Τίτλος

Συγγραφέας

Διπλωματική Εργασία

Επιβλέπων: Π. Βασιλειάδης

Ιωάννινα, Μήνας Έτος

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY OF IOANNINA

Ευχαριστίες

Ευχαριστίες προς κάθε ενδιαφερόμενο (προαιρετικό κεφάλαιο).

Ημερομηνία

Συγγραφέας

Περίληψη στα ελληνικά

Περίληψη στα ελληνικά (έως 200 λέξεις).

Λέξεις Κλειδιά: μπλα μπλα

Abstract

Summary in English (up to 200 words)

Keywords: <keyword 1>, <keyword 2>

Πίνακας περιεχομένων

Κεφάλαιο 1. Εισαγωγή....1

1.1 Αντικείμενο της διπλωματικής....1

1.2 Οργάνωση του τόμου....1

Κεφάλαιο 2. Περιγραφή Θέματος....3

2.1 Στόχος της εργασίας....3

2.2 Σχετικές εργασίες και τεχνολογίες....3

2.3 Ανάλυση απαιτήσεων....3

Κεφάλαιο 3. Σχεδίαση & Υλοποίηση....5

3.1 Ορισμός προβλήματος και αλγόριθμοι επίλυσης....5

3.2 Σχεδίαση και αρχιτεκτονική λογισμικού....5

3.3 Σχεδίαση και αποτελέσματα ελέγχου του λογισμικού....6

3.4 Λεπτομέρειες εγκατάστασης και υλοποίησης....6

3.5 Επεκτασιμότητα του λογισμικού....6

Κεφάλαιο 4. Πειραματική Αξιολόγηση....7

4.1 Μεθοδολογία πειραματισμού....7

4.2 Αναλυτική παρουσίαση αποτελεσμάτων....7

Κεφάλαιο 5. Επίλογος....11

5.1 Σύνοψη και συμπεράσματα....11

5.2 Μελλοντικές επεκτάσεις....11

Κεφάλαιο 1. Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Το JavaScript Object Notation (JSON) είναι ένα open_standard αρχείο που είναι γραμμένο σε γλώσσα ανθρώπου. Τα JSON δημιουργήθηκαν στην ανάγκη για stateless, real-time server-to-browser πρωτόκολλο επικοινωνίας χωρίς να χρησιμοποιεί plugins όπως Flash ή Java applets στις αρχές του 2000. Ο Douglas Crockford ήταν ο πρώτος που καθόρισε και δημοσίευεσε το JSON format. Τα JSON είναι ένα υποσύνολο της JavaScript και είναι συνηθές το φαινόμενο να χρησιμοποιείται σε αυτήν, αλλά είναι language-independent data format, αλλά χρησιμοποιεί κανόνες που είναι παρόμοιοι με της γλώσσες κατηγορίας C περιέχοντας C, C++,C#, Java, JavaScript και πολλές άλλες. Ωστόσο, υπάρχει διαθέσιμος κώδικας από πολλές γλώσσες προγραμματισμού για parsing και generating. Τα JSON αρχεία αποθηκεύονται με την κατάληψη ".json". Όσο για τους τύπους δεδομένων και την σύνταξη που χρησιμοποιεί έχουμε τους βασικούς τύπους δεδομένων:

- Number: Είναι ένας signed decimal number, αλλά δεν γίνεται να χρησιμοποιεί non-numbers και NaN.
- String: Μια πρόταση από μηδέν ή περισσότερους χαρακτήρες Unicode, έχοντας στην αρχή και στο τέλος διπλά εισαγωγικά ("").
- Boolean: Μπορεί να πάρει τιμές true ή false.
- Array: Μια ordered λίστα με μηδέν ή περισσότερες τιμές που ο τύπος καθέ μία από αυτές μπορεί να ναι οτιδήποτε. Τους πίνακες τους γράφουμε με bracket στην αρχή και στο τέλος και κόμμα ανάμεσα σε κάθε στοιχείο.

- Object: Μια unordered συλλογή από name-value ζευγάρια όπου τα names ή αλλιώς και keys είναι strings. Τα objects γράφονται με curly brackers στην αρχή και στο τέλος, χρησιμοποιούμε κόμμα ανάμεσα από κάθε ζευγάρι name-value και ανάμεσα στο name-value άνω και κάτω τελεία (:).
- null: Μια κενή τιμή.

Το whitespace επιτρέπεται αλλά αγνοείται ανάμεσα ή τριγύρω από τα elements. Τέσσερεις συγκεκριμένοι χαρακτήρες θεωρούνται whitespace στα JSON που είναι οι: space, horizontal tab, line feed, carriage return. Ακολουθεί ένα παράδειγμα JSON που περιγράφει ένα άτομο.

```
{
    “firstName”: “John”,
    “lastName”: “Smith”,
    “isAlive”: true,
    “age”: 27,
    “adress”: {
        “streetAddress”: “21 2nd Street”,
        “city”: “New York”,
        “state”: “NY”,
        “postalCode”: “10021-3100”
    },
    “phoneNumbers”: [
        {
            “type”: “home”,
            “number”: “212 555-1234”
        },
        {
            “type”: “office”,
            “number”: “646 555-4567”
        },
        {
            “type”: “mobile”,
```

“number”: “+91 9629787781”
}
],
“children”: [],
“spouse”: null
}

1.2 Οργάνωση του τόμου

Ποια τα κεφάλαια / ενότητες του τόμου αυτού (1 παράγραφος ανά κεφάλαιο).

(Τα κεφάλαια πρέπει να ξεκινούν σε ΜΟΝΗ ΣΕΛΙΔΑ, κι εδώ αφήνουμε κενή σελίδα επίτηδες, ώστε το κεφάλαιο 2 να ξεκινά στη σελ. 3. Αν δεν είχαμε αφήσει την παρούσα κενή σελίδα, το κεφάλαιο θα ξεκινούσε στη σελ. 2 που είναι ζυγή.)

Κεφάλαιο 2. Περιγραφή Θέματος

2.1 Στόχος της εργασίας

Ο στόχος της παρούσας διπλωματικής εργασίας είναι να κατασκευαστεί ενα σύστημα το οποίο θα επιτρέπει στον χρήστη την εισαγωή ενός αρχείου δεδομένων τύπου JSON με αποτέλεσμα να προβάλλονται οπτικοποιημένα το σχήμα των δεδομένων, καθώς και οι αλλαγές που πραγματοποιήθηκαν στη διάρκεια του οποίου, οι διαχειριστές των δεδομένων είτε μπορεί να πρόσθεσαν,να αφαίρεσαν ή να μετέβαλλαν το φώλιασμα των τιμών.

Συγκεκριμένα οι απαιτήσεις του συστήματος οργανώνονται ως εξής:

- Θα πρέπει ο χρήστης να μπορεί να φορτώσει στο σύστημα το αρχείο δεδομένων τύπου JSON που θέλει να εισάγει ώστε να ξεκινήσει η διαδικασία εξαγωγής του σχήματος των δεδομένων. Αντικείμενο της διπλωματικής είναι η εξαγωγή του σχήματος δεδομένων, καθώς και των πολλαπλών εκδόσεων αν υπάρχουν μέσα στον χρόνο.
- Ο χρήστης θα μπορεί να δει τα αποτελέσματα των σχημάτων δεδομένων και των διαφορετικών εκδόσεων.
- Τέλος, έχοντας τελειώσει η επεξεργασία του αρχείου που εισήγαγε ο χρήστης θα έχει την επιλογή να δει τα χαρακτηριστικά που παρουσιάζει η εξέλιξη του σχήματος του JSON αρχείου.

2.2 Σχετικές εργασίες και τεχνολογίες

Υπάρχουν αρκετές βιβλιοθήκες που μπορούν να βοθήσουν κάποιον όταν θέλει να επεξεργαστεί αρχεία τύπου JSON προγραμματίζοντας σε γλώσσα java. Μερικές από αυτές είναι οι mJson. JSON.simple, JSON-P, GSON, Jackson, με τις GSON και Jackson να ναι οι πιο δημοφιλείς από αυτές, παρουσιάζοντας αρκετές ομοιότητες μεταξύ τους. Όπως θα δούμε παρακάτω η mJSON είναι μικρή βιβλιοθήκη και η GSON και Jackson οι βιβλιοθήκες με τις περισσότερες λειτουργίες. Οι JSON.simple και JSON-P έχουν παρόμοια λειτουργικότητα, οπότε ασχοληθήκαμε με mJSON και JSON-P στην αρχή και μετά με τις δύο μεγαλύτερες GSON, Jackson.

mJSON

https://bolerio.github.io/mjson/

Η mJson είναι μία αρκετά μικρή βιβλιοθήκη με πολύ συνοπτικό API. Σε αυτήν την βιβλιοθήκη υπάρχει μια κλάση με όλες τις μεθόδους που μπορεί κάποιος να χρησιμοποιήσει. Σε αντίθεση με άλλες JSON βιβλιοθήκες τις οποίες θα δούμε παρακάτω, επικεντρώνεται στο να διαχειρίζεται τα JSON structures στην Java, υποστηρίζοντας parsing από Json objects σε Java objects και το ανάποδο. Ο κύριος στόχος αυτής της βιβλιοθήκης είναι να επιτρέπει στον προγραμματιστή να δουλεύει με JSON στην Java όπως δουλεύει κανείς σε JavaScript. Μερικά χαρακτηριστικά της είναι:

- Πλήρης υποστήριξη του JSON Schema Draft 4 validation, ένα πρότυπο-δημιουργία του IETF- το οποίο παρέχει ένα σχήμα για την επαλήθευση ενός Json object που περιέχει τον αναμενόμενο τύπο εντός του σχήματος.
- Ενάς μοναδικός τύπος μεταβλητών, τα πάντα είναι Json και δεν χρειάζεται type casting.
- Γρήγορο parsing.
- Περιεκτικές μεθόδους για διάβασμα, τροποποίηση, αντιγραφή και συγχώνευση.
- Μεθόδοι για type-check και πρόσβαση για έλεγχο τιμών.
- Ένα java source file για ολόκληρη την βιβλιοθήκη, χωρίς external dependencies.

GSON

https://github.com/google/gson

Η GSON είναι μια Java-based βιβλιοθήκη την οποία δημιούργησε η Google για προσωπική της χρήση και μετά την παραχώρησε για δημόσια χρήση. Η GSON χρησιμοποιείται για να μετατρέψει ένα JSON string σε ένα ισοδύναμο Java object και το ανάποδο. Υποστηρίζει απλές μεθόδους όπως toJson() και fromJson() για να μετατρέπει ο προγραμματιστής Java objects σε Json objects και vice versa, να διαχειρίζεται collections, generic types και nested classes. Επίσης υποστηρίζει serialization/deserialization που μπορούν εξειδικευθούν. Το serialization είναι ο τρόπος να δημιουγήσουμε Json objects μέσω Java objects και το deserialization το ανάποδο. Όσο αναφορά το custom data part επιτρέπει στον προγραμματιστή να έχει τον πλήρη έλεγχο της όλης επεξεργασίας των Json objects. Ένα απλό deserialization παράδειγμα είναι όταν έχουμε Json objects με πεδία name, supername, age τότε ο προγραμματιστής μπορεί να φτιάξει μια κλάση Person με τα πεδία αυτά ώστε όταν περνάει το Json object να χρησιμοποιείται η κλάση Person και να αποθηκεύονται αυτόματα στα πεδία οι τιμές.

Στην GSON θα βρούμε τα εξής πακέτα:

- com.google.gson: Προσφέρει access στην GSON, την βασική κλάση για να δουλέψει κανείς με την GSON.
- com.google.gson.annotations: Προσφέρει annotation τύπους για χρήση με την GSON.
- com.google.gson.reflect: Δαχειρίζεται πληροφορίες από generic τύπους.
- com.google.gson.stream: Προσφέρει κλάσεις για reading/writing σε Json-encoded τιμές.

Μερικά χαρακτηριστικά της GSON είναι:

- Προσφέρει απλές μεθόδους toJson() και fromJson() για να μετατρέψουμε Java objects σε Json και το ανάποδο.
- Deserialization/Serialization (JSON to/from POJOs).
- Υποστηρίζει Java Generics.
- Υποστηρίζει περίπλοκα αντικείμενα.

Jackson

https://github.com/FasterXML/jackson

Η Jackson είναι ένας high-performance JSON processor για την Java και θεωρείται και η πιο διάσημη βιβλιοθήκη σύμφωνα με την χρήση στο Github. Η Jackson ενώ χρησιμοποιεί διαφορετικούς όρους για να ονομάσει τις λειτουργίες της παρουσιάζει πολλές ομοιότητες με την GSON. Στη διάρκεια των δοκιμών που εκτελέσαμε, παρατηρήσαμε ότι η Jackson δεν είναι τόσο user-friendly, λόγω του αυξημένου μεγέθους της, όταν ξεκινήσαμε να διαβάζουμε ένα json file προσπαθώντας να μετατρέψουμε τα Json objects σε Java objects δυσκολευτήκαμε να κατανοήσουμε τον τρόπο λειτουργίας της βιβλιοθήκης. Σε αντίθεση με την GSON η οποία είναι user friendly. Στην Jackson χρησιμοποιούνται και εδώ οι όροι serialization/deserialization που εξηγήθηκαν παραπάνω στην GSON τι ακριβώς είναι. Στην Jackson θα βρούμε τα εξής πακέτα:

- jackson-core: Ορίζει ένα low-level streaming API και περιλαμβάνει JSON-specific implementations.
- jackson-annotations: Περιλαμβάνει standard Jackson annotations.
- Jackson-databind: Implements data-binding, υποστηρίζει streaming πακέτα και εξαρτάται από τα παραπάνω πακέτα.

Πέρα απο τα πακέτα που χρησιμοποιούνται αυτούσια από την βιβλιοθήκη, έχουμε και εξωτερικά πακέτα που μπορεί κάποιος να χρησιμοποιήσει για να δουλέψει σε δεδομένα τύπου Avro, BSON, CBOR, CSV, Smile, Protobuf, XML, YAML και πολλά άλλα.

Μερικά χαρακτηριστικά της Jackson είναι:

- Χρησιμοποιεί δύο μεθόδους για serialization/deserialization όπως Object Mapper, JsonParser, JsonGenerator.
- Data-binding (JSON to/from POJOs).
- Υποστηρίζει Java Generics.
- Υποστηρίζει περίπλοκα αντικείμενα.
- Treebased data model.
- Mix-in Annotations.
- Polymorphic types.
- Materialized interfaces.

Εδώ θα μπει η περιγραφή της jsonp.

Το benchmark θα γίνει στο επόμενο διάστημα με τις mJson, Gson, Jackson, JSON-P.

Το είχε έτοιμο από άρθρο που σας είχα πει, οπότε θα κάτσω 2 μέρες να το φτίαξω με
δικά μου παραδείγματα αλλά θα είναι έτσι όπως το συγκεκριμένο.

Benchmark

Για να αποφασίσουμε ποια βιβλιοθήκη θα χρησιμοποιήσουμε για το σκοπό της διπλωματικής εργασίας,το benchmark πραγματοποιήθηκε με 2 σενάρια, στο ένα περάστηκαν μεγάλα αρχεία(190 MB) και στο δεύτερο περάστηκαν μικρά αρχεία (1 KB). Τα μεγάλα αρχεία είναι από: https://github.com/zeMirco/sf-city-lots-json και τα μικρά αρχεία φτιάχτηκαν τυχαία από: http://www.json-generator.com/. Έγιναν δέκα εκτελέσεις για κάθε αρχείο ανά βιβλιοθήκη και για τα μεγάλα και για τα μικρά αρχεία. Αφού καθορίστηκε το μέγεθος των μεγάλων αρχείων, έγιναν δέκα επαναλήψεις ανά εκτέλεση για κάθε βιβλιοθήκη. Για κάθε μικρό αρχείο έγιναν δέκα χιλιάδες επαναλήψεις ανά εκτέλεση για κάθε βιβλιοθήκη. Τα αποτελέσματα των μεγάλων αρχείων φαίνονται στον πίνακα παρακάτω:

BIG FILE (190 MB - 10 rounds/time) (ms)				
	JSON.simple	GSON	JACKSON	JSONP
1	140223	291326	129486	206147
2	139053	291666	122788	204040
3	144015	295234	146030	206072
4	138612	293193	157452	201100
5	138457	295358	128371	205098
6	138767	292968	157522	201898
7	140757	291798	128252	198929
8	140143	289552	143727	204569
9	138032	291277	144267	197501
10	142650	285951	125215	205954
Average	140070	291832	138311	203130

Πίνακας: 2.2.1

Όπως παρατηρούμε στο παραπάνω σχήμα υπάρχουν μεγάλες διαφορές, εκτός από την JSON.simple και την Jackson οι οποίες είναι πολύ κοντά ,με την Jackson να ναι λίγο πιο γρήγορη.

Ας περάσουμε στα αποτελέσματα για τα μικρά αρχεία που φαίνονται στον παρακάτω πίνακα:

SMALL FILE (86kb - 10000 rounds/time) (ms)				
	JSON.simple	GSON	JACKSON	JSONP
01.json	1131	967	2938	1837
02.json	1230	1018	3550	2167
03.json	1581	1143	3082	1926
04.json	1098	1014	2917	1821
05.json	1121	1011	2939	1842
06.json	1131	934	2936	1783
07.json	1144	956	2959	1817
08.json	1123	1004	2954	1817
09.json	3424	3347	3329	3314
10.json	3406	3353	3358	3337
11.json	3403	3347	3354	3360
12.json	3405	3334	3385	3339
13.json	3408	3365	3370	3363
14.json	3402	3334	3353	3354
15.json	3430	3310	3358	3352
16.json	3401	3329	3341	3330
17.json	3394	3346	3346	3342
18.json	3404	3334	331	3349
19.json	3429	3342	3352	3339
20.json	1145	933	2951	1761
Average	2410	2286	3180	2677

Πίνακας: 2.2.2

Όπως παρατηρούμε στο παραπάνω σχήμα ο νικητής για τα μικρά αρχεία είναι η GSON με μικρή διαφορά από την JSON.simple, ενώ βλέπουμε ότι η Jackson στα μικρά αρχεία έρχεται τελευταία.

Αν συγκρίνουμε και τους δύο πίνακες καταλήγουμε στο συμπέρασμα ότι:

- Αν θέλουμε να δουλέψουμε με μεγάλα αρχεία η Jackson είναι η πιο κατάλληλη μιας και η GSON φαίνεται να δυσκολεύεται.
- Αν θέλουμε να δουλέψουμε με μικρά αρχεία η GSON είναι η πιο κατάλληλη και φαίνεται ότι η Jackson παίρνει την θέση που κατείχε η GSON οταν επρόκειτο για μεγάλα αρχεία.
- Τέλος, βλέπουμε ότι η JSON.simple ήρθε δεύτερη και στις δύο περιπτώσεις, πράγμα που σημαίνει ότι αν έχουμε να κάνουμε και με μικρά αλλά και με μεγάλα αρχεία η πιο κατάληλλη βιβλιοθήκη είναι η JSON.simple.

Προφανώς όμως όταν θέλουμε να διαλέξουμε ποια βιβλιοθήκη θέλουμε να χρησιμοποιήσουμε δεν παίζει ρόλο μόνο το parsing speed αλλά και άλλα χαρακτηριστικά που είδαμε παραπάνω και θα αναλύσουμε παρακάτω.

Conclusion

Ας δούμε για αρχή έναν περιληπτικό πίνακα για τις βιβλιοθήκες:

	User-friendly	Understandability	Data-types	JSON object's alignment	Features	Community	Sources
mJSON	✓	✓	✓				
JSON.simple	✓	✓	✓	✓	✓		
JSON-P	✓	✓	✓	✓	✓		
GSON	✓	✓	✓	✓	✓	✓	✓
Jackson		✓	✓	✓	✓	✓	✓

Πίνακας: 2.2.3

Αρχίκα απορρίπτουμε την mJson επειδή δεν μπορεί να διαχειριστεί αρχεία με πολλά Json objects και έλλειψη στοίχισης, δηλαδή σε κάθε γραμμή του αρχείου υπάρχει ένα Json object, το οποίο συμβαίνει στην πλειονότητα αλλά όχι σε όλα τα αρχεία. Έπειτα, απορρίπτουμε την JSON-P και την JSOM.simple λόγω έλλειψης πηγών και αλληλεπίδρασης μεταξύ χρηστών στο διαδίκτυο όταν προκύπτει κάποιο πρόβλημα. Συγκρίνοντας τις δύο μεγαλύτερες βιβλιοθήκες, δηλαδή την Jackson και την Gson παρατηρούμε για αρχή ότι η Jackson δεν είναι user friendly και έχει αρκετά μεγάλο μέγεθος με αποτέλεσμα να έχει περισσότερα features. Ενώ παρουσιάζουν ομοιότητες, χρησιμοποιώντας τους όρους serialization και deserialization, η Jackson καταλήψαμε ότι υπερτερεί. Αρχικά η GSON επιστρέφει τύπους δεδομένων όπως JsonObject για τα JSON objects, JsonPrimitive για τιμές όπως String, Integer, Double, JsonArray για πίνακες και JsonNull για τιμές null ενώ η Jackson τα αναλύει λίγο περισσότερο επιστρέφοντας τύπους όπως TextNode για τιμές τύπου String, ObjectNode για JSON objects, ArrayNode για πίνακες, NullNode για τιμές null, IntNode για τιμές Integer και άλλους πολλούς τύπους. Δεδομένο που θα μας χρησιμεύσει και θα μας βοηθήσει στην λειτουργικότητα της παρούσας διπλωματικής εργασίας. Η GSON όπως είπαμε έχει μία μέθοδο για to/from JSON objects ενώ η Jackson έχει δύο μεθόδους που μπορεί κάποιος να μετατρέψει JSON objects σε Java objects ή το ανάποδο τις Mapper και JsonParser ή JsonGenerator. Η Jackson υποστηρίζει και tree model που η GSON δεν το υποστηρίζει.

Τέλος, παρατηρώντας τα αποτελέσματα του benchmark καταλήγουμε ότι η Jackson είναι πιο γρήγορη σε μεγάλα αρχεία από την GSON ενώ σε μικρά αρχεία συμβαίνει το αντίθετο, με βάση όμως το ότι θα δουλέψουμε με μεγαλύτερα αρχεία και δίνοντας βάση και στα παραπάνω features που έχει η Jackson σε σύγκριση με την GSON, αποφασίζουμε ότι θα χρησιμοποιήσουμε την Jackson.

2.3 Ανάλυση απαιτήσεων

Use Case: CreateNewProject

Περιγραφή και Στόχος:

Στόχος της περίπτωσης χρήσης είναι η δημιουργία από το χρήστη ενός νέου project.

Ηθοποιοί:

Χρήστης

Βασική Ροή Γεγονότων:

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πληκτρολογεί στην αντίστοιχη φόρμα το όνομα του project που επιθυμεί.
2. Αν το όνομα του project δεν υπάρχει.
    3.1. Το σύστημα προχωρά στην δημιουργία ενός φακέλου με το όνομα του project που εισήγαγε.
3. Αν το όνομα του project υπάρχει ήδη.
    4.1.1. Το σύστημα εμφανίζει μήνυμα λάθους.
    4.1.2. Το σύστημα ζητάει από το χρήστη διαφορετικό όνομα για το project που επιθυμεί να δημιουργήσει.

Use Case: DeleteExistingProject

Περιγραφή και Στόχος:

Στόχος της περίπτωσης χρήσης είναι η διαγραφή από το χρήστη ενός project που υπάρχει ήδη.

Ηθοποιοί:

Χρήστης

Βασική Ροή Γεγονότων:

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης επιλέξει να διαγράψει κάποιο από τα ήδη υπάρχοντα project.

2. Το σύστημα εμφανίζει στον χρήστη τα πιθανά project για διγραφή.

3. Ο χρήστης επιλέγει project που επιθυμεί να διαγράψει.

Use Case : LoadData

Περιγραφή και Στόχος:

Στόχος της περίπτωσης χρήσης είναι η εισαγωγή από το χρήστη του αρχείου με τα δεδομένα τύπου JSON.

Ηθοποιοί:

Χρήστης

Βασική Ροή Γεγονότων:

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πληκτρολογεί στην αντίστοιχη φόρμα το όνομα του αρχείου με τα δεδομένα που επιθυμεί να εισάγει προς επεξεργασία.

2. Αν το αρχείο υπάρχει:

    2.1. Το σύστημα προχωρά στην επξεργασία των δεδομένων, δηλαδή στον έλεγχο των JSON objects για διαφορές και εξαγωγή των εκδόσεων.

3. Αν το αρχείο δεν υπάρχει.

    3.1. Το σύστημα εμφανίζει μήνυμα λάθους.

3.2. Το σύστημα ζητάει από το χρήστη εκ νέου αρχείο που υπάρχει.

Use Case : SaveProject

Περιγραφή και Στόχος:

Στόχος της περίπτωσης χρήσης είναι η αποθήκευση του project.

Ηθοποιοί:

Χρήστης

Βασική Ροή Γεγονότων:

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης αποφασίσει να αποθηκεύσει το project.
2. Το σύστημα στον φάκελο που έφτιαξε με το όνομα που επέλεξε ο χρήστης όταν δημιούργησε νέο project, αποθηκεύει το αρχείο εισόδου με τα δεδομένα, τις διαφορετικές εκδόσεις σχημάτων που βρέθηκαν και αρχείο με τα χαρακτηριστικά εξέλιξης του σχήματος .
3. Η περίπτωση χρήσης τερματίζεται.

Use Case : ShowSchemaVersions

Περιγραφή και Στόχος:

Στόχος της περίπτωσης χρήσης είναι η εμφάνιση στο χρήστη των διαθέσιμων εκδοχών των σχημάτων που βρέθηκαν μετά την επεξεργασία του αρχείου με τα δεδομένα.

Ηθοποιοί:

Χρήστης

Βασική Ροή Γεγονότων:

1. Η περίπτωση χρήσης ξεκινά εμφανίζοντας στο χρήστη τις διαθέσιμες εκδόσεις.
2. Ο χρήστης επιλέγει μία από τις διαθέσιμες εκδόσεις για οπτικοποίηση.
    2.1. Το σύστημα ανοίγει την εκδοσή που επέλεξε, δείχοντας στο χρήστη το σχήμα.

Use Case : ViewSchemaFeatures

Περιγραφή και Στόχος:
Στόχος της περίπτωσης χρήσης είναι η εμφάνιστη στο χρήστη των χαρακτηριστικών που παρουσιάζει η εξέλιξη του σχήματος.

Ηθοποιοί:
Χρήστης

Βασική Ροή Γεγονότων:

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης επιλέξει να δει τα χαρακτηριστικά που εμφανίζει η εξέλιξη του σχήματος.
2. Αν υπάρχει εξέλιξη στο σχήμα.

2.1. Το σύστημα εμφανίζει στο χρήστη σε κείμενο τα χαρακτηριστικά της εξέλιξης.

3. Αν δεν υπάρχει εξέλιξη στο σχήμα.

3.1. Το σύστημα εμφανίζει μήνυμα ότι δεν υπήρχαν διαφορετικές εκδοχές στο σχήμα.

Κεφάλαιο 3. Σχεδίαση & Υλοποίηση

3.1 Ορισμός προβλήματος και αλγόριθμοι επίλυσης

Αν πρόκειται για θέμα που προσανατολίζεται στην αλγοριθμική επίλυση ενός προβλήματος (αντί π.χ., για την κατασκευή ενός εργαλείου που παρέχει μια λειτουργικότητα) η παρούσα υποενότητα έχει τον πλέον σημαντικό ρόλο. Ενδεχομένως να χρειαστεί μία υποενότητα για κάθε ένα από τα παρακάτω.

- Τυπικός ορισμός του προβλήματος.
- Περιγραφή των χρησιμοποιούμενων δομών δεδομένων.
- Αλγόριθμος / πρωτόκολλο / μέθοδος / … που χρησιμοποιείται για την επίλυση του προβλήματος.

3.2 Σχεδίαση και αρχιτεκτονική λογισμικού

- Περιγραφή των components της αρχιτεκτονικής του προς υλοποίηση συστήματος, του τρόπου με τον οποίο συνεργάζονται και των αλληλοεξαρτήσεών τους.
- Περιγραφή του μοντέλου του λογισμικού μέσω διαγραμμάτων UML κλάσεων. Οι βασικές κλάσεις χρήζουν σχολιασμού και επεξηγήσεων.
- Περιγραφή του προκύπτοντος σχεσιακού σχήματος [αν υπάρχει τέτοιο] και αν χρειάζεται και του αντίστοιχου μοντέλου Οντοτήτων/Εξαρτήσεων.
- Περιγραφή σημαντικών sequence diagrams [αν είναι σημαντικά]

– Επιπλέον, αν χρειάζεται, επεξηγούνται οι υπογραφές / API's των διεπαφών, προγραμμάτων, ή συναρτήσεων, μαζί με όποιες λεπτομέρειες κρίνονται αναγκαίες.

3.3 Σχεδίαση και αποτελέσματα ελέγχου του λογισμικού

Περιγράφονται οι στόχοι του ελέγχου, το πώς οι στόχοι σχετίζονται με τα σχετικά use cases (π.χ., εξηγώντας ένα traceability matrix), ποια τα unit & system tests. Περιγραφή των test fixtures.

Η διενέργεια του ελέγχου του λογισμικού περιγράφεται και τα αποτελέσματά της παρατίθενται συνοπτικά.

3.4 Λεπτομέρειες εγκατάστασης και υλοποίησης

Στην ενότητα αυτή περιγράφονται τα χαρακτηριστικά της συγκεκριμένης υλοποίησης, όπως η πλατφόρμα ανάπτυξης και εκτέλεσης, τα προγραμματιστικά εργαλεία, οι απαιτήσεις της εφαρμογής σε hardware, κ.λ.π.

Είναι σημαντικό να παρατεθούν με συγκροτημένο τρόπο οι απαραίτητες ρυθμίσεις ώστε το πρόγραμμα να εγκατασταθεί και να εκτελείται σωστά. Αυτό δεν αφορά μόνο τις ρυθμίσεις του προγραμματιστικού περιβάλλοντος, αλλά π.χ., και τι πρέπει να μπει σε κάποια αρχεία αρχικοποίησης του συστήματος κ.ο.κ.

3.5 Επεκτασιμότητα του λογισμικού

Όταν σχεδιάζουμε το λογισμικό σκεφτόμαστε και πώς θα επεκταθεί και συντηρηθεί στο μέλλον. Άρα κάνουμε και μια λίστα από πιθανές επεκτάσεις. Στην ενότητα αυτή περιγράφονται τα σημεία του κώδικα τα οποία θα χρειαστεί να αλλαχθούν σε μελλοντικές τέτοιες επεκτάσεις. Επίσης σημειώνουμε τα σημεία που έχουν hard-coded λειτουργικότητες που θα πρέπει να συντηρηθούν στο μέλλον.

Κεφάλαιο 4. Πειραματική Αξιολόγηση

Στην ενότητα αυτή παρουσιάζουμε αναλυτικά την πειραματική αξιολόγηση της μεθόδου μας.

4.1 Μεθοδολογία πειραματισμού

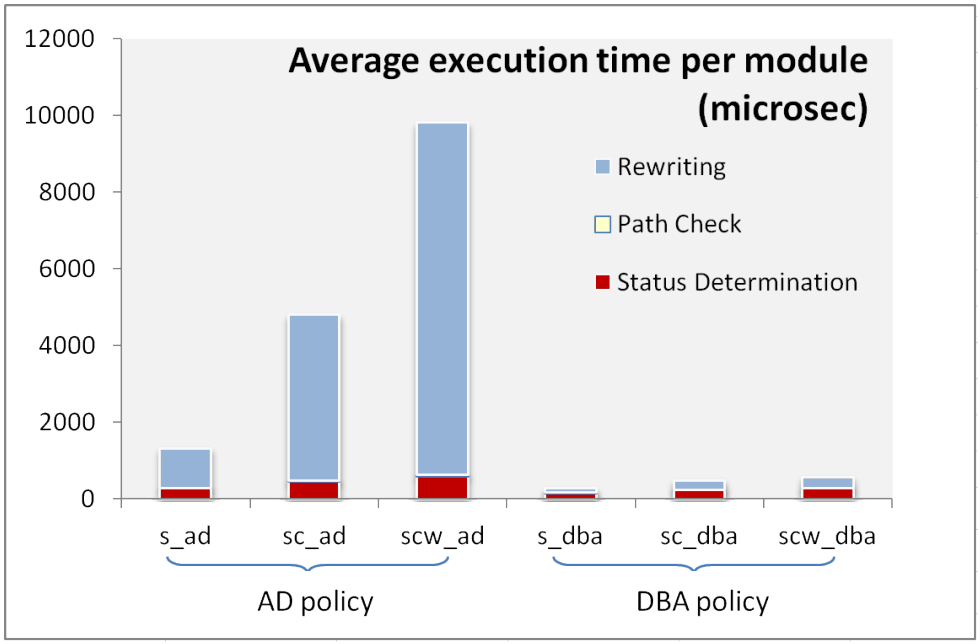
Περιγραφή του σκοπού των πειραμάτων

Δεδομένα που χρησιμοποιήθηκαν, αντίπαλοι αλγόριθμοι και τεχνικές (ή ότι άλλο είναι αρμόζον για την περίσταση)

Περιγραφή του περιβάλλοντος στο οποίο διεξήχθησαν τα πειράματα.

4.2 Αναλυτική παρουσίαση αποτελεσμάτων

Για κάθε παράμετρο που μας ενδιαφέρει να μετρήσουμε, αναλύουμε τα αποτελέσματα του σχετικού πειράματος. Περιγράφουμε τι μεταβάλλαμε, τι μετρήσαμε και τι συμπεράσματα βγαίνουν από το πείραμα. Συνιστάται η παράθεση διαγραμμάτων και δεδομένων από τα διεξαχθέντα πειράματα και οπωσδήποτε ο σχετικός σχολιασμός.



Εικόνα 4.1 Χρόνος εκτέλεσης (microsec) ως συνάρτηση (α) της κατανομής αρμοδιοτήτων και (β) της πολιτικής διαχείρισης της διάδοσης αλλαγών. Εσωτερικά σε κάθε στήλη αναπαρίστανται(i) ο χρόνος επανεγγραφής, (ii) ο χρόνος ελέγχου μονοπατιών διάδοσης και (iii) ο χρόνος αποτίμησης της κατάστασης κόμβων.

Η γενική ιδέα είναι ότι περιγράφουμε πώς μεταβάλλονται οι τιμές στον κάθετο άξονα, σε σχέση με τις τιμές στον οριζόντιο άξονα ή/και τις διάφορες τεχνικές που εμπλέκονται στο πείραμα. Πάντα αναφέρουμε τις μονάδες. Η ασφαλής λύση είναι να χρησιμοποιήσετε bar charts. Το παράδειγμα στην Εικόνα 4.1 που σας δίνεται είναι όσο πιο πολύπλοκο θα μπορούσε να είναι.

Για τους πίνακες: στο οπτικό αποτέλεσμα, ελαχιστοποιήστε τα pixels που δεν είναι περιεχόμενο. Αυτό αφορά κυρίως τα borders αλλά και το χρώμα στο background (το πολύ πολύ να μπει ένα απαλό φόντο σε όλο τον πίνακα, όπως π.χ., έχουμε στην Εικόνα 2.1). Η έμφαση πρέπει να είναι στο να αναδειχθεί το περιεχόμενο του πίνακα. Προσέξτε, οι γραμμές του πίνακα, να έχουν στην παράγραφό τους “Keep with next” ώστε ο πίνακας να μη σπάει σε διαφορετικές σελίδες.

Breakdown of Tables over their Activity Class
(Percentages over Total #Tables)

		Activity Class			Activity Class (%)		
	Total #Tables	RIGID	QUIET	ACTIVE	RIGID	QUIET	ACTIVE
Atlas	88	18	43	27	***20%***	**49%**	31%
BioSQL	45	16	13	16	**36%**	***29%***	**36%**
Castor	91	57	31	3	**63%**	34%	***3%***
SlashCode	68	15	38	15	***22%***	**56%**	***22%***
Zabbix	56	23	30	3	41%	**54%**	***5%***

Πίνακας 4.1 Κατανομή πινάκων σε διαφορετικές κλάσεις δραστηριότητας σε απόλυτες τιμές και ποσοστά (για κάθε σύνολο δεδομένων, η μέγιστη τιμή με ***κόκκινα έντονα*** *γράμματα και η ελάχιστη με μπλε πλαγιαστά)*

Συνιστάται η συντήρηση ενός ΚΑΘΑΡΟΥ spreadsheet, είτε συνολικά για την εργασία, είτε ανά πείραμα, και η παράδοσή του στο τέλος, μαζί με τον κώδικα, τα δεδομένα εισόδου και εξόδου, και την παρούσα αναφορά.

Κεφάλαιο 5. Επίλογος

Στην ενότητα αυτή συνοψίζουμε τη συνεισφορά και τα αποτελέσματα της εργασίας και παραθέτουμε σκέψεις για μελλοντικές επεκτάσεις της.

5.1 Σύνοψη και συμπεράσματα

Στην ενότητα αυτή συνοψίζουμε τα αποτελέσματα τις διπλωματικής (αντιγράφουμε την ενότητα 1.1 χρησιμοποιώντας αόριστο αντί για μέλλοντα χρόνο) και περιγράφουμε λίγο πιο αναλυτικά και τα όποια συμπεράσματα εξάγαμε.

5.2 Μελλοντικές επεκτάσεις

Στην ενότητα αυτή δίνουμε μια λίστα με πράγματα που έχει νόημα / ενδιαφέρον να φτιαχτούν στο μέλλον.

Βιβλιογραφία

[BBC+99] P.A. Bernstein, T. Bergstraesser, J. Carlson, S. Pal, P. Sanders, D. Shutt. Microsoft Repository Version 2 and the Open Information Model. Information Systems 24(2), pp. 71-98, 1999.

[BaCR94] V.R. Basili, G.Caldiera, H.D. Rombach. The Goal Question Metric Approach. Encyclopedia of Software Engineering, pp. 528-532, John Wiley & Sons, Inc, 1994. Also available at http://www.cs.umd.edu/users/basili/papers.html

[Dean97] E.B. Dean. Quality Functional Deployment from the Perspective of Competitive Advantage. Available at http://mijuno.larc.nasa.gov/dfc/qfd.html

[JJQV98] M. Jarke, M.A. Jeusfeld, C. Quix, P. Vassiliadis. Architecture and quality in data warehouses. In Proc. 10th Conference on Advanced Information Systems Engineering (CAiSE '98), pp. 93-113, Pisa, Italy, June 1998.

[JaVa97] M. Jarke, Y. Vassiliou. Foundations of data warehouse quality – a review of the DWQ project. In Proc. 2nd Intl. Conference Information Quality (IQ-97), pp. 299-313, Cambridge, Mass., USA, June 1997.

[Orr98] K. Orr. Data quality and systems theory. In Communications of the ACM, 41(2), pp. 54-57, Feb. 1998.