ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Δημιουργία διαδικτυακής εφαρμογής για την online στατιστική ανάλυση των δεδομένων του προγράμματος «Διαύγεια»

Του φοιτητή

Νικόλαου Ράμμου

Αρ. μητρώου: 001447

Επιβλέπων Καθηγητής

Ευστάθιος Αντωνίου

Θεσσαλονίκη 2016

Πρόλογος

Η εργασία αυτή εκπονήθηκε κατά την περίοδο 2015-2016, στα πλαίσια της φοίτησής μου στο τμήμα Μηχανικών Πληροφορικής του Α.Τ.Ε.Ι.Θ. Το θέμα της είναι η ανάπτυξη μίας διαδικτυακής εφαρμογής, και αν και πρέπει να ομολογήσω ότι δεν είχα ιδιαίτερη κλίση προς τέτοιου είδους εφαρμογές, η εξερεύνηση, η εκμάθηση και ο πειραματισμός κατά τη διάρκεια της εργασίας αποδείχτηκαν ιδιαίτερα ενδιαφέρουσες. Ελπίζω να είναι εξίσου ενδιαφέρουσα και η ανάγνωσή τους.

Θα ήθελα σε αυτό το σημείο να ευχαριστήσω την οικογένειά μου για την υποστήριξή τους όλα αυτά τα χρόνια, καθώς και τον κ. Αντωνίου για τη σημαντικότατη συμβολή του στο αποτέλεσμα της εργασίας.

Περίληψη

Στο έγγραφο αυτό παρουσιάζεται διαδικτυακή εφαρμογή που ανακτά δεδομένα από απομακρυσμένο σύστημα μέσω RESTful API. Στο πρώτο μισό του κειμένου παρουσιάζονται και αναλύονται όλες οι τεχνολογίες που χρησιμοποιήθηκαν, συγκεκριμένα τα RESTful APIs, η AngularJS, και τα Google Charts. Στο δεύτερο μισό παρουσιάζεται η χρήση αυτών των τεχνολογιών στην εφαρμογή καθώς και ο γενικότερος σχεδιασμός και η δομή της εφαρμογής.

Το κείμενο συνοδεύεται από πλήθος παραδειγμάτων για όλες τις προαναφερθείσες τεχνολογίες αλλά και από τμήματα κώδικα από την ίδια την εφαρμογή, καθώς και εικόνες από τη χρήση της.

Abstract

This document is a presentation of a web application which retrieves data from a remote system through a RESTful API. The first half of the text is comprised of the presentation and analysis of all the technologies used in the application, namely RESTful APIs, AngularJS, and Google Charts. In the second half the use of these technologies and the general design and structure of the application are presented.

The text is accompanied by many examples of the aforementioned technologies, as well as code snippets and screenshots from the application itself.

Ευρετήριο περιεχομένων

Ευρετήριο σχημάτων και πινάκων

1 ΕΙΣΑΓΩΓΗ

Σκοπός της εργασίας ήταν η ανάκτηση δεδομένων από το πρόγραμμα Διαύγεια και η παρουσίασή τους συγκεντρωτικά σε γραφική μορφή. Το πρόγραμμα Διαύγεια θεσμοθετήθηκε μέσω του Ν. 3861/2010, ο οποίος καθιστά υποχρεωτική την ανάρτηση νόμων, προεδρικών διαταγμάτων, και λοιπών πράξεων στο διαδίκτυο με σκοπό τη διαφάνεια και την ελεύθερη πρόσβαση πολιτών και οργανισμών σε δημόσια δεδομένα.

Το πρώτο ερώτημα που προκύπτει είναι πώς μπορούν να ανακτηθούν τα δεδομένα. Η Διαύγεια παρέχει ένα Application Programming Interface (API) ακριβώς για αυτόν τον σκοπό. Το API αυτό ακολουθεί την αρχιτεκτονική Representational State Transfer (REST), η οποία περιγράφει έναν τρόπο επικοινωνίας και μεταφοράς δεδομένων από και προς web υπηρεσίες. Η αρχιτεκτονική REST δεν εστιάζει στις λεπτομέρειες υλοποίησης του κάθε συστήματος, και επιτρέπει την επικοινωνία μέσω μορφών δεδομένων όπως XML και JSON, που δεν ανταποκρίνονται απαραίτητα στη μορφή με την οποία αναπαρίστανται τα δεδομένα εσωτερικά στο σύστημα.

Αφού ανακτηθούν τα δεδομένα, τίθεται το θέμα του χειρισμού τους μέσα στην εφαρμογή. Εδώ επιλέχθηκε η AngularJS, ένα JavaScript framework το οποίο εστιάζει στην ανάπτυξη εφαρμογών μίας σελίδας, εφαρμογών δηλαδή των οποίων το περιεχόμενο μπορεί να αλλάζει δυναμικά, χωρίς να ξαναφορτώνεται η σελίδα. Η AngularJS προσφέρει μεταξύ άλλων τη δέσμευση (binding) αντικειμένων της εφαρμογής με JavaScript μεταβλητές, οι τιμές των οποίων μπορούν να προέρχονται από δεδομένα μορφής JSON. Είναι λοιπόν, αν και όχι η μοναδική, μία πολύ χρήσιμη γλώσσα για επικοινωνία με REST υπηρεσίες.

Τέλος, απομένει η παρουσίαση των δεδομένων. Ο στόχος ήταν η παρουσίαση να γίνει με τη μορφή γραφικών παραστάσεων. Για τον σκοπό αυτόν, και μετά από πρόταση και του επιβλέποντα καθηγητή κ. Αντωνίου, αποφασίστηκε να χρησιμοποιηθεί το εργαλείο Google Charts. Προσφέρει πλήθος διαφορετικών γραφικών παραστάσεων οι οποίες μπορούν να είναι διαδραστικές, φορτώνονται δυναμικά, υποστηρίζουν όλους τους σύγχρονους web browser, ενώ είναι και προσαρμόσιμες σε μεγάλο βαθμό.

Στα επόμενα κεφάλαια θα αναλυθούν όλες οι παραπάνω τεχνολογίες καθώς και η εφαρμογή τους στα πλαίσια της εργασίας. Θα εξερευνηθούν οι σχέσεις μεταξύ τους και οι τρόποι διασύνδεσής τους, καθώς και σχεδιαστικά προβλήματα που έπρεπε να αντιμετωπιστούν κατά την υλοποίηση της λύσης. Τέλος, θα παρουσιαστεί το user interface της τελικής ιστοσελίδας όπου συνδυάζονται όλα αυτά τα στοιχεία.

2 ΜΟΡΦΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΗΣ ΔΙΑΥΓΕΙΑΣ

2.1 ΕΙΣΑΓΩΓΗ

Όπως αναφέρθηκε στην εισαγωγή, τα δεδομένα ανακτώνται σε μορφή JSON με κλήσεις στο RESTful API της Διαύγειας. Εδώ θα αναλύσουμε την αρχιτεκτονική REST και τη μορφή JSON, αλλά και τα δεδομένα και τη δομή των πράξεων όπως αυτές παρέχονται από το API της Διαύγειας.

2.2 REST

Η αρχιτεκτονική Representational State Transfer (REST) αναπτύχθηκε από τον Roy Fielding το 2000 στη διδακτορική του εργασία για το πανεπιστήμιο UC Irvine. Χρησιμοποιείται για ανταλλαγή δεδομένων με υπηρεσίες στο web. Συγκεκριμένα, η αρχιτεκτονική εστιάζει στις σχέσεις και αλληλεπιδράσεις μεταξύ των δεδομένων και όχι στον τρόπο αναπαράστασης των δεδομένων σε ένα σύστημα (πχ με ποια μορφή αποθηκεύονται στη βάση δεδομένων).

Οι web υπηρεσίες που συμμορφώνονται με τους περιορισμούς του REST λέγονται κοινώς RESTful APIs. Τα συστήματα αυτά επικοινωνούν συνήθως μέσω HTTP αιτημάτων, με τα ίδια HTTP ρήματα που χρησιμοποιούν και οι web browser (GET, POST, PUT, DELETE, κλπ.). Στους εξωτερικούς χρήστες οι RESTful υπηρεσίες παρουσιάζονται ως πόροι web τους οποίους μπορούν να προσπελάσουν μέσω URIs. [1]

2.2.1 Ιδιότητες

Οι ιδιότητες που επηρεάζονται από τους περιορισμούς της αρχιτεκτονικής REST είναι:

- Performance (Απόδοση) – Η απόδοση του συστήματος επηρεάζεται σε μεγάλο βαθμό από τις αλληλεπιδράσεις μεταξύ των στοιχείων του.
- Scalability (Δυνατότητα κλιμάκωσης) – Η δυνατότητα υποστήριξης μεγάλου αριθμού στοιχείων και αλληλεπιδράσεων μεταξύ τους.
- Simplicity of interfaces (Απλότητα διεπιφανειών) – Εύκολη επικοινωνία ανάμεσα στις υπηρεσίες.

- Modifiability of components (Δυνατότητα μεταβολής στοιχείων) – Γίνονται μεταβολές λόγω νέων αναγκών, ακόμα και κατά την λειτουργία.
- Visibility of communication (Ορατότητα επικοινωνιών) – Μεταξύ στοιχείων και υπηρεσιών.
- Portability of components (Φορητότητα στοιχείων) – Μεταφορά κώδικα μαζί με τα δεδομένα.
- Reliability (Αξιοπιστία) – Το σύστημα πρέπει να είναι ανθεκτικό σε αποτυχίες στοιχείων, συνδέσεων, ή δεδομένων.

[2]

2.2.2 Περιορισμοί

Μια εφαρμογή αποκαλείται RESTful εφόσον συμμορφώνεται με τους περιορισμούς του REST. Αν δε συμμορφώνεται με έναν ή περισσότερους από τους περιορισμούς, τότε δε μπορεί να θεωρηθεί RESTful. Επιγραμματικά οι περιορισμοί είναι:

- Client-server: Ένα interface διαχωρίζει client και server ώστε ο ένας να μην εξαρτάται από τις τεχνικές λεπτομέρειες του άλλου.
- Stateless: Κάθε αίτημα από τον client περιλαμβάνει όλες τις απαραίτητες πληροφορίες για να εξυπηρετηθεί από τον server, και καμία πληροφορία σχετική με την κατάσταση (state) του client δεν αποθηκεύεται στον server.
- Cacheable: Όλες οι απαντήσεις (responses) πρέπει να μπορούν αν οριστούν ως cacheable ή μη.
- Layered system: Ο client δεν έχει γνώση για το αν είναι συνδεδεμένος απευθείας με τον τελικό server ή με κάποιον ενδιάμεσο.
- Code on demand (προαιρετικό): Ο server μπορεί να τροποποιήσει προσωρινά τη λειτουργικότητα ενός client στέλνοντας εκτελέσιμο κώδικα, όπως Java applets ή JavaScript.
- Uniform interface: Ο περιορισμός αυτός είναι θεμελιώδης στη σχεδίαση οποιασδήποτε REST υπηρεσίας. Απλοποιεί και αποσυνδέει την αρχιτεκτονική, και επιτρέπει σε κάθε μέρος να εξελίσσεται ανεξάρτητα. Οι περιορισμοί αυτής της διασύνδεσης είναι:

- Identification of resources: Ξεχωριστοί πόροι αναγνωρίζονται σε αιτήματα, για παράδειγμα με URIs σε REST υπηρεσίες στο web. Οι πόροι είναι εννοιολογικά διαχωρισμένοι από την αναπαράσταση που επιστρέφεται στον client, για παράδειγμα ο server μπορεί να στείλει δεδομένα από τη βάση του σε μορφή HTML, XML ή JSON, καμία από τις οποίες δεν είναι η μορφή που χρησιμοποιεί εσωτερικά.
- Manipulation of resources through these representations: Όταν ένας client κατέχει μια αναπαράσταση ενός πόρου μαζί με τα μεταδεδομένα της απάντησης, τότε έχει αρκετές πληροφορίες ώστε να τροποποιήσει ή να διαγράψει τον πόρο.
- Self-descriptive messages: Κάθε μήνυμα περιέχει αρκετή πληροφορία ώστε να περιγράφει πως πρέπει να γίνει η επεξεργασία του μηνύματος.
- Hypermedia as the engine of application state: Οι ενέργειες που είναι διαθέσιμες στον client περιγράφονται δυναμικά σε hypermedia από τον server. Με εξαίρεση κάποιον σημείων εισόδου στην εφαρμογή, ο client δεν υποθέτει ότι κάποια ενέργεια είναι διαθέσιμη για κάποιον πόρο, πέραν αυτών που περιγράφονται σε αναπαραστάσεις που έχει λάβει προηγουμένως από τον server.

[2] [1]

2.2.3 Σε υπηρεσίες web

RESTful API που επικοινωνούν μέσω HTTP ορίζονται με τα εξής χαρακτηριστικά:

- Base URI, πχ http://example.com/resources/
- Το internet media type των δεδομένων. Συχνά είναι JSON αλλά μπορεί να είναι οποιοδήποτε άλλο είδος, όπως XML, Atom, κλπ.
- Μεθόδους HTTP (π.χ. PUT, POST, DELETE).
- Συνδέσμους hypertext για αναφορά σε κατάσταση (state).
- Συνδέσμους hypertext για αναφορά σε πόρους.

Δεν υπάρχει επίσημο standard για RESTful web APIs καθώς το REST είναι αρχιτεκτονική και όχι πρωτόκολλο. Παρ ’όλα αυτά, οι περισσότερες υλοποιήσεις χρησιμοποιούν standards όπως HTTP, URI, JSON και XML. [1]

2.3 JSON

Το format JavaScript Object Notation (JSON) είναι μια μορφή ανταλλαγής δεδομένων που εστιάζει στο να είναι εύκολη να διαβαστεί και να κατανοηθεί από ανθρώπους, αλλά και στο να αναλυθεί και να δημιουργηθεί από υπολογιστές. Είναι βασισμένο στη JavaScript αλλά δεν εξαρτάται από κάποια συγκεκριμένη γλώσσα. Τα αντικείμενα JSON είναι απλά αρχεία κειμένου, και βιβλιοθήκες για τη δημιουργία και ανάγνωσή τους είναι διαθέσιμες σε πολλές γλώσσες.

Το JSON βασίζεται σε δύο δομές:

- Μία συλλογή από ζευγάρια ονομάτων/τιμών.
- Μία ταξινομημένη λίστα από τιμές.

Πρακτικά όλες οι σύγχρονες γλώσσες προγραμματισμού υποστηρίζουν αντίστοιχες δομές δεδομένων, κάτι που κάνει το JSON ιδανική μορφή για ανταλλαγή δεδομένων ανάμεσα σε διαφορετικά συστήματα, ανεξάρτητα από τις λεπτομέρειες υλοποίησης του κάθε συστήματος.

[3]

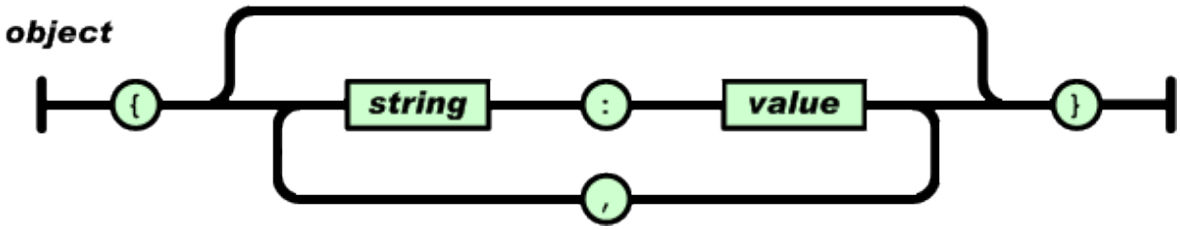
Τα σύμβολα που περιλαμβάνει η γραμματική ενός κειμένου JSON είναι:

- Τα δομικά σύμβολα [(αριστερή αγκύλη), { (αριστερό άγκιστρο),] (δεξιά αγκύλη), } (δεξί άγκιστρο), : (άνω κάτω τελεία), , (κόμμα).
- Τα κυριολεκτικά σύμβολα true, false, null.
- Strings.
- Αριθμοί.

2.3.1 Αντικείμενα

Ένα αντικείμενο αναπαρίσταται ως ένα ζευγάρι αγκίστρων το οποίο περιβάλλει κανένα ή περισσότερα ζευγάρια ονομάτων/τιμών. Ένα όνομα είναι string. Μία άνω

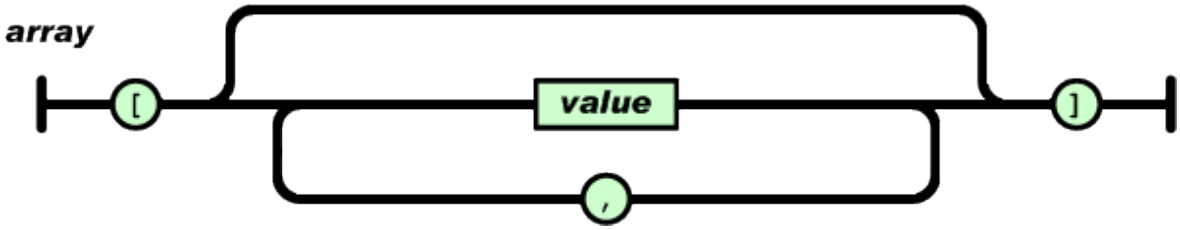
κάτω τελεία διαχωρίζει το όνομα από τη τιμή. Ένα κόμμα διαχωρίζει ένα ζευγάρι ονόματος/τιμής από ένα άλλο.



Εικόνα 1 Αντικείμενο JSON

2.3.2 Πίνακες

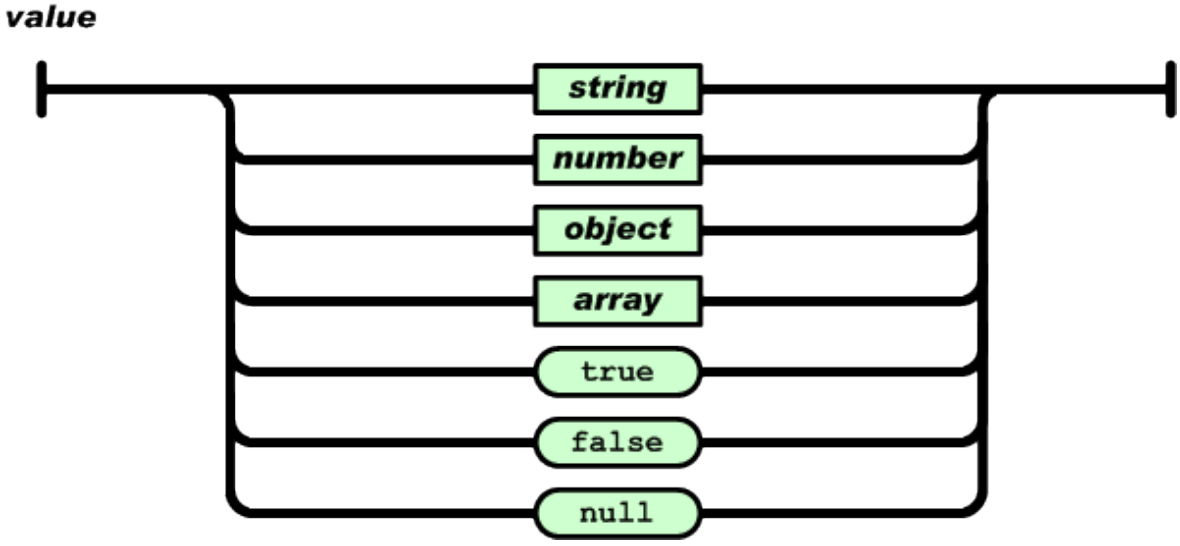
Ένας πίνακας είναι ένα ζευγάρι αγκυλών το οποίο περιβάλλει καμία ή περισσότερες τιμές. Οι τιμές διαχωρίζονται μεταξύ τους με κόμμα.



Εικόνα 2 Πίνακας JSON

2.3.3 Τιμές

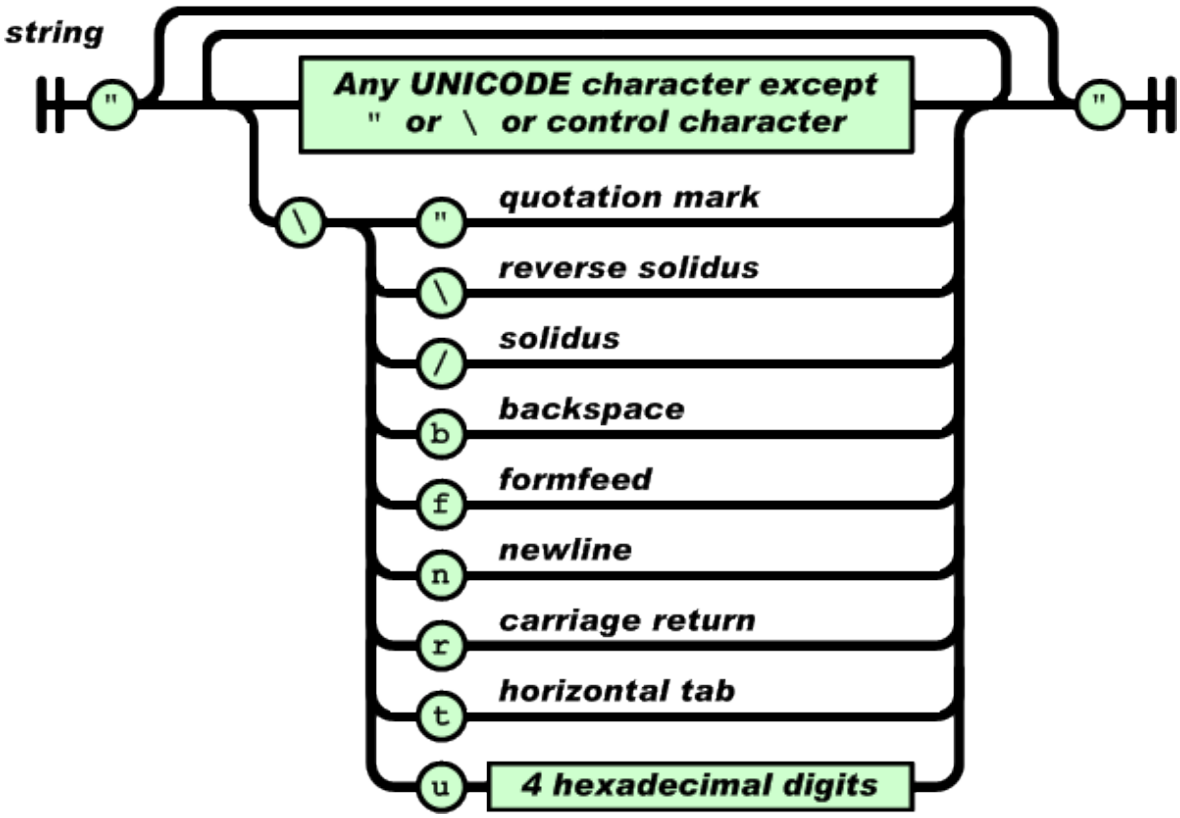
Μία τιμή μπορεί να είναι αντικείμενο, πίνακας, αριθμός, string, true, false, ή null.



Εικόνα 3 Τιμή JSON

2.3.4 Strings

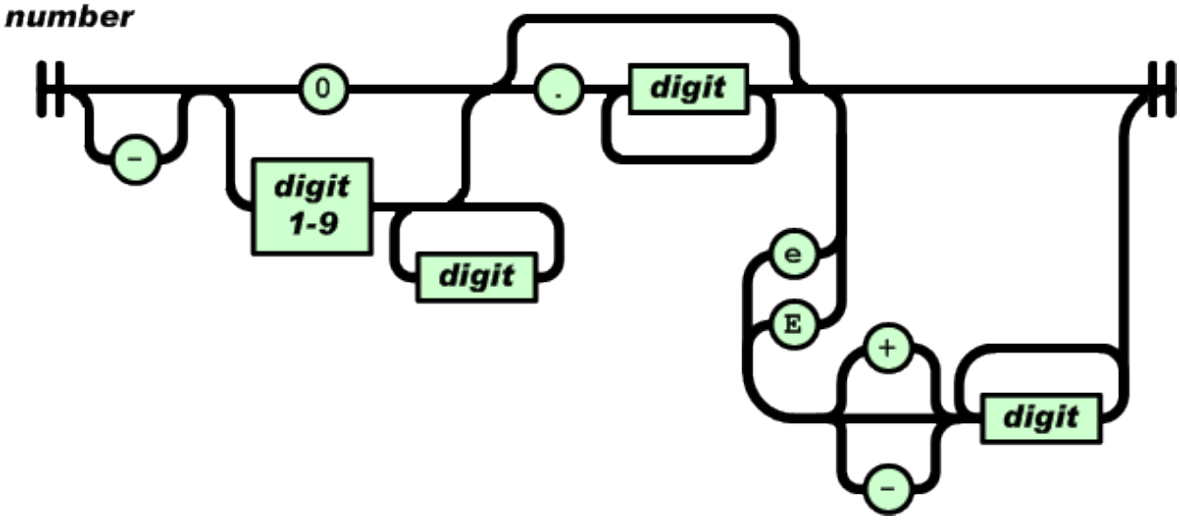
Ένα string είναι ακολουθία από Unicode χαρακτήρες που περιβάλλονται από διπλά εισαγωγικά (“) και μοιάζει πολύ με τα string της C ή της Java. Ένας χαρακτήρας αναπαρίσταται από ένα string του ενός χαρακτήρα. Το escaping γίνεται με ανάποδη κάθετο (\).



Εικόνα 4 JSON string

2.3.5 Αριθμοί

Ένας αριθμός αναπαρίσταται στο δεκαδικό σύστημα. Μπορεί να έχει αρνητικό πρόσημο, δεκαδικό μέρος του οποίου προηγείται τελεία (.), ή/και εκθέτη του δέκα του οποίου προηγείται το e ή το E και προαιρετικά το + ή το -. Αριθμητικές τιμές οι οποίες δε μπορούν να εκφραστούν ως ακολουθίες ψηφίων, όπως το άπειρο ή το NaN, δεν επιτρέπονται.



Εικόνα 5 Αριθμός JSON

[4]

2.4 Δομη δεδομενων στη Διαυγεια

Το API της Διαύγειας έχει δυνατότητα να αποστέλλει τις απαντήσεις σε αιτήματα ενός client τόσο σε JSON όσο και σε XML μορφές. Εδώ χρησιμοποιούμε JSON καθώς είναι πιο εύχρηστη, πιο διαδεδομένη, και έχει πολύ καλή υποστήριξη στην AngularJS, τη γλώσσα στην οποία είναι γραμμένη η εφαρμογή.

Το API παρέχει επίσης πρόσβαση σε μια πληθώρα πληροφοριών για κάθε πράξη που έχει αναρτηθεί. Για τους σκοπούς της εργασίας θα μας απασχολήσει ένα υποσύνολο αυτών και οι σχέσεις μεταξύ τους. Συγκεκριμένα θα χρειαστεί να ανακτήσουμε πληροφορίες για το όνομα, την ημερομηνία έκδοσης, τυχόν δαπάνες που σχετίζονται με κάθε πράξη, τον τύπο της πράξης, τον φορέα της, και τέλος την κατηγορία στην οποία ανήκει ο φορέας.

2.4.1 Πράξεις

Μία πράξη περιλαμβάνει ένα πλήθος δεδομένων που την περιγράφουν, καθώς και αναφορές σε άλλα αντικείμενα όπως φορείς ή τύπους πράξης. Μέσα στα δεδομένα που περιέχει, περιλαμβάνονται το έγγραφο της πράξης σε μορφή PDF, τυχόν συνοδευτικά έγγραφα, ο αριθμός έκδοσης, και ο Αριθμός Διαδικτυακής Ανάρτησης (ΑΔΑ).

Πέρα από τα βασικά αυτά δεδομένα τα οποία περιέχονται σε όλες τις πράξεις, κάθε πράξη περιέχει και ένα σύνολο ειδικών δεδομένων αναλόγως του τύπου της. Για

παράδειγμα, μία πράξη έγκρισης δαπάνης περιλαμβάνει δεδομένα σχετικά με τους ανάδοχους φορείς, το ποσό της δαπάνης, κλπ. Τα ειδικά δεδομένα είναι διαφορετικά για κάθε τύπο πράξης.

Η κλήση (βλ. 2.2.3 παραπάνω, εδώ παραλείπεται το base URI) για την ανάκτηση των δεδομένων μιας πράξης είναι της μορφής:

```
GET /decisions/:ada
```

Εικόνα 6 Παράδειγμα κλήσης στο API για ανάκτηση δεδομένων μιας πράξης

Όπου :ada ο Αριθμός Διαδικτυακής Ανάρτησης της πράξης.

```
{
    "ada": "ΒΛΕΖΝ-Ρ7Ν",
    "protocolNumber": "Α.200",
    "subject": "Πράξη μετάταξης",
    "issueDate": 1380585600000,
    "organizationId": "30",
    "signerIds": [ "28006" ],
    "unitIds": [ "10000" ],
    "decisionTypeId": "Γ.3.5",
    "thematicCategoryIds": [ "611" ],
    "extraFieldValues": {
        "eidosYpMetavolis": "Μετάταξη",
        "documentType": "ΠΡΑΞΗ",
        "relatedDecisions": [
            {
                "relatedDecisionsADA": "ΒΛ9ΛΝ-32Ν"
            }
        ]
    },
    "privateData": false,
    "submissionTimestamp": 1389700140183,
    "status": "PUBLISHED",
    "versionId": "6deb2804-ef9b-4640-8a4e-92f2acecb970",
    "documentChecksum": "034a544f359fe152ca646347502653c5276c786a",
    "attachments": [
        {
            "id": "c5e2ad78-6c6b-452a-8ef9-4cae2b9b691a",
            "description": "Συνοδευτικό έγγραφο",
            "filename": "attachment.pdf",
            "mimeType": "application/pdf",
            "checksum": "e357f6261bcd123bb868535ddeb58471c3265ca7"
        }
    ]
}
```


Εικόνα 7 Παράδειγμα JSON δομής πράξης

Πίνακας 1 Περιγραφή πεδίων πράξης στη Διαύγεια

Πεδίο	Περιγραφή
ada	Αριθμός Διαδικτυακής Ανάρτησης
protocolNumber	Αριθμός Πρωτοκόλλου
subject	Θέμα πράξης
issueDate	Ημερομηνία πράξης σε μορφή Unix Timestamp
decisionTypeId	Κωδικός του τύπου πράξης
organizationId	Κωδικός φορέα
unitIds	Λίστα κωδικών των μονάδων που εμπλέκονται στην έκδοση της συγκεκριμένης πράξης
signerIds	Λίστα κωδικών των τελικών υπογραφόντων της πράξης
thematicCategoryIds	Λίστα κωδικών των θεματικών κατηγοριών στις οποίες ανήκει η πράξη
privateData	Ένδειξη για το αν η πράξη περιέχει προσωπικά δεδομένα
submissionTimestamp	Ημερομηνία και ώρα τελευταίας τροποποίησης της πράξης σε μορφή Unix Timestamp
status	Ένδειξη για την κατάσταση της πράξης
versionId	Αριθμός έκδοσης, αλλάζει σε περίπτωση που τροποποιηθούν τα δεδομένα της πράξης
documentChecksum	Τιμή SHA-1 του εγγράφου της πράξης
attachments	Λίστα περιγραφών των συνημμένων εγγράφων της πράξης
extraFieldValues	Ενότητα ειδικών πεδίων, τα περιεχόμενά της εξαρτώνται από τον τύπο της πράξης
correctedVersionId	Αριθμός πρωτότυπης έκδοσης πράξης

Ο κωδικός τύπου πράξης και ο κωδικός φορέα που θα μας απασχολήσουν στη συνέχεια, αναφέρονται σε άλλους πόρους που μπορούμε να ανακτήσουμε μέσω του API, οι οποίοι περιέχουν πληροφορίες που περιγράφουν τον τύπο ή τον φορέα αντίστοιχα.

2.4.2 Τύποι πράξεων

Ο κωδικός τύπου πράξης αναφέρεται σε έναν πόρο που περιγράφει τον τύπο. Από το API μπορούμε να ανακτήσουμε είτε συνοπτική είτε λεπτομερής περιγραφή του τύπου, επειδή όμως ενδιαφερόμαστε μόνο για την ονομασία του τύπου (label), μας αρκεί η συνοπτική μορφή. Η λεπτομερής περιγραφή περιλαμβάνει και τα ειδικά δεδομένα που ορίζονται στον συγκεκριμένο τύπο πράξης.

```
GET /types/:uid
```

Εικόνα 8 Παράδειγμα κλήσης στο API για ανάκτηση δεδομένων ενός τύπου πράξης

```
{
    "uid": "2.4.6.1",
    "label": "ΠΡΑΞΕΙΣ ΧΩΡΟΤΑΞΙΚΟΥ - ΠΟΛΕΟΔΟΜΙΚΟΥ ΠΕΡΙΕΧΟΜΕΝΟΥ",
    "allowedInDecisions": true
}
```

Εικόνα 9 Παράδειγμα JSON δομής τύπου πράξης

2.4.3 Φορείς

Ο κωδικός φορέα που περιέχεται σε μία πράξη αναφέρεται σε έναν πόρο που περιέχει τα δεδομένα του συγκεκριμένου φορέα. Για τους φορείς μας ενδιαφέρει η ονομασία του φορέα (label) και η κατηγορία του (category).

```
GET /organizations/:uid
```

Εικόνα 10 Παράδειγμα κλήσης στο API για ανάκτηση δεδομένων φορέα

```
{
    "uid": "30",
    "latinName": "min-interior",
    "abbreviation": "ΥΠΕΣ",
    "label": "Υπουργείο Εσωτερικών",
    "status": "active",
    "category": "MINISTRY",
    "vatNumber": "...",
    "fekNumber": "1",
    "fekIssue": "fektype_A",
    "fekNumber": "2012",
    "website": "http://...",
    "organizationDomains": ["LocalGovernance"],
    "supervisorOrgUid": null,
    "supervisorOrgName": null,
}
```

Εικόνα 11 Παράδειγμα JSON δομής φορέα

Η κατηγορία εδώ δεν είναι η πραγματική ονομασία της κατηγορίας, αλλά ένα string το οποίο αναφέρεται σε μία τιμή λεξικού που περιέχει την ονομασία της κατηγορίας.

2.4.4 Κατηγορίες φορέων

Για την ανάκτηση της ονομασίας της κατηγορίας στην οποία ανήκει ένας φορέας, η λογική είναι ελαφρώς πιο σύνθετη. Όπως αναφέρθηκε, στα δεδομένα του φορέα περιέχεται το πεδίο category το οποίο περιέχει μία αναφορά σε μία μεταβλητή. Η μεταβλητή αυτή περιέχεται σε ένα λεξικό και περιέχει την τιμή της ονομασίας της κατηγορίας.

Αρχικά πρέπει να ανακτήσουμε τα διαθέσιμα λεξικά για να βρούμε το λεξικό που μας ενδιαφέρει (στην προκειμένη περίπτωση το λεξικό που περιγράφει τις κατηγορίες των φορέων/οργανισμών).

```
GET /dictionaries
```

Εικόνα 12 Παράδειγμα κλήσης στο API για ανάκτηση διαθέσιμων λεξικών

```
{

    "dictionaries": [
        {
            "uid": "THK",
            "label": "Τύπος Θεματικής Κατηγορίας"
        },
        {
            "uid": "CURRENCY",
            "label": "Currencies available"
        },
        {
            "uid": "FEKTYPES",
            "label": "Τύποι τευχών ΦΕΚ"
        },
        {
            "uid": "ORG_CATEGORY",
            "label": "Κατηγορίες οργανισμών"
        },
        {
            "uid": "ORG_UNIT_CATEGORY",
            "label": "Τύποι Οργανωτικών μονάδων"
        },
        {
            "uid": "ADMIN_STRUCTURE_KALLIKRATIS",
            "label": "Δομή Καλλικρατικών Περιφερειών και Δήμων"
        },
        {
            "uid": "ORG_DOMAIN",
            "label": "Πεδίο αρμοδιότητος φορέα"
        }
    ]
}
```


Εικόνα 13 Παράδειγμα JSON δομής διαθέσιμων λεξικών

Στη συνέχεια μπορούμε, βάσει του UID του λεξικού που μας ενδιαφέρει (ORG_CATEGORY στην προκειμένη περίπτωση), να το ανακτήσουμε και να αναζητήσουμε το όνομα της μεταβλητής που περιέχεται στα δεδομένα του φορέα, ώστε να πάρουμε την τιμή της, δηλαδή την ονομασία της κατηγορίας στην οποία ανήκει ο φορέας.

```
GET /dictionaries/:uid
```

Εικόνα 14 Παράδειγμα κλήσης στο API για ανάκτηση ενός λεξικού

```
{
    "name": "ORG_CATEGORY",
    "items": [
        {
            "uid": "DEYA",
            "label": "ΔΕΥΑ"
            "parent": null
        },
        {
            "uid": "PRIME_MINISTER",
            "label": "Γραφείο πρωθυπουργού"
            "parent": null
        },
        {
            "uid": "MINISTRY",
            "label": "Υπουργείο"
            "parent": null
        },
        {
            "uid": "LOCAL_AUTHORITY",
            "label": "ΟΤΑ (μόνο Δήμοι ή Αιρετές περιφέρειες)"
            "parent": null
    ]
}
```

Εικόνα 15 Παράδειγμα JSON δομής λεξικού

[5]

2.5 ΣΥΝΟΨΗ ΚΕΦΑΛΑΙΟΥ

Βλέπουμε λοιπόν ότι η επικοινωνία με ένα RESTful API είναι μία αρκετά απλή υπόθεση. Στο επόμενο κεφάλαιο θα αναλύσουμε τον τρόπο με τον οποίο μπορούμε να χειριστούμε τα δεδομένα που έχουμε ανακτήσει σε μορφή JSON μέσω της AngularJS.

3 ΧΕΙΡΙΣΜΟΣ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

3.1 ΕΙΣΑΓΩΓΗ

Εφόσον έχουμε ανακτήσει τα δεδομένα από το API, μένει να τα χειριστούμε μέσα στην εφαρμογή μας ώστε να προβάλλουμε τα δεδομένα που μας ενδιαφέρουν, στη μορφή που θέλουμε. Στο παρόν κεφάλαιο θα αναλύσουμε όχι μόνο πώς η AngularJS μας βοηθάει να μετατρέψουμε τα δεδομένα στη δομή που θέλουμε να χρησιμοποιήσουμε, αλλά και πώς μας δίνει τη δυνατότητα να ανανεώνουμε δυναμικά το περιεχόμενο της ιστοσελίδας της εφαρμογής.

3.2 ANGULARJS

Η AngularJS (επίσης Angular.js ή απλά Angular) είναι ένα web application framework βασισμένο στην JavaScript, το οποίο αναπτύχθηκε από την Google αρχικά για να διευκολύνει την ανάπτυξη web εφαρμογών μίας σελίδας. Ως εκ τούτου ο κύριος στόχος της είναι η ικανότητα να αλλάζει δυναμικά το περιεχόμενο μιας HTML σελίδας. Για να το πετύχει αυτό δημιουργεί νέες HTML δομές και ετικέτες, και ‘μαθαίνει’ στον browser νέα σύνταξη μέσω των λεγόμενων directives. Με αυτόν τον τρόπο προσφέρει:

- Model-View-Controller (MVC) ή Model-View-ViewModel (MVVM) αρχιτεκτονική. Το μοντέλο στην Angular λέγεται scope, και η ‘όψη’ (view) είναι η σελίδα HTML και τα directives.
- Two-way data binding: όταν ανανεώνονται τα δεδομένα ανανεώνεται και η όψη της εφαρμογής, και αντίστροφα.
- Dependency injection: Ο client κώδικας απλά ζητάει τις υπηρεσίες που χρειάζεται και είναι ευθύνη του injector να τις παρέχει. Ο client δε χρειάζεται να γνωρίζει τίποτα για τη δημιουργία των υπηρεσιών που καλεί.
- Cross-browser compliant: Η Angular αναλαμβάνει αυτόματα να χειρίζεται τον κώδικα JavaScript ανάλογα με τον browser στον οποίο τρέχει.
- Testability: Ο διαχωρισμός ευθυνών που προσφέρει η Angular κάνει πολύ πρακτική την επαλήθευση του κώδικα μέσω δοκιμών υπομονάδων (unit tests).

Επιγραμματικά αναφέρονται οι έννοιες που χρησιμοποιούνται στην AngularJS πολλές από τις οποίες θα συναντήσουμε στη συνέχεια, και ακολουθεί ανάλυση των πιο σημαντικών από αυτές.

Πίνακας 2 Έννοιες της AngularJS

Έννοια	Περιγραφή
Template	HTML με νέο markup που καθορίζεται από την Angular. Το επεξεργάζεται ο compiler και από αυτό δημιουργείται το view. Το νέο markup περιλαμβάνει directives και expressions.
Directives	Επεκτείνουν την HTML με νέες ιδιότητες και στοιχεία στα οποία εφαρμόζουν ειδική λειτουργικότητα.
Model	Τα δεδομένα τα οποία εμφανίζονται στον χρήστη στο view, και με τα οποία ο χρήστης αλληλοεπιδρά.
Scope	Το πλαίσιο στο οποίο αποθηκεύεται το μοντέλο ώστε τα controllers, directives, και expressions να έχουν πρόσβαση σε αυτό.
Expressions	Μικρά κομμάτια κώδικα που επιτρέπουν στην Angular να διαβάζει και να γράφει μεταβλητές στο scope. Περιέχονται σε διπλά άγκιστρα: {{ expression | filter }}
Compiler	Επεξεργάζεται το template και δημιουργεί τα directives και expressions.
Filter	Μορφοποιεί την τιμή ενός expression για εμφάνιση στον χρήστη.
View	Αυτό που βλέπει ο χρήστης (το DOM).
Data Binding	Συγχρονισμός των δεδομένων ανάμεσα στο μοντέλο και την όψη.
Controller	Το business logic πίσω από τα views.
Dependency Injection	Δημιουργεί και συνδέει αντικείμενα και μεθόδους.
Injector	Dependency injection container.
Module	Περιέχει τα μέρη μιας εφαρμογής, όπως controllers, services, directives.
Service	Επαναχρησιμοποιήσιμη business logic, ανεξάρτητη από views.

```
<div ng-app="invoice2" ng-controller="InvoiceController as invoice">
  <b>Invoice:</b>
  <div>
    Quantity: <input type="number" min="0" ng-model="invoice.qty" required>
  </div>
  <div>
    Costs: <input type="number" min="0" ng-model="invoice.cost" required>
    <select ng-model="invoice.inCurr">
      <option ng-repeat="c in invoice.currencies">{{c}}</option>
    </select>
  </div>
  <div>
    <b>Total:</b>
    <span ng-repeat="c in invoice.currencies">
      {{invoice.total(c) | currency:c}}
    </span>
    <button class="btn" ng-click="invoice.pay()">Pay</button>
  </div>
</div>
```

Εικόνα 16 Παράδειγμα HTML κώδικα με AngularJS directives και expressions

```
angular.module('invoice2', ['finance2'])
.controller('InvoiceController', ['currencyConverter', function(currency-
Converter) {
  this.qty = 1;
  this.cost = 2;
  this.inCurr = 'EUR';
  this.currencies = currencyConverter.currencies;

  this.total = function total(outCurr) {
    return currencyConverter.convert(this.qty * this.cost, this.inCurr,
outCurr);
  };
  this.pay = function pay() {
    window.alert("Thanks!");
  };
}]);
```

Εικόνα 17 Παράδειγμα κώδικα controller σε AngularJS

```
angular.module('finance2', [])
.factory('currencyConverter', function() {
  var currencies = ['USD', 'EUR', 'CNY'];
  var usdToForeignRates = {
    USD: 1,
    EUR: 0.74,
    CNY: 6.09
  };
  var convert = function (amount, inCurr, outCurr) {
    return amount * usdToForeignRates[outCurr] / usdToForeignRates[inCurr];
  };

  return {
    currencies: currencies,
    convert: convert
  };
});
```

Εικόνα 18 Παράδειγμα κώδικα service σε AngularJS

[6]

3.2.1 Model-View-Controller

Το μοντέλο περιέχει τα δεδομένα και στην Angular είναι ένα JavaScript αντικείμενο με το όνομα scope. Συνδέεται με ένα controller, και έτσι παραλαμβάνει τα δεδομένα από μία πηγή (για παράδειγμα από ένα JSON αρχείο, αλλά μπορεί να είναι και στατικά) και τα παραδίδει στο το view (και αντίστροφα).

Το view αποτελείται από HTML κώδικα, directives, και expressions. Είναι το κομμάτι της εφαρμογής το οποίο είναι ορατό στον χρήστη μέσω του browser. Ένα κομμάτι κώδικα που καθορίζει ένα view σε Angular εφαρμογή λέγεται template.

Ο controller περιέχει το business logic και ελέγχει τη συμπεριφορά της εφαρμογής.

3.2.2 Scope

Είναι αντικείμενο που περιέχεται στο μοντέλο, και μεταφέρει δεδομένα στο view. Όλες οι μεταβλητές που εμφανίζονται σε κάποιο view και δεν προέρχονται από κάποιον controller, είναι μεταβλητές που ανήκουν στο scope.

3.2.3 Two-way data binding

Η Angular μπορεί να δημιουργήσει μία σύνδεση μεταξύ ενός model και ενός view ώστε να είναι πάντα συγχρονισμένα. Στην Εικόνα 16 το directive ng-model συνδέει για παράδειγμα τη μεταβλητή invoice.cost με ένα input box. Έτσι, όταν ο χρήστης αλλάξει την τιμή στο input box θα ενημερωθεί και η τιμή της μεταβλητής του μοντέλου. Αν αλλάξει η τιμή της μεταβλητής του μοντέλου από κάποιο άλλο κομμάτι κώδικα (πχ λήψη νέων δεδομένων από κάποια υπηρεσία), τότε θα ενημερωθεί και η τιμή στο input box.

3.2.4 Templates

Ολόκληρος ο κώδικας στην Εικόνα 16 είναι ένα Angular template. Μοιάζει με συνηθισμένο HTML κώδικα με κάποια νέα στοιχεία (directives και expressions). Για να γίνει όμως δυναμικό πρέπει να προστεθούν ένας controller και ένα model.

3.2.5 Directives

Τα Angular directives είναι ιδιότητες που προσαρτώνται σε HTML στοιχεία μέσα στο view, και προσδίδουν νέες συμπεριφορές στα στοιχεία αυτά, ‘λένε’ δηλαδή στην Angular να τα χειριστεί με έναν συγκεκριμένο τρόπο. Στην Εικόνα 16 τα ng-app, ng-controller, ng-model, κλ.π. είναι directives. Για παράδειγμα, το ng-model όπως

είδαμε προηγουμένως λέει στην Angular να συνδέσει ένα HTML element με μία μεταβλητή.

3.2.6 Expressions

Περιέχονται μέσα σε διπλά άγκιστρα και συνδέουν μεταβλητές με HTML στοιχεία. Έχουν τη μορφή {{ variable | filter }}.

3.2.7 Filters

Μορφοποιούν τα δεδομένα πριν αυτά παρουσιαστούν στον χρήστη. Στην Εικόνα 16 το {{ invoice.total(c) | currency: c }} μετατρέπει την τιμή μιας αριθμητικής μεταβλητής σε μορφή τιμής νομίσματος.

3.2.8 Modules

Είναι τμήματα της εφαρμογής που εσωκλείουν μια λειτουργικότητα ή ένα αυτούσιο εννοιολογικό κομμάτι της εφαρμογής. Είναι ολοκληρωμένες οντότητες που περιέχουν τα δικά τους templates, controllers, services, κλπ. Μία εφαρμογή μπορεί να αποτελείται από ένα ή περισσότερα modules.

3.2.9 Promises

Μία λειτουργία της Angular που δεν έχει αναφερθεί έως εδώ είναι τα promises. Πρόκειται για έναν τρόπο ασύγχρονου εκτέλεσης μεθόδων μέσω της υπηρεσίας $q της AngularJS.

Αρχικά δημιουργείται ένα deferred αντικείμενο. Όταν μία ασύγχρονη μέθοδος ολοκληρωθεί, μπορεί να επιστρέψει το αντικείμενο promise που περιέχεται στο deferred αντικείμενο, είτε επιτυχημένο, είτε αποτυχημένο. Στη συνέχεια μπορεί να χρησιμοποιηθεί η μέθοδος then() του αντικειμένου promise ώστε να γίνει ο χειρισμός του αποτελέσματος της μεθόδου ανάλογα με την έκβασή της (επιτυχία/αποτυχία). Η μέθοδος then() δέχεται ως παράμετρο μία callback συνάρτηση επιτυχίας. Η συνάρτηση αυτή μπορεί να επιστρέφει με τη σειρά της ένα άλλο promise αντικείμενο του οποίου μπορεί να κληθεί η then(), και έτσι να καλούνται αλυσιδωτά μια σειρά από ασύγχρονες μεθόδους που πρέπει να τρέξουν η μία μετά την άλλη.

```
// for the purpose of this example let's assume that variables `$q` and
`okToGreet`
// are available in the current lexical scope (they could have been
injected or passed in).

function asyncGreet(name) {
  var deferred = $q.defer();

  setTimeout(function() {
    deferred.notify('About to greet ' + name + '.');

    if (okToGreet(name)) {
      deferred.resolve('Hello, ' + name + '!');
    } else {
      deferred.reject('Greeting ' + name + ' is not allowed.');
    }
  }, 1000);

  return deferred.promise;
}

var promise = asyncGreet('Robin Hood');
promise.then(function(greeting) {
  alert('Success: ' + greeting);
}, function(reason) {
  alert('Failed: ' + reason);
}, function(update) {
  alert('Got notification: ' + update);
});
```

Εικόνα 19 Παράδειγμα χρήσης υπηρεσίας $q για ασύγχρονο προγραμματισμό

[7]

Μπορούμε έτσι να βεβαιωθούμε ότι μία μέθοδος θα τρέξει μόνο αφού τελειώσει κάποια άλλη, για παράδειγμα ότι η μέθοδος που τροφοδοτεί ένα γράφημα με δεδομένα θα τρέξει μόνο αφού η μέθοδος που ανακτά τα δεδομένα από το API έχει ολοκληρωθεί και τα δεδομένα είναι διαθέσιμα.

3.3 ΔΟΜΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Η εφαρμογή ξεκινά με την αρχική σελίδα (index.html), η οποία είναι κυρίως το σημείο εισόδου της εφαρμογής. Ορίζει την κεφαλίδα της σελίδας και αναλαμβάνει να φορτώσει τις απαραίτητες βιβλιοθήκες και να ξεκινήσει το (μοναδικό) module από το app.js.

Στο app.js ορίζεται το module της εφαρμογής (με όνομα diavgeiaApp) και τα dependencies του. Εδώ ορίζεται επίσης το routing της εφαρμογής, σε ποιο template μεταβαίνει δηλαδή η εφαρμογή όταν ακολουθείται κάποιος σύνδεσμος από την κεφαλίδα, και ποιος controller είναι υπεύθυνος για το κάθε template.

Τα templates με τη σειρά τους είναι τρία:

- home.html: Είναι η αρχική σελίδα, εδώ εμφανίζονται γραφικές παραστάσεις για όλα τα είδη πράξεων, ομαδοποιημένα ανά ημέρα, κατηγορία φορέα, ή τύπο πράξης.
- financial.html: Περιέχει γραφικές παραστάσεις για πράξης του τύπου ‘Έγκριση Δαπάνης’.
- modal.html: Αναδυόμενος διάλογος όπου εμφανίζονται όλες οι πράξεις ενός σημείου μιας γραφικής παράστασης, η κάθε μία με σύνδεσμο που μεταβαίνει στη σελίδα της πράξης στη Διαύγεια.

Για κάθε ένα από τα templates υπάρχει και ο αντίστοιχος controller ο οποίος καθορίζει τη λειτουργικότητά τους: HomeController (home-controller.js), FinancialController (financial-controller.js), και ModalController (modal-controller.js).

Τέλος, υπάρχουν δύο services που παρέχουν λειτουργικότητα ανεξάρτητη από views:

- ApiService (api-service.js): Παρέχει λειτουργίες για ανάκτηση δεδομένων από το API με βάση κριτήρια αναζήτησης ή προκαθορισμένα σε μεθόδους πρότυπα (για παράδειγμα όλες τις πράξεις έγκρισης δαπάνης για έναν συγκεκριμένο φορέα).
- ChartService (chart-service.js): Περιέχει μέθοδο η οποία δημιουργεί μία γραφική παράσταση ενός δεδομένου τύπου σε ένα δεδομένο στοιχείο HTML.

3.4 ΣΥΝΟΨΗ ΚΕΦΑΛΑΙΟΥ

Σε αυτό το κεφάλαιο αναλύθηκαν συνοπτικά οι αρχές και οι βασικές έννοιες της AngularJS. Είδαμε επίσης πως μεταφράστηκαν αυτές στην αρχιτεκτονική της εφαρμογής αυτής της εργασίας. Σε επόμενα κεφάλαια θα δούμε πώς αυτά μεταφράζονται στον κώδικα που θα μορφοποιήσει τα δεδομένα όπως θέλουμε και θα τα τροφοδοτήσει στα γραφήματα της εφαρμογής. Πρώτα όμως πρέπει να γίνει μία σύντομη παρουσίαση της βιβλιοθήκης που χρησιμοποιήθηκε για τις γραφικές παραστάσεις, ώστε να υπάρχει πλήρη εικόνα του τι ακριβώς ζητάμε από την εφαρμογή.

4 ΓΡΑΦΙΚΕΣ ΠΑΡΑΣΤΑΣΕΙΣ

4.1 ΕΙΣΑΓΩΓΗ

Κατά την εκπόνηση της εργασίας αποφασίστηκε από αρκετά νωρίς ότι η έξοδος της εφαρμογής, η παρουσίαση δηλαδή των αποτελεσμάτων στον χρήστη, θα γινόταν αποκλειστικά με τη μορφή γραφικών παραστάσεων. Επιλέχθηκε τελικά η βιβλιοθήκη Google Charts που προσφέρει πλήθος διαδραστικών και δυναμικών γραφήματα, είναι εύκολη στη χρήση, με καλή υποστήριξη, και δυνατότητες παραμετροποίησης. Στις επόμενες σελίδες θα παρουσιαστεί ο τρόπος λειτουργίας τους και δυνατότητες που φάνηκαν χρήσιμες κατά την ανάπτυξη της εφαρμογής.

4.2 GOOGLE CHARTS

Η βιβλιοθήκη Google Charts είναι διάδοχος του Google Chart API που είχε αναπτύξει η Google αρχικά για εσωτερική χρήση, και αργότερα έκανε διαθέσιμο ελεύθερα σε web developers. Ενώ όμως το παλιό API δημιουργούσε και επέστρεφε γραφήματα ως PNG εικόνες με βάση παραμέτρους σε HTTP requests, η νέα βιβλιοθήκη δημιουργεί γραφήματα ως JavaScript αντικείμενα μέσα σε κώδικα JavaScript.

Παράδειγμα κλήσης δημιουργίας γραφήματος στο Google Chart API:

http://chart.apis.google.com/chart?chs=200x200&chdlp=b&chtt=Uberman&chdl=Asleep%7CAwake&chd=t:1,11,1,11,1,11,1,11,1,11,1,11&cht=p&chco=586F8E,7D858F,586F8E,7D858F,586F8E,7D858F,586F8E,7D858F,586F8E,7D858F,586F8E,7D858F

[8]

Στα Google Charts, χρειάζεται απλά η φόρτωση της βιβλιοθήκης στην εφαρμογή (μία φορά), στη συνέχεια δημιουργείται ένα JavaScript αντικείμενο που περιέχει τα δεδομένα του γραφήματος (παρόμοιο με πίνακα), ένα αντικείμενο που προαιρετικά περιέχει τις επιλογές για την παραμετροποίηση του πίνακα, και τέλος δημιουργείται το γράφημα με ένα ID ενός στοιχείου HTML. Το γράφημα θα εμφανιστεί στο στοιχείο με το ID αυτό εφόσον υπάρχει στη σελίδα.

4.2.1 Φόρτωση της βιβλιοθήκης

Για να χρησιμοποιήσουμε τη βιβλιοθήκη στην εφαρμογή, αρχικά χρειάζεται να τη φορτώσουμε. Αυτό γίνεται με μία απλή εντολή JavaScript η οποία φορτώνει τη βιβλιοθήκη και τυχόν έξτρα πακέτα που μπορεί να χρειαζόμαστε. Στη συνέχεια μπορούμε προαιρετικά να δηλώσουμε κάποια μέθοδο ως callback που θα εκτελεστεί όταν η βιβλιοθήκη φορτωθεί και είναι έτοιμη για χρήση.

Στον κώδικα που ακολουθεί, για παράδειγμα, φορτώνουμε μόνο το βασικό πακέτο (corechart). Η μέθοδος drawChart ζωγραφίζει τη γραφική παράσταση, και επειδή θέλουμε να τρέξει αφού φορτωθεί η βιβλιοθήκη (η οποία φορτώνεται ασύγχρονα), την ορίζουμε ως callback. Έτσι, όταν τελειώσει η φόρτωση της βιβλιοθήκης, αυτόματα θα κληθεί ο κώδικας που ζωγραφίζει τη γραφική παράσταση.

```
google.load('visualization', '1', {packages: ['corechart']});
google.setOnLoadCallback(drawChart);
```

Εικόνα 20 Παράδειγμα φόρτωσης της βιβλιοθήκης Google Charts

[9]

4.2.2 Προετοιμασία των δεδομένων

Για την τροφοδοσία μιας γραφικής παράστασης με δεδομένα, πρέπει τα δεδομένα να εισαχθούν σε μία κλάση JavaScript, την DataTable. Πρόκειται για έναν δισδιάστατο πίνακα, κάθε στήλη του οποίου έχει έναν τύπο δεδομένων, και προαιρετικά ένα αναγνωριστικό (ID) ή/και μία ετικέτα (label). Ο πίνακας παραμένει πλήρως επεξεργάσιμος και τα δεδομένα μπορούν να τροποποιηθούν, να διαγραφούν, ή να εισαχθούν νέα. Η μορφή του πίνακα εξαρτάται από το είδος γραφικής παράστασης στην οποία θα χρησιμοποιηθεί.

```
var data = new google.visualization.DataTable();
data.addColumn('string', 'Topping');
data.addColumn('number', 'Slices');
data.addRows([
    ['Mushrooms', 3],
    ['Onions', 1],
    ['Olives', 1],
    ['Zucchini', 1],
    ['Pepperoni', 2]
]);
```

Εικόνα 21 Παράδειγμα εισαγωγής δεδομένων σε Google Charts DataTable

[9]

4.2.3 Παραμετροποίηση του γραφήματος

Επιλογές για το γράφημα όπως το πλάτος, ο τίτλος, τα χρώματα, κ.α., μπορούν προαιρετικά να καθοριστούν σε ένα απλό JavaScript αντικείμενο ως ζεύγη ονομάτων μεταβλητών/τιμών μεταβλητών.

```
var options = {'title':'How Much Pizza I Ate Last Night',
               'width':400,
               'height':300};
```

Εικόνα 22 Παράδειγμα επιλογών γραφήματος Google Charts

[9]

4.2.4 Εμφάνιση του γραφήματος

Τέλος για την εμφάνιση του γραφήματος, πρέπει να δημιουργηθεί ένα αντικείμενο της κλάσης του είδους της γραφικής παράστασης που θέλουμε, για παράδειγμα Line Chart ή Bar Chart. Ο δομητής του αντικειμένου παίρνει ως παράμετρο μία αναφορά σε ένα HTML στοιχείο (συνήθως div), το οποίο θα περιέχει το γράφημα.

Στη συνέχεια καλείται η μέθοδος draw του αντικειμένου, η οποία δέχεται δύο παραμέτρους: το DataTable με τα δεδομένα του γραφήματος, και το προαιρετικό αντικείμενο με τις επιλογές παραμετροποίησης. Όταν καλέσουμε τη μέθοδο draw, το γράφημα θα εμφανιστεί στη σελίδα αντικαθιστώντας οτιδήποτε υπάρχει μέσα στο HTML στοιχείο που περάσαμε σαν παράμετρο στον δομητή.

```
var chart = new
google.visualization.PieChart(document.getElementById('chart_div'
));
chart.draw(data, options);

// Div that will hold the pie chart
<div id="chart_div" style="width:400; height:300"></div>
```

Εικόνα 23 Παράδειγμα εμφάνισης γραφήματος Google Charts

[9]

4.2.5 Tooltips

Τα tooltips είναι αναδυόμενα ‘κουτιά’ που εμφανίζονται σε κάποιο σημείο ενός γραφήματος. Εξ’ ορισμού παρέχονται ως SVG (VML σε Internet Explorer 8), αλλά μπορούν να δημιουργηθούν και ως HTML. Δημιουργούνται αυτόματα από τα δεδομένα των γραφημάτων, αλλά είναι και αυτά παραμετροποιήσημα σε μεγάλο βαθμό (μπορούν να περιέχουν ακόμα και άλλα γραφήματα). Η παραμετροποίηση των tooltips γίνεται με ανάθεση τιμών σε μεταβλητές μέσα στο αντικείμενο επιλογών που δίνεται ως παράμετρος κατά τη δημιουργία του γραφήματος.

```
var options = {
    tooltip: {isHtml: true},
    legend: 'none'
};
chart.draw(dataTable, options);
```

Εικόνα 24 Παράδειγμα δημιουργίας HTML tooltip σε Google Charts

[9]

Υπάρχουν πολλές άλλες επιλογές στη δημιουργία των tooltips, αλλά αυτή που θα μας απασχολήσει κυρίως εδώ είναι η δυνατότητα ανάθεσης κώδικα JavaScript ως ‘ενέργεια’ (action), δηλαδή συμπεριφορά όταν γίνεται κλικ σε ένα tooltip. Αυτό γίνεται μέσω της μεθόδου setAction του αντικειμένου του γραφήματος. Η μέθοδος αυτή δέχεται ως παράμετρο ένα JavaScript αντικείμενο το οποίο περιέχει ID, το κείμενο που θα εμφανίζεται μέσα στο tooltip και στο οποίο θα κάνει κλικ ο χρήστης για να καλέσει την ενέργεια, και ένα callback με τη λειτουργικότητα της ενέργειας.

```
chart.setAction({
    id: 'sample',
    text: 'See sample book',
    action: function() {
        selection = chart.getSelection();
        switch (selection[0].row) {
            case 0:
                alert('Ender\'s Game');
                break;
            case 1:
                alert('Feynman Lectures on Physics');
                break;
            case 2:
                alert('Numerical Recipes in JavaScript');
                break;
        }
    }
});
```

Εικόνα 25 Παράδειγμα ανάθεσης ενέργειας σε Google Chart tooltip

[9]

Οι ενέργειες φυσικά δεν χρειάζεται να είναι alerts. Μπορούν να είναι οτιδήποτε μπορεί να κάνει κανείς με JavaScript.

4.3 ΣΥΝΟΨΗ ΚΕΦΑΛΑΙΟΥ

Ακόμα και σε αυτή τη σύντομη παρουσίαση, μπορεί εύκολα να δει κανείς ότι η βιβλιοθήκη Google Charts είναι τόσο εύχρηστη, όσο και δυνατή. Στο επόμενο κεφάλαιο θα συνδυάσουμε όσα ειπώθηκαν στα τρία προηγούμενα, για να δούμε τελικά πως γίνεται η τροφοδοσία των γραφημάτων με δεδομένα από ένα RESTful API μέσω της AngularJS.

5 ΧΡΗΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

5.1 ΕΙΣΑΓΩΓΗ

Στα προηγούμενα κεφάλαια παρουσιάστηκαν τα δεδομένα στη μορφή που τα ανακτούμε από το API της Διαύγειας, ο τρόπος δημιουργίας των γραφημάτων, και κάποιοι τρόποι που προσφέρονται από την Angular ώστε να μετασχηματιστούν τα δεδομένα στη μορφή που θέλουμε. Σε αυτό το κεφάλαιο θα γίνει η παρουσίαση της λειτουργικότητας αυτής, δείχνοντας των κώδικα της εφαρμογής που την αναλαμβάνει.

Συγκεκριμένα θα δούμε πώς τα δεδομένα, αφού ανακτηθούν, μετατρέπονται και εισάγονται στο κάθε γράφημα (η εφαρμογή περιέχει πέντε διαφορετικά γραφήματα: αναρτήσεις ανά ημέρα/κατηγορία φορέα/είδος πράξης, και ανά φορέα/ανά ημέρα για πράξεις έγκρισης δαπάνης), καθώς και πώς ρυθμίζονται οι επιλογές του κάθε γραφήματος.

5.2 ΦΟΡΤΩΣΗ ΔΕΔΟΜΕΝΩΝ

Όταν καλείται η μέθοδος φόρτωσης των δεδομένων στον HomeController όπου φορτώνονται τα γραφήματα αναρτήσεων ανά ημέρα, κατηγορία φορέα, ή τύπο πράξης, αρχικά ανακτώνται από το API όλες οι αναρτήσεις στο εύρος ημερομηνιών που έχει ορίσει ο χρήστης. Στη συνέχεια φορτώνονται τα δεδομένα όλων των γραφημάτων, και τέλος καλείται για κάθε γράφημα η μέθοδος που εισάγει τα δεδομένα σε αυτό (το γράφημα δεν εμφανίζεται όμως, παρά μόνο εάν είναι ορατή η σελίδα του γραφήματος εκείνη τη στιγμή).

```
ApiService.getDecisionsInRange($scope.dateFrom, $scope.dateTo)
    .then(function(response) {
        $scope.data = response.data;
        loadDecisionsPerDay();
        ApiService.getOrgCategoryLabels()
        .then(function(orgCategoryLabels) {
            $scope.categoryLabels = orgCategoryLabels;
            loadDecisionsPerOrgCategory();
        })
        .then(function() {
            return ApiService.getDecisionTypes();
        })
        .then(function(decisionData) {
            $scope.decisionTypes = decisionData.data.decisionTypes;
            loadDecisionsPerType();
        })
        .catch(function(error) {
            alert('Error: ' + error);
        });
    })
    .catch(function(error) {
        alert('Error: ' + error);
    })
    .finally(function() {
        // Reactivate the page controls.
        $scope.status.loading = false;
    });
```

Εικόνα 26 Φόρτωση δεδομένων της εφαρμογής

Στον FinancialController ο οποίος χειρίζεται τα δεδομένα των οικονομικών γραφικών παραστάσεων, εάν δεν έχει επιλεχθεί κάποιος φορέας ή ανάδοχος τότε ανακτώνται και φορτώνονται όλα τα οικονομικά δεδομένα ανά φορέα για το επιλεγμένο εύρος ημερομηνιών, αλλιώς ανακτώνται τα δεδομένα ανά ημέρα μόνο για τον επιλεγμένο φορέα/ανάδοχο (ή και για συνδυασμό τους).

```
if(!$scope.selectedOrganization && !$scope.selectedSponsor) {
    ApiService.getFinancialData($scope.dateFrom, $scope.dateTo)
    .then(function(response) {
        loadFinancialData(response);
    }).catch(function(error) {
        alert('Error:' + error);
    }).finally(function() {
        $scope.status.loading = false;
    });
}
else {
    ApiService.getFinancialDataByEntity($scope.dateFrom, $scope.dateTo,
        $scope.selectedOrganization, $scope.selectedSponsor)
    .then(function(response) {
        loadFinancialDataByEntity(response);
    }).catch(function(error) {
        alert('Error: ' + error);
    }).finally(function() {
        $scope.status.loading = false;
    });
}
```

Εικόνα 27 Φόρτωση οικονομικών δεδομένων της εφαρμογής

5.2.1 Αναρτήσεις ανά ημέρα

Κατά τη φόρτωση του γραφήματος αναρτήσεων ανά ημέρα, αρχικά δημιουργείται το αντικείμενο DataTable το οποίο θα κρατήσει τα δεδομένα του γραφήματος.

Στη συνέχεια δημιουργείται ένας πίνακας με όλες τις ημερομηνίες που υπάρχουν στο διάστημα που έχει επιλέξει ο χρήστης. Αυτό γίνεται ώστε να εμφανιστούν όλες οι ημερομηνίες στο γράφημα και όχι μόνο αυτές στις οποίες έχουν γίνει αναρτήσεις, καθώς στα δεδομένα της Διαύγειας υπάρχουν μόνο οι τελευταίες.

Έπειτα, τα δεδομένα ταξινομούνται κατά ημερομηνία, και σειριακά για κάθε ημερομηνία στο επιλεγμένο εύρος υπολογίζεται το άθροισμα των αναρτήσεων που της αντιστοιχούν. Κάθε ζεύγος ημερομηνίας-αθροίσματος εισάγεται στα δεδομένα του γραφήματος.

Τέλος, ρυθμίζονται οι επιλογές εμφάνισης του γραφήματος (τίτλος, υπόμνημα, μορφή τιμών κάθετου άξονα, tooltips), και εάν η σελίδα των αναρτήσεων ανά ημέρα είναι ορατή, ζωγραφίζεται το γράφημα στη σελίδα.

```
var loadDecisionsPerDay = function() {
    $scope.charts.day.data = new google.visualization.DataTable();
    $scope.charts.day.data.addColumn('string', 'Ημέρα');
    $scope.charts.day.data.addColumn('number', 'Αναρτήσεις');

    var dates = [];
    var d = new Date();
    for(d.setTime($scope.dateFrom.getTime());
            d.getTime() <= $scope.dateTo.getTime();
            d.setDate(d.getDate() + 1)) {
        dates.push($filter('date')(d));
    }

    var orderedData = $filter('orderBy')($scope.data.decisions,
        'issueDate');

    for(var i = 0; i < dates.length; i++) {
        var sum = 0;
        for(var j = 0; j < orderedData.length; j++) {
            if($filter('date')(orderedData[j].issueDate) ===
                    dates[i]) {
                sum++;
            }
        }
        $scope.charts.day.data.addRow([dates[i], sum]);
    }

    $scope.charts.day.options = {
        title: 'Αριθμός αναρτήσεων ανά ημέρα ' + dates[0] + ' - ' +
            dates[dates.length - 1],
        legend: 'none',
        vAxis: {format : 'decimal'},
        tooltip: {trigger: 'selection', isHtml: true}
    };

    if($scope.status.dayTabIsActive) {
        $scope.drawDayChart();
    }
};
```

Εικόνα 28 Φόρτωση αναρτήσεων ανά ημέρα

5.2.2 Αναρτήσεις ανά κατηγορία φορέα

Ο χειρισμός των δεδομένων για τις αναρτήσεις ανά κατηγορία φορέα είναι πολύ παρόμοιος με τις αναρτήσεις ανά ημέρα. Το κύριο που αλλάζει, όπως είναι αναμενόμενο, είναι τα δεδομένα που εισάγονται στο γράφημα. Εδώ δημιουργείται αρχικά ένας πίνακας στον οποίο εισάγεται η κατηγορία φορέα για κάθε ανάρτηση

που υπάρχει στα αποτελέσματα. Στη συνέχεια δημιουργείται ένας νέος πίνακας όπου κάθε κατηγορία φορέα είναι μοναδική, δεν υπάρχουν δηλαδή διπλότυπες σειρές. Τέλος, για κάθε μοναδική κατηγορία φορέα από τον δεύτερο πίνακα, εισάγεται στα δεδομένα του γραφήματος ο αριθμός των εμφανίσεών της στον πρώτο πίνακα.

```
var loadDecisionsPerOrgCategory = function() {
    $scope.charts.org.data = new google.visualization.DataTable();
    $scope.charts.org.data.addColumn('string', 'Κατηγορία');
    $scope.charts.org.data.addColumn('number', 'Αναρτήσεις');

    var labelsCount = [];
    for(var i = 0; i < $scope.data.decisions.length; i++) {
        for(var j = 0; j < $scope.categoryLabels.length; j++) {
            if($scope.data.decisions[i].organizationId ===
                    $scope.categoryLabels[j].orgId) {
                labelsCount.push({
                    label: $scope.categoryLabels[j].categoryLabel
                });
            }
        }
    }

    var uniqueLabels = $filter('unique')(labelsCount, 'label');
    uniqueLabels = $filter('orderBy')(uniqueLabels, 'label');
    for(var i = 0; i < uniqueLabels.length; i++) {
        var count = $filter('filter')
            (labelsCount, uniqueLabels[i].label).length;
        $scope.charts.org.data.addRow([uniqueLabels[i].label, count]);
    }

    $scope.charts.org.options = {
        title: 'Αριθμός αναρτήσεων ανά κατηγορία φορέα ' +
            $filter('date')($scope.dateFrom) + ' - ' +
            $filter('date')($scope.dateTo),
        pieHole: 0.4,
        tooltip: {trigger: 'selection', isHtml: true}
    };

    if($scope.charts.org.data.getNumberOfRows() === 1) {
        $scope.charts.org.options.pieSliceTextStyle = {color: 'black'};
    }

    if($scope.status.orgTabIsActive) {
        $scope.drawOrgChart();
    }
};
```

Εικόνα 29 Φόρτωση αναρτήσεων ανά κατηγορία φορέα

5.2.3 Αναρτήσεις ανά τύπο πράξης

Όπως και για τις προηγούμενες γραφικές παραστάσεις, έτσι και εδώ η λειτουργικότητα είναι σχεδόν ίδια. Για κάθε ονομασία είδους πράξης (τις οποίες τις έχουμε ανακτήσει με ξεχωριστή κλήση στο API), εισάγεται στα δεδομένα του γραφήματος ο αριθμός των αναρτήσεων των οποίων το ID είδους πράξης αντιστοιχεί στο ID της ονομασίας.

```
var loadDecisionsPerType = function() {
    $scope.charts.type.data = new google.visualization.DataTable();
    $scope.charts.type.data.addColumn('string', 'Είδος πράξης');
    $scope.charts.type.data.addColumn('number', 'Αναρτήσεις');

    for(var i = 0; i < $scope.decisionTypes.length; i++) {
        var sum = 0;
        for(var j = 0; j < $scope.data.decisions.length; j++) {
            if($scope.data.decisions[j].decisionTypeId ===
                    $scope.decisionTypes[i].uid) {
                sum++;
            }
        }
        if(sum > 0) {
            $scope.charts.type.data.addRow(
                    [$scope.decisionTypes[i].label, sum]);
        }
    }

    $scope.charts.type.options = {
        title: 'Αριθμός αναρτήσεων ανά είδος πράξης ' +
            $filter('date')($scope.dateFrom) + ' - ' +
            $filter('date')($scope.dateTo),
        legend: 'none',
        tooltip: {trigger: 'selection', isHtml: true}
    };

    if($scope.status.typeTabIsActive) {
        $scope.drawTypeChart();
    }
};
```

Εικόνα 30 Φόρτωση αναρτήσεων ανά τύπο πράξης

5.3 ΕΜΦΑΝΙΣΗ ΓΡΑΦΙΚΩΝ ΠΑΡΑΣΤΑΣΕΩΝ

5.3.1 Ασύγχρονη δημιουργία

Στο τέλος κάθε μίας από τις παραπάνω μεθόδους καλείται η μέθοδος που εμφανίζει το γράφημα και δημιουργεί το tooltip action, εάν η εφαρμογή βρίσκεται στην αντίστοιχη σελίδα. Επειδή πρέπει πρώτα να φορτώσει η υπόλοιπη σελίδα προτού μπορέσει να εμφανιστεί το γράφημα, αρχικά καλείται μία μέθοδος η οποία περιμένει 100 millisecond, και στη συνέχεια εμφανίζει τη γραφική παράσταση με μία κλήση στο ChartService (βλ. 5.3.3 παρακάτω). Επειδή με τη σειρά του το tooltip action μπορεί να προσαρτηθεί στο γράφημα μόνο αφού το γράφημα έχει δημιουργηθεί, και επειδή η μέθοδος που εμφανίζει το γράφημα είναι ασύγχρονη, η μέθοδος αυτή επιστρέφει ένα αντικείμενο promise με το γράφημα.

```
var drawChartDelayed = function(data, options, chartType, element) {
    var def = $q.defer();

    if(data) {
        $timeout(function() {
            def.resolve(ChartService.drawChart(data, options,
                chartType, element));
        }, 100);
    }
    else {
        def.reject();
    }

    return def.promise;
};
```

Εικόνα 31 Ασύγχρονη εφάνιση γραφικής παράστασης με καθυστέρηση

5.3.2 Tooltip actions

Αφού ολοκληρωθεί η δημιουργία και εμφάνιση της γραφικής παράστασης, δημιουργείται το tooltip action. Εδώ πρέπει να οριστεί ένα ID, το κείμενο που θα εμφανίζεται ως επιλογή και το action το ίδιο. Αρχικά παίρνουμε το επιλεγμένο σημείο της γραφικής παράστασης και την τιμή της παράστασης στο σημείο αυτό. Αν η τιμή είναι μεγαλύτερη του μηδενός (αν δηλαδή υπάρχουν δεδομένα για εκείνο το σημείο, ή με άλλα λόγια αν υπάρχουν αναρτήσεις), τότε εμφανίζεται ένας modal popup διάλογος με τις αναρτήσεις που αντιστοιχούν σε εκείνο το σημείο της γραφικής παράστασης. Στη συνέχεια ορίζονται ο τίτλος του διαλόγου και τα δεδομένα του τα οποία είναι ένας πίνακας με αναρτήσεις.

Παίρνοντας ως παράδειγμα το γράφημα αναρτήσεων ανά ημέρα, για κάθε ανάρτηση που υπάρχει στα αποτελέσματα που έχουμε ανακτήσει από το API για το επιλεγμένο εύρος ημερομηνιών, εάν η ημερομηνία ανάρτησής της είναι ίδια με την ημερομηνία στο επιλεγμένο σημείο της γραφικής παράστασης, τότε η ανάρτηση αυτή προστίθεται στον πίνακα του διαλόγου.

```
$scope.drawDayChart = function() {
  drawChartDelayed($scope.charts.day.data, $scope.charts.day.options,
    Constants.ChartTypes['LINE_CHART'],
    document.getElementById('chart_div_day'))
  .then(function(chart) {
    chart.setAction({
      id: 'modal',
      text: 'Αναρτήσεις',
      action: function() {
        var selection = chart.getSelection();
        var row = selection[0].row;
        var no = $scope.charts.day.data.getValue(row, 1);

        if(no > 0) {
          $uibModal.open({
            animation: true,
            size: 'lg',
            templateUrl: 'views/modal.html',
            controller: 'ModalController',
            controllerAs: 'modal',
            resolve: {
              title: function() {
                return $scope.charts.day.data.getValue(row, 0);
              },
              data: function() {
                var modalData = [];
                  for(var i = 0; i < $scope.data.decisions.length;
                      i++) {
                    if($scope.charts.day.data.getValue(row, 0) ===
                        $filter('date')
                        ($scope.data.decisions[i].issueDate)) {
                      modalData.push($scope.data.decisions[i]);
                    }
                  }
                  return modalData;
                }
              }
            });
          }
        }
      });
    });
};
```

Εικόνα 32 Δημιουργία του γραφήματος αναρτήσεων ανά ημέρα

5.3.3 Chart Service

Είδαμε πως η δημιουργία μίας γραφικής παράστασης γίνεται σε ξεχωριστό σημείο, σε μία υπηρεσία που ευθύνη της είναι αποκλειστικά η δημιουργία των γραφημάτων.

Η υπηρεσία περιέχει μία μέθοδο η οποία ανάλογα με το είδος του γραφήματος που έχει περαστεί ως παράμετρος δημιουργεί το αντίστοιχο αντικείμενο. Στη συνέχεια καλεί τη μέθοδο draw, η οποία εμφανίζει το γράφημα στο καθορισμένο στοιχείο HTML, και τέλος ορίζει έναν handler ο οποίος αναλαμβάνει να ζωγραφίζει ξανά τη γραφική παράσταση όταν αλλάζει το μέγεθος του browser.

```
app.service('ChartService', ['Constants', function(Constants) {

    this.drawChart = function(chartData, options, chartType, element) {
        var chart;
        switch(chartType) {
            case Constants.ChartTypes['LINE_CHART']:
                chart = new google.visualization.LineChart(element);
                break;
            case Constants.ChartTypes['PIE_CHART']:
                chart = new google.visualization.PieChart(element);
                break;
            case Constants.ChartTypes['COLUMN_CHART']:
                chart = new google.visualization.ColumnChart(element);
                break;
            case Constants.ChartTypes['BAR_CHART']:
                chart = new google.visualization.BarChart(element);
                break;
            case Constants.ChartTypes['COMBO_CHART']:
                chart = new google.visualization.ComboChart(element);
                break;
            default: // Not used
        }

        chart.draw(chartData, options);

        // Resizes the chart when the browser window is resized.
        function resizeHandler() {
            chart.draw(chartData, options);
        }
        if(window.addEventListener) {
            window.addEventListener('resize', resizeHandler, false);
        }
        else if(window.attachEvent) {
            window.attachEvent('onresize', resizeHandler);
        }

        return chart;
    };
}]);
```

Εικόνα 33 Service δημιουργίας γραφημάτων

5.4 MODAL ΔΙΑΛΟΓΟΣ

Το tooltip κάθε γραφήματος περιέχει και ένα action, προσφέρει δηλαδή στον χρήστη τη δυνατότητα να εμφανίσει έναν διάλογο με τις αναρτήσεις που αντιστοιχούν στο σημείο της γραφικής παράστασης που έχει επιλεχθεί. Για παράδειγμα, σε ένα σημείο της γραφικής παράστασης αναρτήσεων ανά ημέρα, μπορούμε να εμφανίσουμε έναν διάλογο που περιέχει όλες τις αναρτήσεις που έγιναν εκείνη την ημέρα.

Ο τρόπος εμφάνισης των αναρτήσεων γίνεται στο HTML κομμάτι του διαλόγου (βλ. 7.4.5 παρακάτω). Ο controller του διαλόγου περιλαμβάνει δύο απλές μεθόδους. Η μία επιτρέπει στον χρήστη να επιλέξει μία ανάρτηση από τον διάλογο, οπότε και ανοίγει τη σελίδα της πράξης στη Διαύγεια σε νέα καρτέλα του browser. Η άλλη απλά κλείνει τον διάλογο όταν ο χρήστης επιλέξει το κουμπί ΄κλείσιμο'.

```
app.controller('ModalController',
        ['$scope', '$uibModalInstance', 'title', 'data',
        function($scope, $uibModalInstance, title, data) {

    this.title = title;
    this.data = data;

    $scope.openDecisionInNewTab = function(ada) {
        window.open('https://diavgeia.gov.gr/decision/view/' + ada,
            '_blank');
    };

    $scope.close = function() {
        $uibModalInstance.close();
    };
}]);
```

Εικόνα 34 Controller modal διαλόγου

5.5 ΣΥΝΟΨΗ ΚΕΦΑΛΑΙΟΥ

Παρουσιάστηκε σε αυτό το κεφάλαιο το τι συμβαίνει με τα δεδομένα αφού ανακτηθούν και τη διαδικασία δημιουργίας των γραφημάτων. Στο επόμενο κεφάλαιο θα γίνει παρουσίαση του τρόπου αλληλεπίδρασης με το API, και τις κλήσεις που γίνονται.

6 ΑΝΑΚΤΗΣΗ ΔΕΔΟΜΕΝΩΝ

6.1 ΕΙΣΑΓΩΓΗ

Στη συνέχεια θα δούμε πως ακριβώς γίνονται οι κλήσεις στο API. Συγκεκριμένα, όλες οι αλληλεπιδράσεις με το API της Διαύγειας γίνονται μέσω ενός PHP client που τρέχει στον server του τμήματος ‘Αετός’. Εδώ αποκαλούμε αυτό το τμήμα ως client διότι, αν και είναι η server side πλευρά της εφαρμογής, είναι client του API της Διαύγειας.

Ο client είναι βασισμένος σε παράδειγμα που προσφέρει η Διαύγεια [10], και θα αναλυθεί στη συνέχεια. Η γενική ροή είναι ότι το API service της εφαρμογής αρχικά κάνει μία κλήση στον client, ο οποίος με τη σειρά του καλεί το API της Διαύγειας και επιστρέφει την απόκριση (τα δεδομένα) πίσω στην εφαρμογή υπό μορφή JSON αντικειμένου.

6.2 API SERVICE

Όλες οι κλήσεις του API service βασίζονται στη μέθοδο getRequest. Η μέθοδος αυτή παίρνει σαν παράμετρο ένα ερώτημα (query) προς το API της Διαύγειας σε μορφή string. Το ερώτημα αυτό περιγράφει ποια δεδομένα μας ενδιαφέρουν. Στη συνέχεια η μέθοδος στέλνει με HTTP request το ερώτημα στον PHP client ο οποίος θα το προωθήσει στο API της Διαύγειας και θα μας επιστρέψει την απόκριση. Τέλος, η μέθοδος επιστρέφει ένα αντικείμενο promise με το αποτέλεσμα της κλήσης.

```
this.BASE_URL = '//aetos.it.teithe.gr/~nrammos/php/getdata.php';

this.getRequest = function(queryString) {
    var def = $q.defer();

    return $http.get(this.BASE_URL, {
        params: {
            "queryString": queryString
        }
    }).success(function(response) {
        def.resolve(response);
    }).error(function() {
        def.reject('Error');
    });

    return def.promise;
};
```

Εικόνα 35 Η μέθοδος getRequest του API service

Έτσι, για να ανακτήσουμε για παράδειγμα πράξεις που έχουν αναρτηθεί εντός κάποιου εύρους ημερομηνιών υπάρχει η μέθοδος getDecisionsInRange. Δέχεται σαν παραμέτρους τις ημερομηνίες αρχής και τέλους του διαστήματος που μας ενδιαφέρει, μεταβιβάζει το κατάλληλο ερώτημα, και επιστρέφει το αποτέλεσμα.

```
this.getDecisionsInRange = function(dateFrom, dateTo) {
    return this.getRequest(
        '/search?from_issue_date=' + this.toIsoDate(dateFrom) +
        '&to_issue_date=' + this.toIsoDate(dateTo) +
        '&size=' + this.PAGE_SIZE);
};
```

Εικόνα 36 Η μέθοδος getDecisionsInRange του API service

Σημειώνεται ότι εδώ η μέθοδος toIsoDate μετατρέπει ένα JavaScript αντικείμενο Date σε μορφή ISO 8601 (YYYY-MM-DD), καθώς αυτήν τη μορφή δέχεται το API της Διαύγειας.

```
this.toIsoDate = function(date) {
    return date.getFullYear() + '-' +
        (date.getMonth() + 1) + '-' +
        date.getDate();
};
```

Εικόνα 37 Η μέθοδος toIsoDate του API service

6.3 PHP CLIENT

Ο PHP client αποτελείται από δύο κλάσεις. Η πρώτη είναι wrapper για τις αποκρίσεις του API της Διαύγειας, και περιλαμβάνει τον κωδικό και τα δεδομένα της απόκρισης.

```php
class ApiResponse {
    public $code = 200;
    public $data = null;

    function __construct($code, $data) {
        $this->code = $code;
        $this->data = $data;
    }
}
```

Εικόνα 38 Κλάση ApiResponse του PHP client

Η κλάση OpenDataClient περιέχει το URL του API της Διαύγειας και μεθόδους για αυθεντικοποίηση οι οποίες δε χρησιμοποιούνται εδώ. Κυρίως όμως περιέχει τη μέθοδο getResource η οποία είναι αυτή που αναλαμβάνει τελικά την κλήση προς το API και επιστρέφει ένα αντικείμενο ApiResponse με τα δεδομένα εάν δεν παρουσιαστεί κάποιο σφάλμα (σε αντίθετη περίπτωση σηκώνει exception). Η getResource δέχεται σαν παράμετρο ένα ερώτημα προς το API.

```
public function getResource($resource) {
    $req = new Http_Request($this->baseUrl . $resource,
        array('method'=>HTTP_REQUEST_METHOD_GET));
    $req->addHeader('Connection', 'Keep-Alive');
    $req->addHeader('Accept', 'application/json');

    if(!PEAR::isError($req->sendRequest())) {
        $responseCode = $req->getResponseCode();
        $responseBody = $req->getResponseBody();
        $data = null;

        if($responseCode === 200) {
            $data = $responseBody;
        }

        return new ApiResponse($responseCode, $data);
    }
    else {
        throw new Exception("Error while getting resource $resource");
    }
}
```

Εικόνα 39 Η μέθοδος getRequest του PHP client

Τελικά, το API service της εφαρμογής απευθύνει τις κλήσεις του στο αρχείο getdata.php. Εκεί δημιουργείται ένα αντικείμενο OpenDataClient και καλείται η getResource περνώντας σαν παράμετρο το ερώτημα. Στη συνέχεια τυπώνεται το αποτέλεσμα ώστε να το παραλάβει το API service.

```
<?php

require_once "opendata.php";

$resource = filter_input(INPUT_GET, 'queryString');
$client = new OpenDataClient();
$response = $client->getResource($resource);
if($response->code === 200) {
    print $response->data;
}
else {
    print "Error " . $response->code;
}
```

Εικόνα 40 Το αρχείο getdata.php του PHP client

6.4 CACHE

Στον server 'Αετός' μαζί με την εφαρμογή διατηρείται και ένας προσωρινός αποθηκευτικός χώρος για κάποια σταθερά δεδομένα της Διαύγειας, όπως είναι οι εγγεγραμμένοι φορείς. Δεδομένα που αφορούν έμμεσα κάποια πράξη, όπως το σε ποια κατηγορία ανήκει ο φορέας της πράξης, διατηρούνται πολλές φορές σε ξεχωριστούς πόρους (βλ. 2.4.4 παραπάνω). Στην περίπτωση αυτή, για να αντιστοιχιστεί η πράξη με την κατηγορία του φορέα της, απαιτούνται αρκετές διαδοχικές κλήσεις. Επιπρόσθετα, καθώς τέτοιες αναζητήσεις πολλές φορές γίνονται σειριακά, είναι εύκολο να φανταστεί κανείς ότι μια τέτοια κλήση είναι ιδιαίτερα ακριβή. Με τη χρήση της cache ο χρόνος ολοκλήρωσης τέτοιων κλήσεων έχει περιοριστεί από δευτερόλεπτα (ή και δεκάδες δευτερολέπτων) σε millisecond.

Τα δεδομένα είναι αποθηκευμένα στην cache σε μορφή JSON. Η ανάκτηση αντικειμένων από την cache γίνεται με τη μέθοδο getCachedObject του API service, η οποία λειτουργεί παρόμοια με την getRequest. Δέχεται ως παράμετρο το όνομα του αρχείου που περιέχει το αντικείμενο που θέλουμε, και επιστρέφει ένα promise με το αποτέλεσμα της κλήσης.

```
this.CACHE_URL = '//aetos.it.teithe.gr/~nrammos/cache/';

this.getCachedObject = function(fileName) {
    var def = $q.defer();

    $http.get(this.CACHE_URL + fileName)
    .success(function(response) {
        def.resolve(response);
    })
    .error(function() {
        def.reject('Error');
    });

    return def.promise;
};
```

Εικόνα 41 Η μέθοδος getCachedObject του API service

Έτσι, για παράδειγμα, για να ανακτήσουμε τις κατηγορίες των εγγεγραμμένων φορέων, ανακτούμε το αρχείο org_categories.json το οποίο περιέχει τους φορείς ήδη αντιστοιχισμένους με την ονομασία της κατηγορίας στην οποία ανήκουν.

```
this.getOrgCategoryLabels = function() {
    return this.getCachedObject('org_categories.json');
};
```

Εικόνα 42 Παράδειγμα κλήσης της getCachedObject

6.5 ΣΥΝΟΨΗ ΚΕΦΑΛΑΙΟΥ

Στο κεφάλαιο αυτό έγινε περιγραφή των αλληλεπιδράσεων με το API της Διαύγειας, της server side πλευράς της εφαρμογής, και της cache. Εδώ ολοκληρώνεται το κομμάτι της ανάκτησης και διαχείρισης των δεδομένων. Στο τελευταίο κεφάλαιο θα παρουσιαστεί η χρήση της εφαρμογής και ο τρόπος παρουσίασης των δεδομένων.

7 USER INTERFACE

7.1 ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό θα παρουσιαστεί η χρήση της εφαρμογής και ο τρόπος παρουσίασης των δεδομένων στον χρήστη. Για το user interface χρησιμοποιήθηκε η βιβλιοθήκη UI Bootstrap, η οποία θα παρουσιαστεί στη συνέχεια, μαζί με τη βιβλιοθήκη Bootstrap στην οποία βασίστηκε.

7.2 BOOTSTRAP

Το Bootstrap είναι ένα front-end framework για ανάπτυξη web εφαρμογών, το οποία αρχικά αναπτύχθηκε για εσωτερική χρήση στο Twitter. Προσφέρει πληθώρα έτοιμων στοιχείων και χειριστηρίων όπως ετικέτες (labels), μενού, και κεφαλίδες πλοήγησης, με σκοπό να κάνει την ανάπτυξη ιστοσελίδων μία πιο γρήγορη και εύκολη υπόθεση.

Ένα πολύ σημαντικό χαρακτηριστικό του Bootstrap είναι ότι δίνει τη δυνατότητα για να δημιουργήσει κανείς responsive user interfaces, όπου η διάταξη των στοιχείων της σελίδας αλλάζει δυναμικά ανάλογα με το μέγεθος της οθόνης στην οποία προβάλλεται. Αυτό γίνεται με τη χρήση ενός συστήματος πλέγματος, όπου τα στοιχεία της σελίδας διατάσσονται σε γραμμές και στήλες. Κάθε γραμμή έχει 12 στήλες και σε κάθε στοιχείο ανατίθεται μία ή περισσότερες κλάσεις, η κάθε μία από τις οποίες περιγράφει τον αριθμό στηλών που θα καταλαμβάνει το στοιχείο για ένα συγκεκριμένο μέγεθος οθόνης. Έτσι μπορεί για παράδειγμα κάθε στοιχείο να είναι σε διαφορετική γραμμή σε πολύ μικρές οθόνες (π.χ. σε κινητά), ενώ σε μεγαλύτερες να εμφανίζονται πολλά στοιχεία σε μία γραμμή. Δεδομένης της δυνατότητας εμφώλευσης, μπορούν να δημιουργηθούν ιδιαίτερα περίτεχνες και δυναμικές διατάξεις.

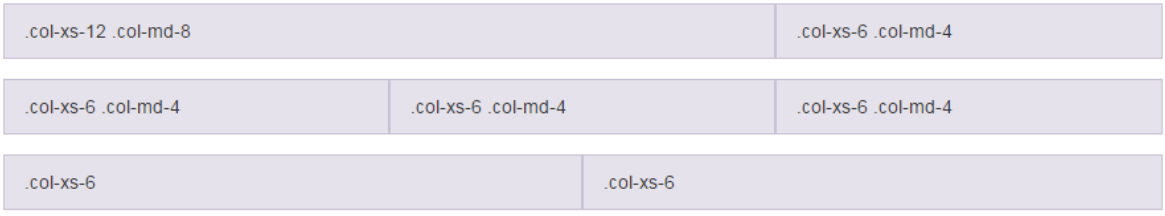
Στο παράδειγμα που ακολουθεί, τα δύο στοιχεία της πρώτης γραμμής εμφανίζονται μαζί σε μεσαίου μεγέθους (medium - md) οθόνες, καθώς το πρώτο καταλαμβάνει 8 στήλες και το δεύτερο 4, σύνολο 12 στήλες, όσο το επιτρεπτό σύνολο στηλών σε μία γραμμή. Σε πολύ μικρές (extra small - xs) οθόνες όμως, το πρώτο στοιχείο καταλαμβάνει 12 στήλες, οπότε θα εμφανίζεται σε μία γραμμή μόνο του. Το δεύτερο στοιχείο της πρώτης γραμμής θα καταλαμβάνει 6 στήλες. Αν και δεν συμπληρώνει

όλες τις στήλες της γραμμής του, παρόλα αυτά θα εμφανίζεται και αυτό σε μία γραμμή μόνο του καθώς τα στοιχεία που ακολουθούν περιέχονται σε διαφορετικές γραμμές.

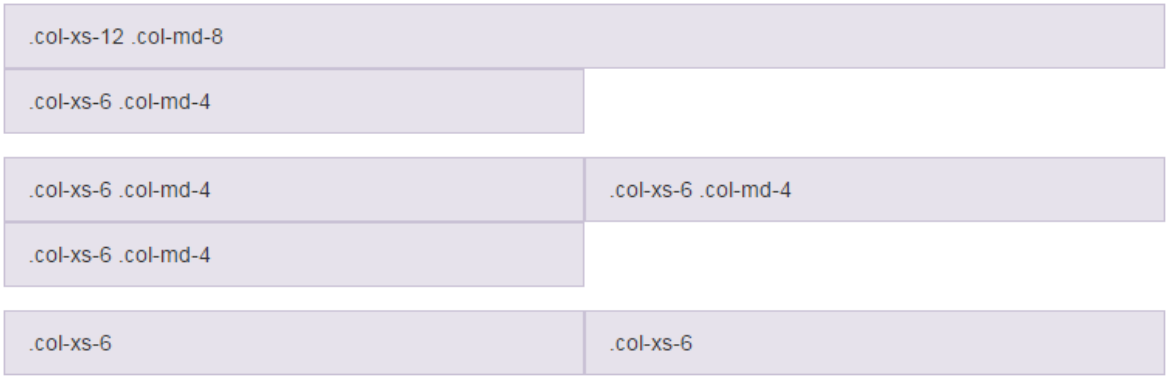
Να σημειωθεί εδώ ότι εάν δεν οριστεί αριθμός στηλών για κάποιο μέγεθος, τότε ισχύει όποιος αριθμός έχει οριστεί για μικρότερα μεγέθη. Σε αυτό το παράδειγμα δηλαδή, η διάταξη για πολύ μικρές οθόνες ισχύει και για τις μικρές (small - sm), ενώ η διάταξη για τις μεσαίου μεγέθους ισχύει και για τις μεγάλες (large - lg). Αντίστοιχα, οι τιμές στα στοιχεία της τελευταίας γραμμής ισχύουν για όλα τα μεγέθη.

```
<!-- Stack the columns on mobile by making one full-width and the other
half-width -->
<div class="row">
    <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>
    <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div

<!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on
desktop -->
<div class="row">
    <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
    <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
    <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

<!-- Columns are always 50% wide, on mobile and desktop -->
<div class="row">
    <div class="col-xs-6">.col-xs-6</div>
    <div class="col-xs-6">.col-xs-6</div>
</div>
```

Εικόνα 43 Παράδειγμα πλέγματος Bootstrap



Εικόνα 44 Παράδειγμα διάταξης σε μικρή οθόνη



Εικόνα 45 Παράδειγμα διάταξης σε μεσαίου μεγέθους οθόνη

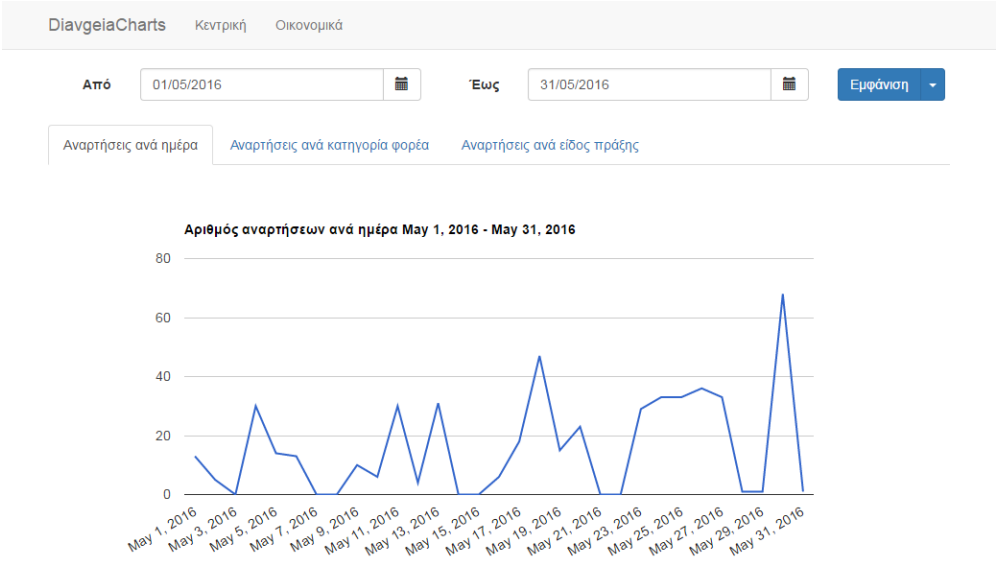
[11]

7.3 UI Bootstrap

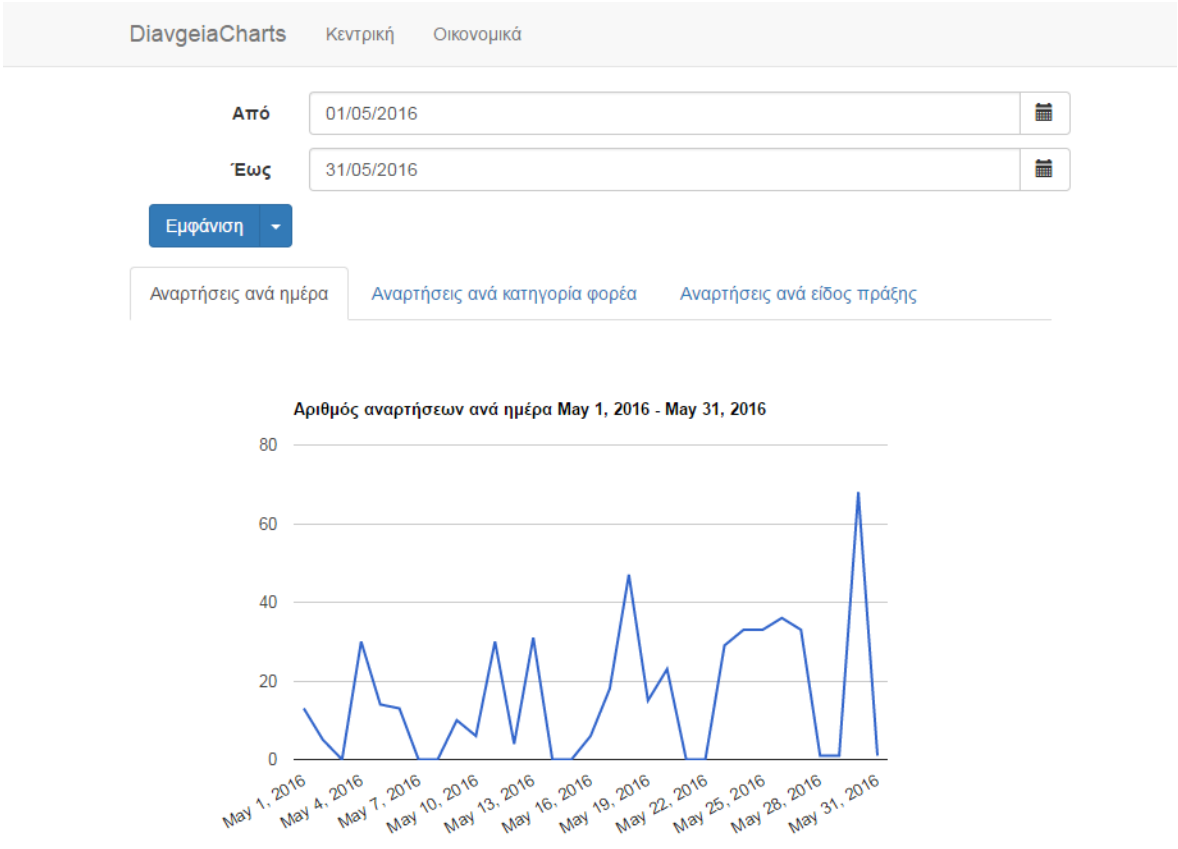
Δυστυχώς το Bootstrap βασίζεται σε jQuery και δε συνεργάζεται καλά με την AngularJS. Γι' αυτόν τον λόγο δημιουργήθηκε το UI Bootstrap από την ομάδα Angular UI, το οποίο είναι γραμμένο σε AngularJS και διασυνδέει το Bootstrap με το data binding και τα directives της AngularJS.

7.4 Σχεδιαση εφαρμογης

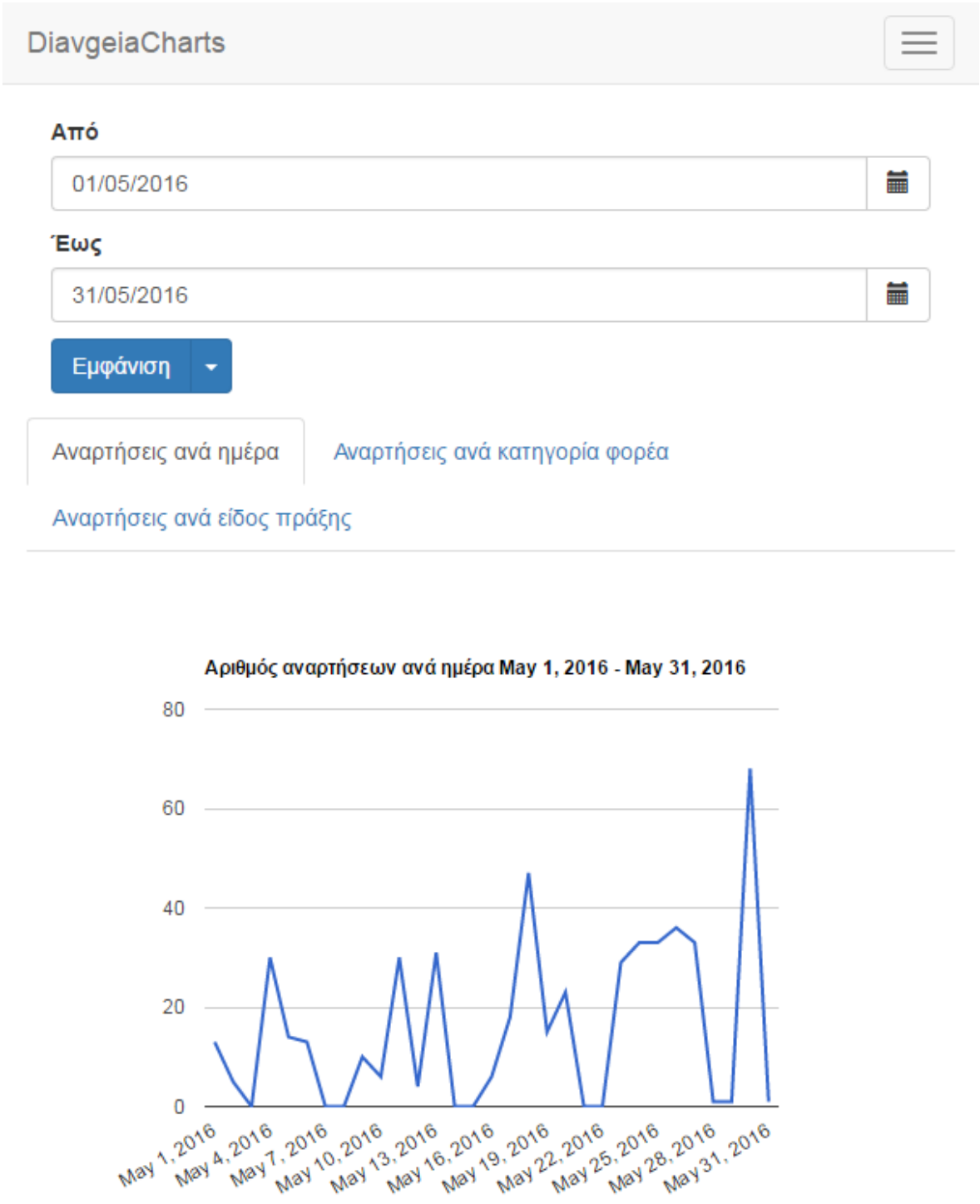
Τα κύρια γραφικά στοιχεία της εφαρμογής είναι μία κεφαλίδα πλοήγησης στην κορυφή, τα χειριστήρια επιλογής εύρους ημερομηνιών ή/και φορέα και αναδόχου για τα οικονομικά γραφήματα, και την περιοχή εμφάνισης των γραφημάτων. Η διάταξη των στοιχείων αλλάζει ανάλογα με το μέγεθος της οθόνης.



Εικόνα 46 Αρχική σελίδα της εφαρμογής, μεσαίου/μεγάλου μεγέθους οθόνη



Εικόνα 47 Αρχική σελίδα της εφαρμογής, μικρή οθόνη



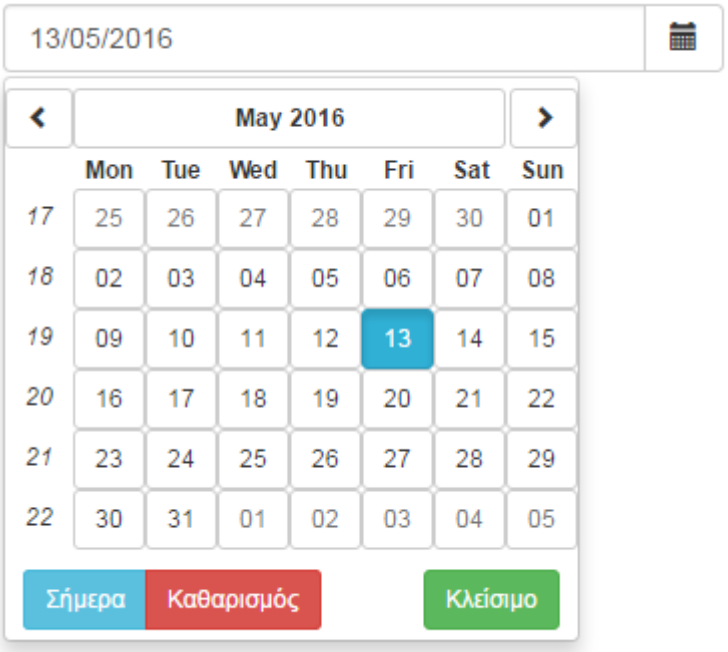
Εικόνα 48 Αρχική σελίδα της εφαρμογής, πολύ μικρή οθόνη

7.4.1 Επιλογή ημερομηνίας

Τα χειριστήρια επιλογής ημερομηνίας αποτελούνται από ένα αντικείμενο input κειμένου που περιέχει την επιλεγμένη ημερομηνία, και ένα κουμπί το οποίο εμφανίζει τον αναδυόμενο διάλογο για επιλογή ημερομηνίας.

Στο αντικείμενο input ορίζουμε τη μορφή της ημερομηνίας, το κείμενο των κουμπιών που περιέχει, και το κείμενο που εμφανίζεται όταν είναι κενό. Ορίζουμε επίσης τη μεταβλητή που περιέχει την τιμή της επιλεγμένης ημερομηνίας με το directive ng-model, με το directive ng-disabled ορίζεται μία μεταβλητή η οποία όταν είναι true απενεργοποιείται το χειριστήριο, και ορίζουμε και μία μεταβλητή που δείχνει εάν ο αναδυόμενος διάλογος είναι εμφανισμένος. Ο χειρισμός των τελευταίων αυτών μεταβλητών γίνεται στον αντίστοιχο controller της εκάστοτε σελίδας.

Στο κουμπί ορίζουμε και πάλι την ίδια μεταβλητή για απενεργοποίηση (αυτή χρησιμοποιείται γενικώς για να απενεργοποιούνται όλα τα χειριστήρια όταν φορτώνει δεδομένα η σελίδα), και με το directive ng-click η μέθοδος που θα καλείται όταν ο χρήστης επιλέγει το κουμπί.



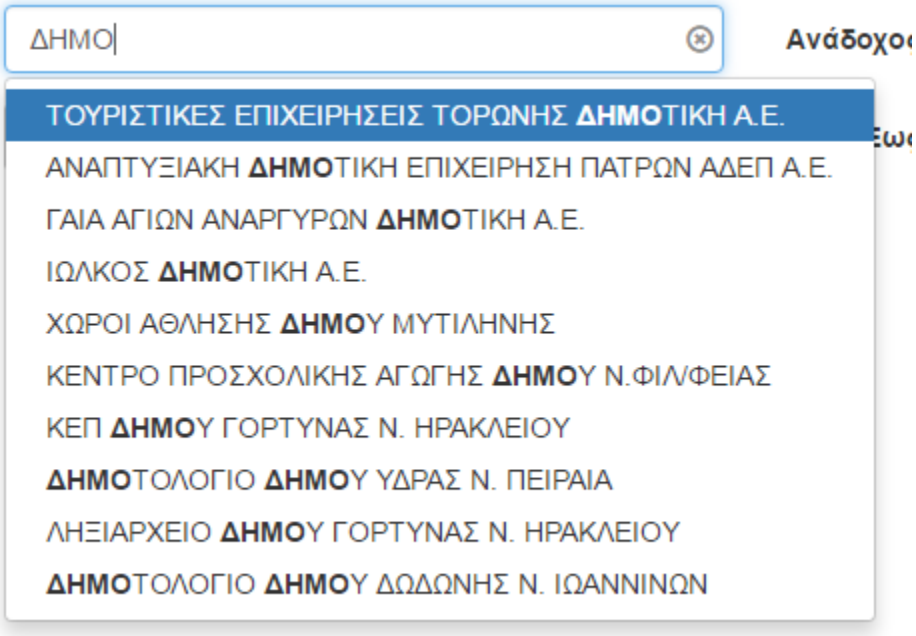

```
<div class="col-xs-12 col-sm-10 col-md-4">
    <p class="input-group">
        <input type="text" class="form-control"
            uib-datepicker-popup="dd/MM/yyyy" ng-model="dateFrom"
            is-open="status.dateFromIsOpen"
            datepicker-options="dateOptions" ng-required="true"
            current-text="Σήμερα" clear-text="Καθαρισμός"
            close-text="Κλείσιμο" ng-disabled="status.loading"
            placeholder="Επιλέξτε ημερομηνία" id="fldDateFrom"/>
        <span class="input-group-btn">
            <button type="button" class="btn btn-default"
                    ng-click="openDateFrom()"
                    ng-disabled="status.loading">
                <i class="glyphicon glyphicon-calendar"></i>
            </button>
        </span>
    </p>
</div>
```

Εικόνα 49 Επιλογή ημερομηνίας

7.4.2 Επιλογή φορέα/αναδόχου

Στη σελίδα οικονομικών γραφημάτων ο χρήστης έχει τη δυνατότητα επιλογής φορέα ή/και αναδόχου. Τα χειριστήρια που χρησιμοποιούνται για αυτόν τον σκοπό περιέχουν ένα στοιχείο input κειμένου και ένα στοιχείο που χρησιμοποιείται για να σβήνει το υπάρχον κείμενο.

Το input εκτός από τα συνηθισμένα περιέχει και directives για αυτόματη φόρτωση προτεινόμενων αποτελεσμάτων. Το πιο σημαντικό είναι το uib-typeahead, το οποίο καλεί τη μέθοδο getOrganizations του controller, και εμφανίζει μία λίστα με ονόματα φορέων που ταιριάζουν με το κείμενο που έχει εισάγει ο χρήστης.




```
<div class="btn-group col-xs-12 col-sm-10 col-md-4">
    <input type="text" class="form-control" id="fldOrgSearch"
        ng-model="selectedOrganization" placeholder="Προαιρετικό"
        ng-disabled="status.loading"
        typeahead-loading="loadingOrganizations"
        typeahead-noresults="noOrganizationsFound"
        typeahead-min-length="3"
        uib-typeahead="label for label in
            getOrganizations($viewValue)">
    <span id="organizationSearchClear"
        class="textFieldClear glyphicon glyphicon-remove-circle"
        ng-click="clearOrganization()"></span>
</div>
```

Εικόνα 50 Επιλογή φορέα/αναδόχου

Η μέθοδος getOrganizations ανακτά το αντικείμενο με τα ονόματα των φορέων από την cache και εμφανίζει τα πρώτα δέκα.

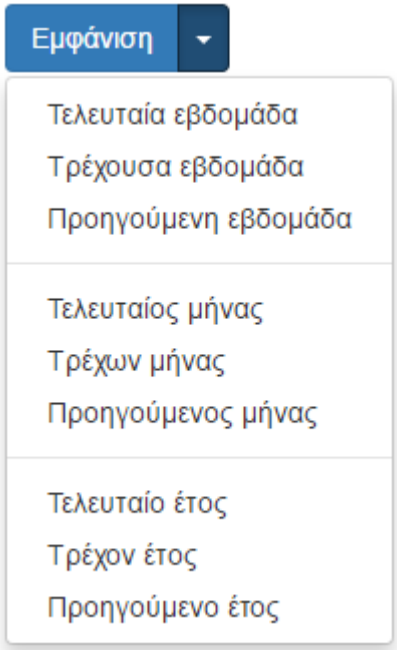
```
$scope.getOrganizations = function(value) {
    return ApiService.getCachedObject('organizations.json')
    .then(function(response) {
        var output = [];

        for(var i = 0; i < response.organizations.length &&
                output.length < 10; i++) {
            if(response.organizations[i].label.toLowerCase()
                    .indexOf(value.toLowerCase()) > -1) {
                output.push(response.organizations[i].label);
            }
        }

        return output;
    }).catch(function(error) {
        alert('Error: ' + error);
    });
};
```

Εικόνα 51 Η μέθοδος getOrganizations του Financial controller

Αξίζει εδώ να σημειωθεί ότι σε αυτό το σημείο της εφαρμογής η χρήση της cache όχι απλά είχε ως αποτέλεσμα μία θεαματική αύξηση στην ταχύτητα, αλλά ήταν και το σημείο το οποίο αποκάλυψε την ανάγκη ύπαρξης cache εξ ’αρχής. Συγκεκριμένα, ενώ αυτή η κλήση αρχικά χρειαζόταν πάνω από δέκα δευτερόλεπτα για να ολοκληρωθεί, με χρήση cache ολοκληρώνεται στιγμιαία.

7.4.3 Κουμπί εμφάνισης γραφημάτων

Το κουμπί ‘Εμφάνιση’ χρησιμοποιείται για να εμφανίσει τελικά το γράφημα στη σελίδα, ανάλογα με τις επιλογές του χρήστη. Το κουμπί έχει δύο τρόπους χρήσης. Κάνοντας απλά κλικ σε αυτό, φορτώνεται το αντίστοιχο γράφημα για το εύρος ημερομηνιών που έχει εισάγει ο χρήστης. Κάνοντας κλικ στο βελάκι στη δεξιά μεριά, ο χρήστης έχει τη δυνατότητα να επιλέξει γρήγορα ένα από τα προ ρυθμισμένα εύρη ημερομηνιών, όπως τελευταίος μήνας (τελευταίες 30 ημέρες), τρέχων μήνας (πρώτη έως τελευταία ημέρα του μήνα που διανύουμε), προηγούμενος μήνας (πρώτη έως τελευταία ημέρα του προηγούμενου μήνα). Χάριν συντομίας δεν παρουσιάζεται ο κώδικας όλων των επιλογών.




```
<div class="col-xs-12 col-md-2">
    <div class="btn-group" uib-dropdown>
        <button id="split-button" type="button" class="btn btn-primary"
                ng-disabled="status.loading" ng-click="load()">
            Εμφάνιση
            <i class="glyphicon glyphicon-refresh"
                ng-show="status.loading"></i>
        </button>
        <button type="button" class="btn btn-primary"
                uib-dropdown-toggle ng-disabled="status.loading">
            <span class="caret"></span>
        </button>
        <ul class="uib-dropdown-menu" role="menu">
            <li role="menuitem">
                <a href="" ng-click="loadLastWeek()">
                    Τελευταία εβδομάδα
                </a>
            </li>
            <li role="menuitem">
                <a href="" ng-click="loadCurrentWeek()">
                    Τρέχουσα εβδομάδα
                </a>
            </li>
        </ul>
    </div>
</div>
```

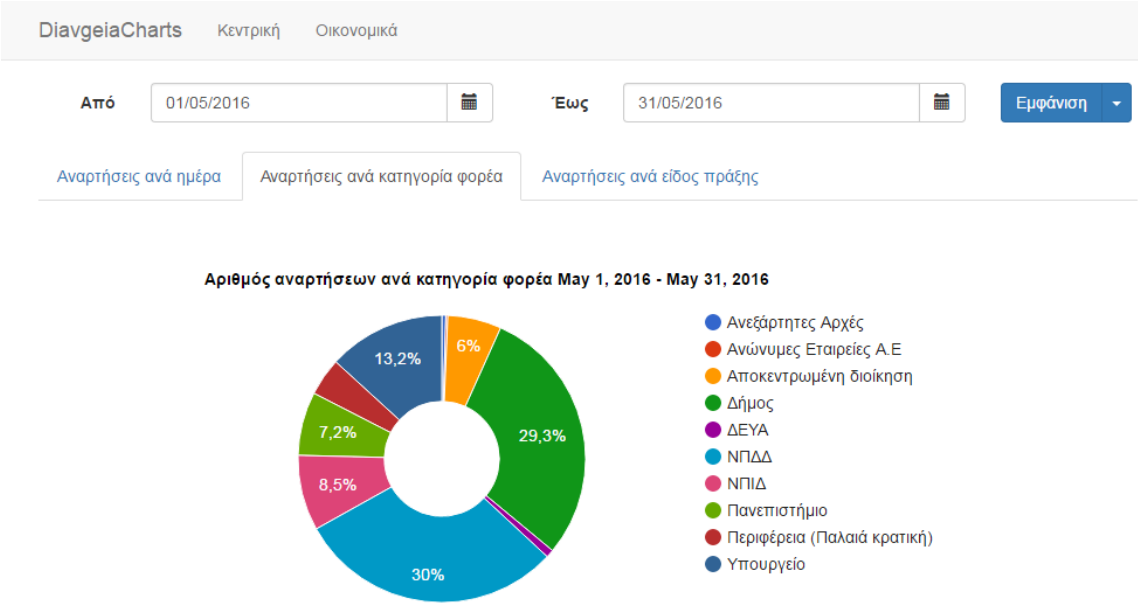
Εικόνα 52 Κουμπί εμφάνισης γραφημάτων

7.4.4 Γραφικές παραστάσεις

Ανάλογα με τις επιλογές του χρήστη, εμφανίζεται η αντίστοιχη γραφική παράσταση. Στην κεντρική σελίδα υπάρχει ένας χώρος με tabs, όπου ο χρήστης μπορεί να εναλλάσσει μεταξύ των διαφορετικών παραστάσεων χωρίς να ξαναφορτώνει τα δεδομένα. Όταν ο χρήστης επιλέγει ένα από αυτά τότε καλείται η αντίστοιχη μέθοδος η οποία εφόσον υπάρχουν διαθέσιμα δεδομένα δημιουργεί και εμφανίζει τη γραφική παράσταση.

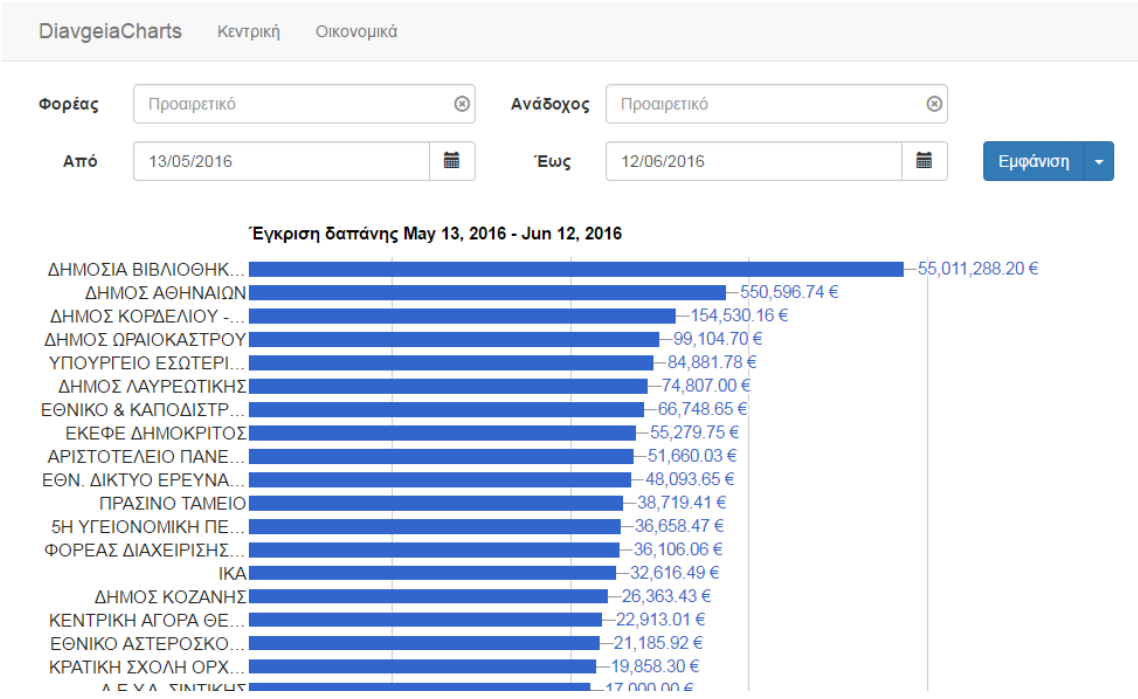
```
<uib-tabset >
    <uib-tab heading="Αναρτήσεις ανά ημέρα"
            active="status.dayTabIsActive" select="drawDayChart()">
        <div class="row">
            <div class="col-xs-12 chart_div" id="chart_div_day"></div>
        </div>
    </uib-tab>
    <uib-tab heading="Αναρτήσεις ανά κατηγορία φορέα"
            active="status.orgTabIsActive" select="drawOrgChart()">
        <div class="row">
            <div class="col-xs-12 chart_div" id="chart_div_org"></div>
        </div>
    </uib-tab>
    <uib-tab heading="Αναρτήσεις ανά είδος πράξης"
            active="status.typeTabIsActive" select="drawTypeChart()">
        <div class="row">
            <div class="col-xs-12 chart_div" id="chart_div_type"></div>
        </div>
    </uib-tab>
</uib-tabset>
```

Εικόνα 53 Tabs με διαφορετικά είδη γραφημάτων

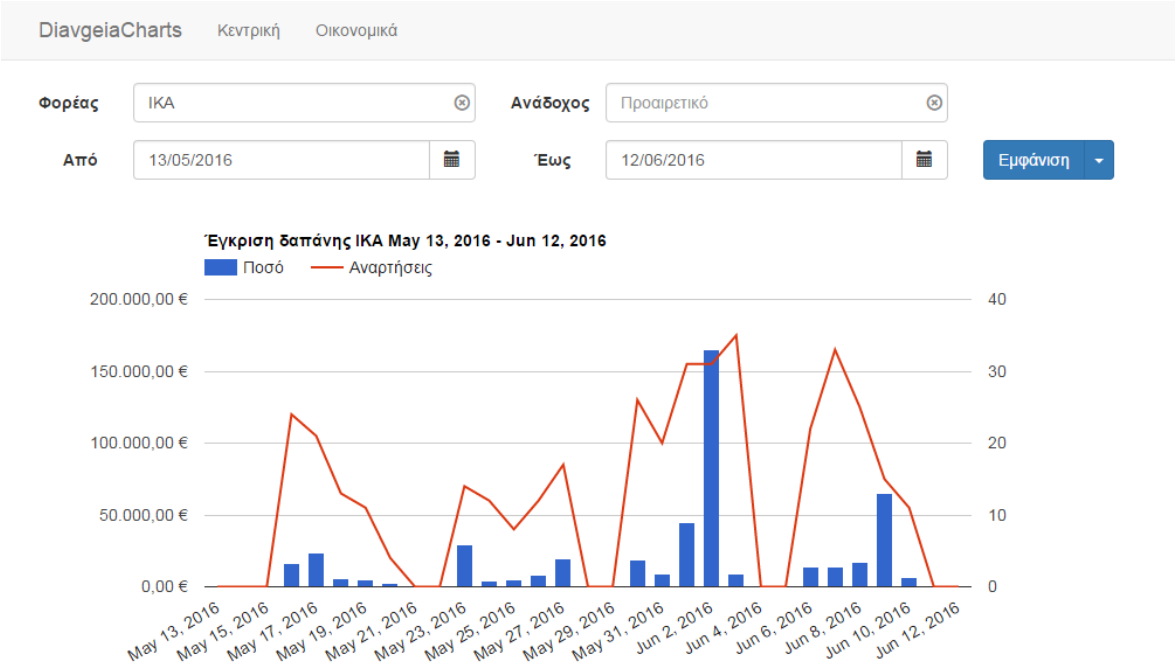


Εικόνα 54 Γραφική παράσταση αναρτήσεων ανα κατηγορία φορέα

Στη σελίδα των οικονομικών εμφανίζεται είτε ένα γράφημα με το συνολικό ποσό ανά φορέα εάν ο χρήστης δεν έχει επιλέξει κάποιον φορέα ή ανάδοχο, διαφορετικά εμφανίζεται ένα σύνθετο γράφημα για τον φορέα ή ανάδοχο το οποίο δείχνει τόσο τον αριθμό αναρτήσεων όσο και το συνολικό ποσό για κάθε ημέρα της επιλεγμένης περιόδου.



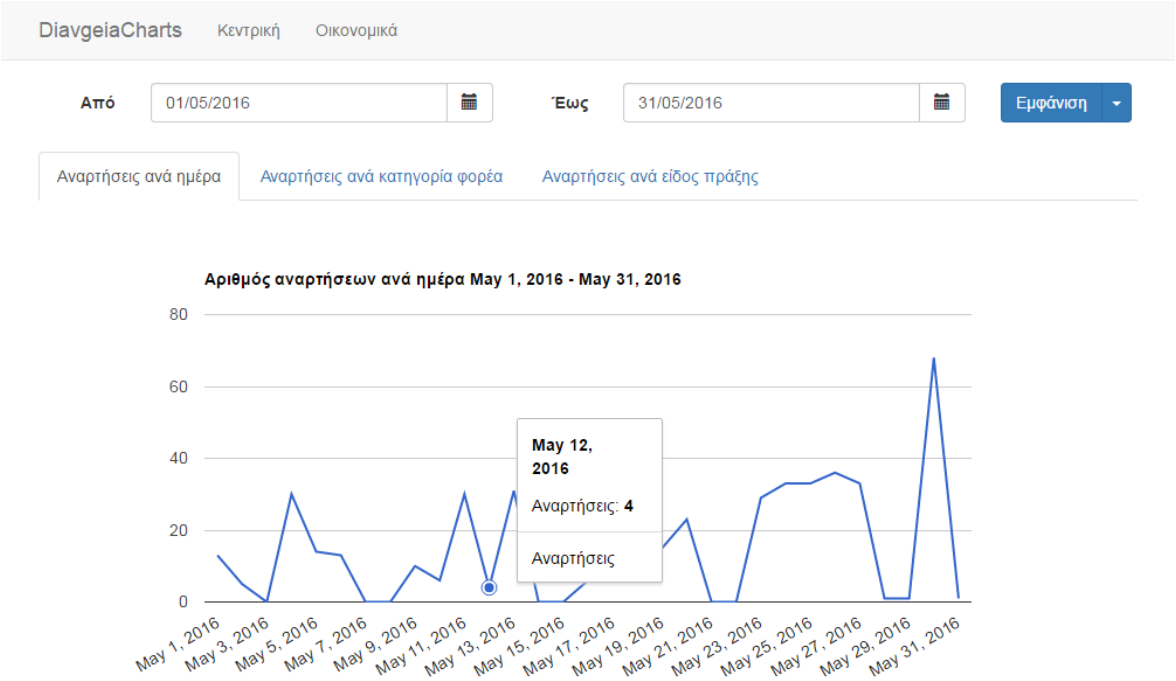
Εικόνα 55 Γραφική παράσταση συνολικών δαπανών ανά φορέα



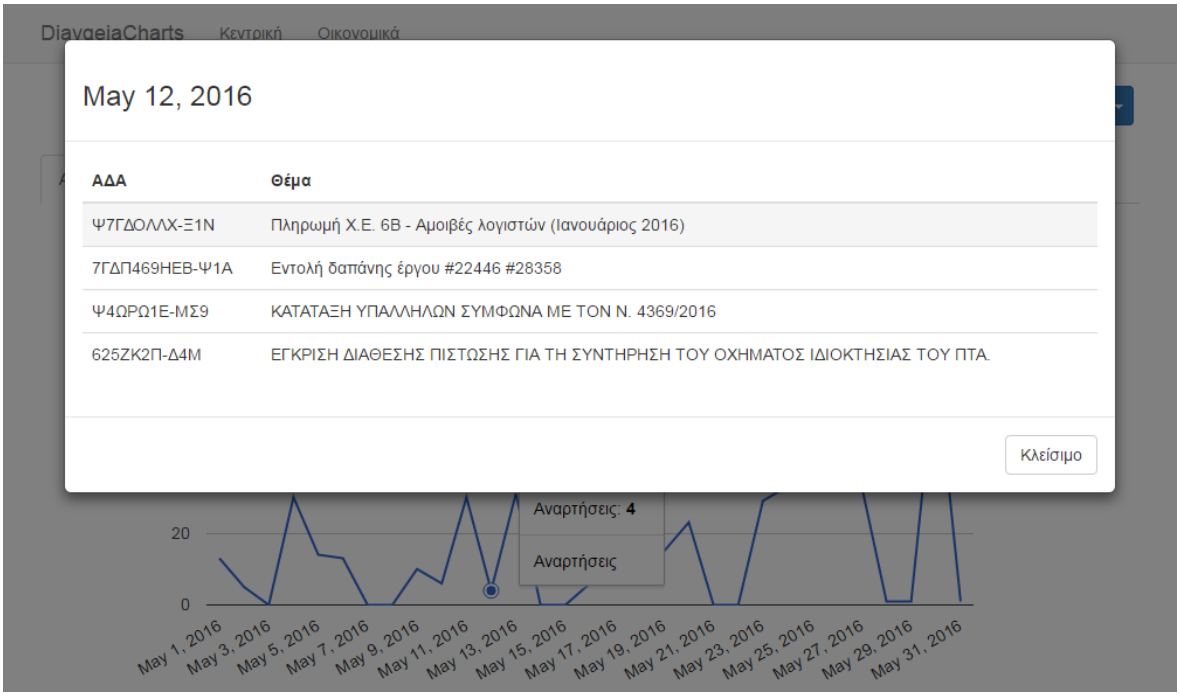
Εικόνα 56 Γραφική παράσταση αναρτήσεων και δαπανών φορέα/αναδόχου ανά ημέρα

7.4.5 Tooltips

Ο χρήστης έχει τη δυνατότητα, επιλέγοντας κάποιο σημείο ενός γραφήματος, να εμφανίσει το tooltip του γραφήματος. Το tooltip περιέχει κάποιες βασικές πληροφορίες για το επιλεγμένο σημείο (π.χ. ημερομηνία και αριθμό αναρτήσεων). Δίνει επίσης στον χρήστη την επιλογή να εμφανίσει έναν διάλογο με τη λίστα των αναρτήσεων για το επιλεγμένο σημείο, από όπου επιλέγοντας μία ανάρτηση μεταφέρεται στη σελίδα της ανάρτησης στη Διαύγεια (βλ. 5.3.2 και 5.4 παραπάνω). Από τη συγκεντρωτική γραφική παράσταση οικονομικών, το tooltip παρέχει επίσης την επιλογή για εμφάνιση του σύνθετου γραφήματος για τον επιλεγμένο φορέα.



Εικόνα 57 Tooltip γραφήματος



Εικόνα 58 Διάλογος με τις αναρτήσεις για το επιλεγμένο σημείο του γραφήματος

Ο διάλογος με τις αναρτήσεις περιέχει έναν πίνακα με τις αναρτήσεις του επιλεγμένου σημείου τις οποίες παίρνει από το tooltip, και ένα κουμπί για κλείσιμο του διαλόγου (ο διάλογος κλείνει και όταν ο χρήστης κάνει κλικ οπουδήποτε εκτός του διαλόγου).

```
<table class="table table-hover table-responsive">
    <thead>
        <tr>
            <th>ΑΔΑ</th>
            <th>Θέμα</th>
        </tr>
    </thead>
    <tbody>
        <tr ng-repeat="decision in modal.data track by $index"
                ng-click="openDecisionInNewTab(decision.ada)"
                style="cursor: pointer">
            <td style="white-space: nowrap">{{decision.ada}}</td>
            <td>{{decision.subject}}</td>
        </tr>
    </tbody>
</table>
```

Εικόνα 59 Πίνακας αναρτήσεων στον διάλογο

7.5 Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκαν με συντομία τα βασικά στοιχεία της διεπιφάνειας χρήστη της εφαρμογής όπως η επιλογή ημερομηνίας και η εμφάνιση των γραφημάτων, καθώς και οι τεχνολογίες που χρησιμοποιήθηκαν για τη δημιουργία της, φτάνοντας έτσι στο τέλος της παρουσίασης της εφαρμογής και του κειμένου αυτής της εργασίας.

8 ΣΥΜΠΕΡΑΣΜΑΤΑ

Από τη σύντομη ενασχόληση κατά την ανάπτυξη της εφαρμογής προκύπτει ότι ένα καλοσχεδιασμένο RESTful API κάνει την επικοινωνία ιδιαίτερα εύχρηστη και αποδοτική. Τα Google charts προσφέρουν την πολύ γρήγορη ενσωμάτωση οπτικοποίησης δεδομένων σε μία εφαρμογή, ενώ αντίστοιχα το Bootstrap δίνει τη δυνατότητα να δημιουργήσει κανείς μια εμφανίσιμη εφαρμογή με ελάχιστο κόπο. Όταν όλα αυτά συνδυάζονται με τη δύναμη της Angular, τότε ακόμα και σχετικά αρχάριοι μπορούν να δημιουργήσουν επαγγελματικές εφαρμογές εύκολα.

Όλα αυτά έχουν ιδιαίτερη σημασία καθώς η απλότητα και ταχύτητα ανάπτυξης προωθούν την ανάπτυξη ακόμα πιο περίτεχνων και δυνατών εφαρμογών. Αν και καμία από αυτές τις τεχνολογίες δεν εξερευνήθηκε πλήρως στα πλαίσια της εργασίας, αυτές προσφέρουν ακόμα πιο δυνατά εργαλεία από αυτά που παρουσιάστηκαν εδώ και σίγουρα θα εξελιχθούν ακόμη περισσότερο στο μέλλον.

9 ΑΝΑΦΟΡΕΣ

[1] «Representational State Transfer,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer. [Πρόσβαση 7 May 2016].

[2] R. Fielding, «Architectural Styles and the Design of Network-based Software Architectures,» 2000. [Ηλεκτρονικό]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. [Πρόσβαση 7 May 2016].

[3] «JSON,» [Ηλεκτρονικό]. Available: www.json.org. [Πρόσβαση 7 May 2016].

[4] Ecma International, «The JSON Data Interchange Format,» October 2013. [Ηλεκτρονικό]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf. [Πρόσβαση 7 May 2016].

[5] «Δι@ύγεια - API Ανοιχτών Δεδομένων,» [Ηλεκτρονικό]. Available: https://diavgeia.gov.gr/api/help. [Πρόσβαση 15 May 2016].

[6] «AngularJS: Developer Guide: Conceptual Overview,» [Ηλεκτρονικό]. Available: https://docs.angularjs.org/guide/concepts. [Πρόσβαση 15 May 2016].

[7] «AngularJS: API: $q,» [Ηλεκτρονικό]. Available: https://docs.angularjs.org/api/ng/service/$q. [Πρόσβαση 22 May 2016].

[8] «Google Chart API,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Google_Chart_API. [Πρόσβαση 15 May 2016].

[9] «Charts | Google Developers,» [Ηλεκτρονικό]. Available: https://developers.google.com/chart/. [Πρόσβαση 15 May 2016].

[10] «PHP samples of OpenData API usage for new Diavgeia system,» [Ηλεκτρονικό]. Available: https://github.com/diavgeia/opendata-client-samples-php. [Πρόσβαση 11 May 2016].

[11] «CSS - Bootstrap,» [Ηλεκτρονικό]. Available: http://getbootstrap.com/css/. [Πρόσβαση 12 June 2016].