, με τα αντίστοιχα δικαιώματα χρήστη, πραγματοποιεί την ενέργεια αυτή και τερματίζει.

4.6 Csrf προστασία

Το CSRF middleware και template tag παρέχουν μία εύκολη στη χρήση προστασία ενάντια σε Cross Site Request επιθέσεις. Αυτός ο τύπος επίθεσης συμβαίνει όταν κάποια κακόβουλη ιστοσελίδα περιέχει ένα σύνδεσμο, ένα κουμπί φόρμας, ή κώδικα javascript ο οποίος είναι σχεδιασμένος για να εκτελέσει κάποια ενέργεια στην ιστοσελίδα/εφαρμογή ιστού μας, χρησιμοποιώντας τα διαπιστευτήρια ενός ήδη συνδεδεμένου χρήστη, ο οποίος επισκέπτεται την κακόβουλη ιστοσελίδα μέσω του φυλλομετρητή ιστού του.

Επειδή η Extjs δεν είναι plain html, αλλά javascript, ο μηχανισμός με τον οποίο το Django διαχειριζόταν το csrf token στα Django template, δεν μπορούσε να χρησιμοποιηθεί, με αποτέλεσμα να μην επιτρέπεται η κλήση οποιασδήποτε συνάρτησης με μέθοδο POST, PUT, DELETE μέσω της Extjs. Αυτή η λειτουργικότητα ενσωματώθηκε στην εφαρμογή δημιουργώντας ένα Cookie: csrftoken και πριν από κάθε αίτημα ajax, να τοποθετεί στα ajax default headers την αντίστοιχη μεταβλητή – τιμή από το συγκεκριμένο Cookie.

ΚΕΦΑΛΑΙΟ 5 - Συμπεράσματα

5.1 Σε αυτήν την εργασία παρουσιάστηκε

Η διπλωματική αυτή εργασία είχε αντικείμενό της την δημιουργία μιας υπηρεσίας και εφαρμογής ιστού, η οποία με την χρήση του πρωτόκολλου REST θα επιτρέπει σε χρήστες να συνδέονται σε απομακρυσμένο διακομιστή, μέσω κάποιου φυλλομετρητή ιστού και να χρησιμοποιούν το TORQUE PBS, μέσα από ένα γραφικό περιβάλλον φιλικό προς το χρήστη, παρακολουθώντας την κατάσταση και τα δεδομένα του συστήματος, αλλά και διαχειρίζοντας τις εργασίες και τα αρχεία τους, χωρίς να απαιτείται η εκμάθηση εντολών κελύφους.

5.2 Μελλοντικές επεκτάσεις

Πρώτη και βασική επέκταση της εφαρμογής, είναι η δημιουργία λογαριασμού διαχειριστή του συστήματος. Οι λογαριασμοί τέτοιου τύπου θα μπορούν να διαχειρίζονται τους απλούς χρήστες – δημιουργία νέου λογαριασμού, διαγραφή λογαριασμού, εξουσιοδότηση λογαριασμών με δικαιώματα διαχειριστή, διαχείριση κόμβων – προσθήκη νέου, διαγραφή/προσαρμογή κάποιου ήδη υπάρχον κόμβου, διαχείριση ουρών – προσθήκη νέας, διαγραφή/προσαρμογή υπάρχον ουράς.

Βελτιωμένος διαχειριστής αρχείων, ο οποίος θα δίνει την δυνατότητα στους χρήστες να δημιουργούν – διαγράφουν, υπό-καταλόγους μέσα στον αρχικό τους κατάλογο αρχείων, αλλά και την δυνατότητα να βλέπει ο χρήστης τα αρχεία του και να τα επεξεργάζεται online.

Η εφαρμογή θα πρέπει να μπορεί να εντοπίζει περισσότερους από έναν TORQUE PBS διακομιστή και να δίνει την δυνατότητα στους χρήστες της να συνδεθούν με αυτόν που επιθυμούν, καθώς και να λειτουργεί – υποστηρίζει όλες τις εκδόσεις του TORQUE.

Τέλος χρήσιμη θα ήταν η προσθήκη μιας επέκτασης της εφαρμογής για κινητά τηλέφωνα η οποία τουλάχιστον στην αρχή θα επιτρέπει στους χρήστες να παρακολουθούν τα δεδομένα του διακομιστή του TORQUE, με την χρήση της SENCHA Touch ή της νεότερης Extjs 6.

ΒΙΒΛΙΟΓΡΑΦΙΑ

https://oss.trac.surfsara.nl/pbs_python

https://docs.djangoproject.com

http://www.django-rest-framework.org/

https://docs.sencha.com/extjs/5.1/application_architecture/application_architecture.html

https://en.wikipedia.org/wiki/Job_scheduler#Batch_queuing_for_HPC_clusters

https://kb.iu.edu/d/avmy

https://code.google.com/p/modwsgi/

https://httpd.apache.org/docs/

https://uwsgi-docs.readthedocs.org/en/latest/

http://nginx.org/en/

http://gunicorn.org/

https://en.wikipedia.org/wiki/Computer_cluster

https://en.wikipedia.org/wiki/Grid_computing

https://en.wikipedia.org/wiki/Portable_Batch_System

https://en.wikipedia.org/wiki/TORQUE

https://el.wikipedia.org/wiki/Python

https://en.wikipedia.org/wiki/SQLite

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

https://en.wikipedia.org/wiki/Representational_state_transfer

https://en.wikipedia.org/wiki/Application_programming_interface

https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface

https://en.wikipedia.org/wiki/Gunicorn_(HTTP_server)

https://en.wikipedia.org/wiki/Graphical_user_interface

http://sourceforge.net/projects/pbsclusterviz/

http://pbsclusterviz.sourceforge.net/

http://sourceforge.net/projects/torquemon/

http://ganglia.sourceforge.net/

http://www.google-melange.com/gsoc/org2/google/gsoc2014/ganglia

http://sourceforge.net/projects/myjam/