ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΕΙΔΙΚΑ ΘΕΜΑΤΑ ΤΕΧΝΟΛΟΓΙΑΣ ΛΟΓΙΣΜΙΚΟΥ

Τεχνική αναφορά για την εργασία υλοποίησης

The Beginners

Αναστασίου Δημήτριος - 8120006

Τριανταφύλλου Ευστράτιος - 8120133

Διδάσκων καθηγητής:

Δ. Σπινέλλης

Αθήνα, Ιούνιος 2015

Περιεχόμενα

1. Παρουσίαση του έργου 3
2. Συνεισφορά 5
3. Συνεργασία με την ομάδα ανάπτυξης 7
4. Οργάνωση στο GitHub 7
5. Κριτική στη δουλειά της ομάδας Overriders 7
6. Σύνδεσμοι 8

1. Παρουσίαση του έργου

Το diaspora* ιδρύθηκε το 2010 από τέσσερις φοιτητές του Ινστιτούτου Μαθηματικών Επιστημών Courant (Courant Institute of Mathematical Sciences), του Πανεπιστημίου της Νέας Υόρκης (New York University). Με την ονομασία diaspora* μπορεί κανείς να αναφέρεται σε δύο πράγματα: αφενός σε ένα κοινωνικό δίκτυο κατανεμημένης αρχιτεκτονικής, όπου τα δεδομένα βρίσκονται σε διασκορπισμένους εξυπηρετητές (pods) και όχι σε κάποια κεντρική βάση, αφετέρου στο λογισμικό ανοιχτού κώδικα που τρέχει σε αυτούς τους εξυπηρετητές.

Πιο συγκεκριμένα, το diaspora* μοιάζει σαν ένα πλήρως ενοποιημένο δίκτυο από επιμέρους εφαρμογές κοινωνικής δικτύωσης, που τρέχουν όλες τον ίδιο κώδικα. Κάθε εγκατάσταση του diaspora* έχει τη δική της βάση δεδομένων, τους δικούς της χρήστες και δεδομένα και το δικό της εξυπηρετητή διαδικτύου, και επικοινωνεί με τις υπόλοιπες με μια μορφή web services. Μια εγκατάσταση του diaspora* ονομάζεται pod και ο ιδιοκτήτης της podmin.

Η λογική πίσω από το ιδιόρρυθμο αυτό κοινωνικό δίκτυο είναι ότι ο χρήστης πρέπει να είναι κυρίαρχος των προσωπικών του δεδομένων. Κάποιος που θέλει να μπει στην κοινότητα του diaspora* μπορεί αρκετά εύκολα να εγκαταστήσει το ανοιχτό λογισμικό σε δικό του υπολογιστή και να ενωθεί κανονικά με τα υπόλοιπα pods του δικτύου. Ακόμη κι αν δεν θέλει να δεσμεύσει ένα μηχάνημα γι' αυτό το σκοπό, έχει τη δυνατότητα να επιλέξει το pod στο οποίο θέλει να πραγματοποιήσει εγγραφή.

Κάθε pod μπορεί να τρέχει διαφορετική έκδοση του λογισμικού, αλλά και διαφοροποιημένη διεπαφή χρήστη και τρόπο αποθήκευσης των δεδομένων στο βαθμό που δίνεται η δυνατότητα παραμετροποίησης στον podmin. Η λογική και τα πρωτόκολλα πίσω από τη σύνδεση μεταξύ των διαφορετικών pods αναφέρονται με τον όρο "federation" και αναγνωρίζουν την ασυμβατότητα που προκύπτει, αποσκοπώντας πάντα στο να είναι η επικοινωνία μεταξύ των εγκαταστάσεων ανεπαίσθητη στο χρήστη, ο οποίος αισθάνεται σαν να βρίσκεται σε ένα κανονικό (κεντρικοποιημένο) κοινωνικό δίκτυο. Για τις ανάγκες του federation, χρησιμοποιείται το πρωτόκολλο Salmon, το οποίο βασίζεται στην ανταλλαγή «μηνυμάτων» με τη μορφή xml. Όταν, για παράδειγμα, ένας χρήστης κάνει ένα σχόλιο σε ανάρτηση χρήστη από άλλο pod, τότε το σχόλιό του αποθηκεύεται τοπικά μεν, με αναφορά σε μια εγγραφή ανάρτηση σε κάποια απομακρυσμένη βάση δεδομένων, η οποία λαμβάνει με xml την απαραίτητη πληροφορία. Όταν εμφανίζεται ο αριθμός σχολίων για μια ανάρτηση, δηλαδή, ο αριθμός αυτός δεν αντιστοιχεί στο πλήθος των σχολίων που είναι διαθέσιμα μόνο τοπικά, αλλά και απομακρυσμένα, σε άλλες εγκαταστάσεις του diaspora* για τις οποίες το τρέχον pod γνωρίζει από την ανταλλαγή μηνυμάτων xml.

Το diaspora* είναι γραμμένο κυρίως σε Ruby (62.7% του κώδικα, σύμφωνα με τα στοιχεία του αποθετηρίου του στο GitHub), και συγκεκριμένα σε Ruby on Rails. Το Rails είναι ένα πλαίσιο ανάπτυξης εφαρμογών διαδικτύου, στα πρότυπα του Model-View-Controller (MVC). Το κομμάτι του Ruby on Rails αντιστοιχεί ουσιαστικά στη λειτουργικότητα του diaspora* από την πλευρά του εξυπηρετητή (server-side). Τα MVC είναι βασισμένα στην αρχή της διάκρισης της διεπαφής χρήστη από τη λογική της εφαρμογής (business logic), και της διάκρισης και των δύο από τη βάση δεδομένα, και καθιστούν την ανάπτυξη και υλοποίηση μιας εφαρμογής σε υψηλότερο επίπεδο. Τα Models αναλαμβάνουν την επικοινωνία με τη βάση δεδομένων και τη διαθεσιμότητα στην εφαρμογή των δεδομένων από αυτή, οι Controllers αναλαμβάνουν τη διαχείριση όλης της λογικής της εφαρμογής, ενώ τα Views το «στήσιμο» της διεπαφής με τον πελάτη (client). Το Rails συνεργάζεται μόνο σχεσιακές βάσεις δεδομένων, και το diaspora* δίνει τη δυνατότητα επιλογής ανάμεσα στη MySQL, τη MariaDB (εξέλιξη της MySQL) και την PostgreSQL.

Η λειτουργικότητα από την πλευρά του πελάτη (client-side) είναι γραμμένη σε γλώσσα JavaScript (20.0%) και οργανωμένη στο πλαίσιο Backbone.js, το οποίο έχει λογική παρόμοια με τη λογική των MVC. Το C (Collection) βέβαια στο Backbone δεν έχει ονοματολογική και νοηματική σχέση με το C (Controller) των Model-View-Controller. Το Collection ουσιαστικά είναι μια δομή δεδομένων που περιέχει Models. Όπως τα Models του Rails είναι υπεύθυνα για την πρόσβαση και αποθήκευση δεδομένων από και προς τη βάση, έτσι και τα Models του Backbone αποθηκεύουν και ενημερώνουν το server-side κομμάτι της εφαρμογής σε πραγματικό χρόνο. Τα Views έχουν πάλι το ρόλο της διαχείρισης του τρόπου διάθεσης και εμφάνισης του περιεχομένου στο χρήστη. Το Backbone δημιουργεί την αίσθηση συνεχούς σύνδεσης με τον εξυπηρετητή, καθώς οποιαδήποτε αλλαγή σε κάποιο μοντέλο του Rails ενημερώνει αυτόματα το αντίστοιχο μοντέλο του Backbone, ενώ ακολουθείται η λογική της αποσπασματικής ανανέωσης του περιεχομένου που προτείνει η τεχνολογία AJAX (Asynchronous JavaScript and XML). Το Backbone ουσιαστικά δίνει έναν τρόπο οργάνωσης της JavaScript, ενώ ταυτόχρονα απαλλάσσει τα pods από ένα μεγάλο κομμάτι της λειτουργικότητας που διαφορετικά θα έπρεπε να εκτελείται server-side.

Η διεπαφή χρήστη είναι γραμμένη σε Haml (server-side) και σε Handlebars.js (client-side), ενώ η μορφοποίηση σε SCSS (Sass). Αντίθετα με τις αντίστοιχες στατικές HTML και CSS, οι παραπάνω «γλώσσες» έχουν δυναμική διάσταση και δίνουν τη δυνατότητα συμπερίληψης λειτουργικού κώδικα (με μεταβλητές και μεθόδους) με Ruby, JavaScript και Sass αντίστοιχα.

2. Συνεισφορά

Στα πλαίσια της εργασίας υλοποίησης, κινηθήκαμε σε δύο άξονες συνεισφοράς: πρώτον, την εισαγωγή στο έργο της δυνατότητας υποστήριξης θεμάτων μορφοποίησης, για παράδειγμα χρωματικών θεμάτων (color themes), και την προσθήκη της δυνατότητας ανάρτησης εκδηλώσεων (events), στα οποία οι χρήστες μπορούν να δηλώσουν συμμετοχή και να ενημερώνονται για την εξέλιξή τους, παρέχοντας δυνατότητες διαχείρισης.

Ο πρώτος άξονας αφορούσε την παροχή της δυνατότητας στο χρήστη να επιλέγει από μια προκαθορισμένη λίστα θεμάτων για την προσαρμογή της εμφάνισης της διεπαφής. Η ανάγκη για τη λειτουργικότητα αυτή προήλθε από θέμα (issue) στο αποθετήριο του έργου στο GitHub (#4297), στα πλαίσια του οποίου και σχετικών συζητήσεων στο Loomio, υπήρχε συμφωνία στο να υποστηρίζει το diaspora* μόνο προκαθορισμένα χρωματικά θέματα προς το παρόν.

Ουσιαστικά, αντικείμενο της συνεισφοράς ήταν η κατασκευή και η παραμετροποίηση της υποδομής για την υποστήριξη αυτής της λειτουργικότητας. Εφόσον κάθε χρήστης θα είχε τη δυνατότητα να επιλέξει ένα χρωματικό θέμα, προστέθηκε ένα πεδίο color_theme στο μοντέλο User και ανανεώσαμε τον αντίστοιχο πίνακα στη βάση. Η λίστα των προκαθορισμένων χρωματικών θεμάτων γίνεται διαθέσιμη μέσω του αρχείου config/color_themes.yml, το οποίο ουσιαστικά περιλαμβάνει τα κωδικά ονόματα (συγχρόνως και ονόματα των αντίστοιχων φακέλων των θεμάτων) και τις ονομασίες των διαθέσιμων θεμάτων, μέσω μιας κλάσης αρχικοποίησης (initializer), στο config/initializers/color_themes.rb, η οποία διαβάζει το αρχείο και καταχωρεί τα περιεχόμενα σε αντίστοιχες μεταβλητές περιβάλλοντος, μια για τις αντιστοιχίσεις κωδικών και ονομασιών, και μια μόνο για τους κωδικούς.

Τα χρωματικά θέματα είναι ουσιαστικά αρχεία SCSS διαθέσιμα στο φάκελο app/assets/stylesheets/color_themes, τα οποία περιλαμβάνουν παρακάμψεις των μεταβλητών του Bootstrap, δηλαδή ουσιαστικά τροποποιήσεις των κανόνων μορφοποίησης της κύριας SCSS. Επειδή το Rails, σε περιβάλλον παραγωγής, μεταγλωττίζει τα SCSS στην έναρξη λειτουργίας του εξυπηρετητή, κάθε μεταγλωττισμένο αρχείο θέματος (ανά θέμα ένα για την έκδοση desktop, και ένα για την έκδοση mobile της διεπαφής) περιλαμβάνει όλη τη CSS που χρησιμοποιείται στην εφαρμογή. Οι υπόλοιπες αλλαγές στο λογισμικό έχουν σχέση με την οργάνωση των θεμάτων στο σύστημα αρχείων και τη διαχείρισή τους από την εφαρμογή.

Η τελική μορφή της συνεισφοράς στο pull request (#6033) εμφανίζεται στο GitHub ως 333 προσθήκες και 19 διαγραφές σε ένα σύνολο 38 αρχείων, που περιλαμβάνουν τα απαραίτητα σχόλια και τους απαραίτητους ελέγχους μονάδων σε RSpec. Το pull request έγινε δεκτό ύστερα από τις απαραίτητες

διορθώσεις από άποψης μορφοποίησης κώδικα, και συζήτηση πάνω στον τρόπο υλοποίησης.

Ο δεύτερος άξονας της συνεισφοράς περιλάμβανε την εισαγωγή της δυνατότητας δημιουργίας events, η ανάγκη για την οποία επίσης διατυπώθηκε σε issue στο GitHub (#1359). Η λογική της υλοποίησης για τη συνεισφορά είναι πιο εύκολα κατανοητή. Στο κομμάτι του Rails οι αλλαγές αφορούν τη δημιουργία πινάκων, μοντέλων και ελεγκτών, αλλά και την τροποποίηση των ήδη υπαρχόντων συστατικών για την ολοκλήρωση της λειτουργικότητας. Προστέθηκαν οι απαραίτητοι presenters για τη μορφοποίηση της πληροφορίας προς διαθεσιμότητα στο κομμάτι του πελάτη και έγιναν οι απαραίτητες προσθήκες και προσαρμογές στη λειτουργικότητα του πελάτη και τη διεπαφή. Στα πλαίσια της συνεισφοράς έγινε η ελάχιστη απαραίτητη τεκμηρίωση (σύμφωνα με τους κανόνες της κοινότητας) και έχουν γραφεί έλεγχοι μονάδων για τα αρχεία Ruby και μερικοί έλεγχοι ολοκλήρωσης στο πλαίσιο Cucumber.

Η συνεισφορά περιλαμβάνει ουσιαστικά τη δημιουργία events και τη δήλωση συμμετοχής σε αυτά. Η ανανέωση της συμμετοχής δεν υλοποιήθηκε, αφού απαιτεί αναπροσαρμογή της λογικής του federation, κάτι στο οποίο η κοινότητα ήδη δουλεύει, συνεπώς δεν είναι ασφαλές να προτείνουμε δική μας υλοποίηση, μιας και η αναπροσαρμογή θα γίνει μια φορά για όλες τις περιπτώσεις επικοινωνίας των pods. Το pull request δεν έχει ολοκληρωθεί ακόμη, ενώ βρίσκεται, όπως εμφανίζεται στο GitHub, στις 760 προσθήκες και 16 διαγραφές, σε σύνολο 40 αρχείων. Η προσθήκη είναι πλήρως λειτουργική, απλώς πρέπει να προστεθούν ορισμένοι έλεγχοι παραπάνω.

Το ότι η υλοποίηση της συνεισφοράς μας περιλάμβανε δουλειά σε διαφορετικά και καινούργια για εμάς πλαίσια προγραμματισμού ήταν σίγουρα πρόκληση, ωστόσο τα μεγαλύτερα θέματα με τέτοια έργα είναι η μεγάλη αλληλεξάρτηση μεταξύ πολλών διαφορετικών συστατικών (ακόμη και για μικρές αλλαγές μπορεί να χρειαστεί συνεισφορά σε πολλαπλά αρχεία) και κυρίως η ασάφεια των λειτουργικών και μη λειτουργικών απαιτήσεων.

Η συνεισφορά σε ένα τέτοιο έργο μάς εξανάγκασε σε ορισμένες περιπτώσεις να ακολουθήσουμε υπάρχουσες πρακτικές, συνήθως μακριά από τις καλύτερες που θα μπορούσαν να υλοποιηθούν (ενδεικτικά παραδείγματα η υλοποίηση στο Backbone και η παραλληλία της δεύτερης συνεισφοράς με την υπάρχουσα υλοποίηση των polls). Αυτό μπορεί να καθυστερεί τη μεταφορά του έργου σε καλύτερες πρακτικές, αλλά διευκολύνει τη συνολική προσαρμογή τους αργότερα.

3. Συνεργασία με την ομάδα ανάπτυξης

Η επικοινωνία με την ομάδα ανάπτυξης έγινε αρχικά μέσω του IRC channel του έργου. Μετά την επιλογή των συνεισφορών, διατηρήσαμε επικοινωνία μέσω IRC για γενικότερα θέματα, ενώ για πιο συγκεκριμένα ζητήματα είχαμε επαφή μέσα από τα αντίστοιχα issues και pull requests στο GitHub. Η γρήγορη ανταπόκριση τόσο στο IRC όσο και στο GitHub, βοήθησε στο να ξεπεράσουμε αρκετές από τις ασάφειες που υπήρχαν αρχικά από άποψης απαιτήσεων, επιθυμιών και προτιμήσεων της κοινότητας σχετικά με τις συνεισφορές.

4. Οργάνωση στο GitHub

Σε ότι αφορά την οργάνωση στο GitHub, δημιουργήσαμε ένα οργανισμό από τον οποίο και κάναμε fork το repository του έργου. Για κάθε μία από τις δύο συνεισφορές δημιουργήσαμε δύο διαφορετικά branches, ένα στο οποίο κάναμε commits αρκετά συχνά και ένα στο οποίο τα commits του άλλου branch γίνονταν squashed ώστε στο τέλος το pull request να μην περιλάμβανε ένα μεγάλο πλήθος commits. Και τα δύο branches είχαν ως όνομα το όνομα της συνεισφοράς, με αυτό στο οποίο τα commits γινόντουσαν συχνότερα να περιλαμβάνει και την κατάληξη test. Τα pull requests έγιναν προφανώς από το επίσημο branch (δηλαδή αυτό το οποίο δεν περιλάμβανε την κατάληξη test). Επίσης, προς το τέλος των συνεισφορών, δημιουργήσαμε ορισμένα issues, προκειμένου να μπορούμε να είμαστε μονίμως ενήμεροι για το τι απομένει να γίνει ακόμα. Στα πλαίσια της ομάδας, δουλέψαμε από κοινού, ενώ για τα commits τα οποία έγιναν, δεν ακολουθήσαμε κάποιο συγκεκριμένο μοτίβο για το ποιος θα έπρεπε να τα κάνει με αποτέλεσμα μεγάλο πλήθος αυτών να έχουν γίνει τελικά μέσα από το λογαριασμό efstrian.

5. Κριτική στη δουλειά της ομάδας Overriders

Η συγκεκριμένη ενότητα αποτελεί μια μικρή κριτική πάνω στη συνολική εργασία της ομάδας Overriders. Οι συνεισφορές των Overriders είναι ποικίλες, αφού σχετίζονται με την υλοποίηση διαφόρων ειδών αλγορίθμων που αφορούν την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης, εύρεσης συντομότερων μονοπατιών και κοινοτήτων αλλά και οπτικής αναπαράστασης γράφων. Πρόκειται, κατά συνέπεια, για μια πολυεπίπεδη εργασία που δεν ασχολείται με ένα μόνο ζήτημα και απαιτεί φυσικά την καλή κατανόηση της λειτουργίας των αλγορίθμων. Σημαντικό είναι ότι όλες οι συνεισφορές συνοδεύονται από τους κατάλληλους ελέγχους

ενώ οι μέθοδοι είναι πάντοτε τεκμηριωμένες. Αξιοσημείωτη είναι μάλιστα, η τεκμηρίωση στις μεθόδους που σχετίζονται με την επίλυση προβλημάτων η οποία είναι εκτενής και πλήρης ακολουθώντας το πρότυπο του ίδιου του έργου. Τέλος, έχει δοθεί, όπως διακρίνεται και από τις συζητήσεις των Overriders με την ομάδα ανάπτυξης, ιδιαίτερη σημασία στο θέμα της απόδοσης και συγκεκριμένα στο θέμα του χρόνου που απαιτείται για την εκτέλεση των μεθόδων, κάτι το οποίο έχει μεγάλη σημασία για το εν λόγω έργο.

6. Σύνδεσμοι

Repository της ομάδας στο GitHub:

https://github.com/ATSE-TheBeginners/diaspora

Test branches των δύο συνεισφορών:

https://github.com/diaspora/diaspora/compare/develop...ATSE-TheBeginners:4297-color-themes-test

https://github.com/diaspora/diaspora/compare/develop...ATSE-TheBeginners:1359-basic-events-test

Pull requests:

https://github.com/diaspora/diaspora/pull/6033

https://github.com/diaspora/diaspora/pull/5979