MongoDB
&
Mongoose

Εισαγωγή

- Η MongoDB αποτελεί μια μη σχεσιακή βάση δεδομένων NoSQL
- Ενώ οι σχεσιακές βάσεις δεδομένων είναι ευρέως διαδεδομένες, δεν είναι κατάλληλες να διαχειρίζονται τον συνεχώς αυξανόμενο όγκο των δεδομένων
- Οι μη σχεσιακές βάσεις δεδομένων έρχονται για να «σώσουν» την κατάσταση με πολλούς διαδικτυακούς ιστότοπους να τις αξιοποιούν όπως η Amazon, η Google, το Netflix και το Facebook οι οποίες εξαρτώνται από μεγάλο όγκο δεδομένων
- Οι NoSQL βάσεις δεδομένων μπορούν να λειτουργήσουν αποτελεσματικά **με μη δομημένα δεδομένα**, όπως αυτά από τα κοινωνικά δίκτυα, το ηλεκτρονικό ταχυδρομείο και διάφορα έγγραφα.

Εισαγωγή

- Μια βάση δεδομένων MongoDB περιέχει μια ή περισσότερες Συλλογές ή «Collections»
- Σε περίπτωση που μέχρι τώρα χρησιμοποιούσατε σχεσιακές βάσεις δεδομένων, μπορείτε να σκεφτείτε μια **συλλογή** ως ένα **πίνακα**!
- Κάθε συλλογή περιέχει τα **documents** δηλαδή τα δεδομένα της εγγραφής σε μορφή value-pair
- Σκεφτείτε το σαν **row σε ένα table**

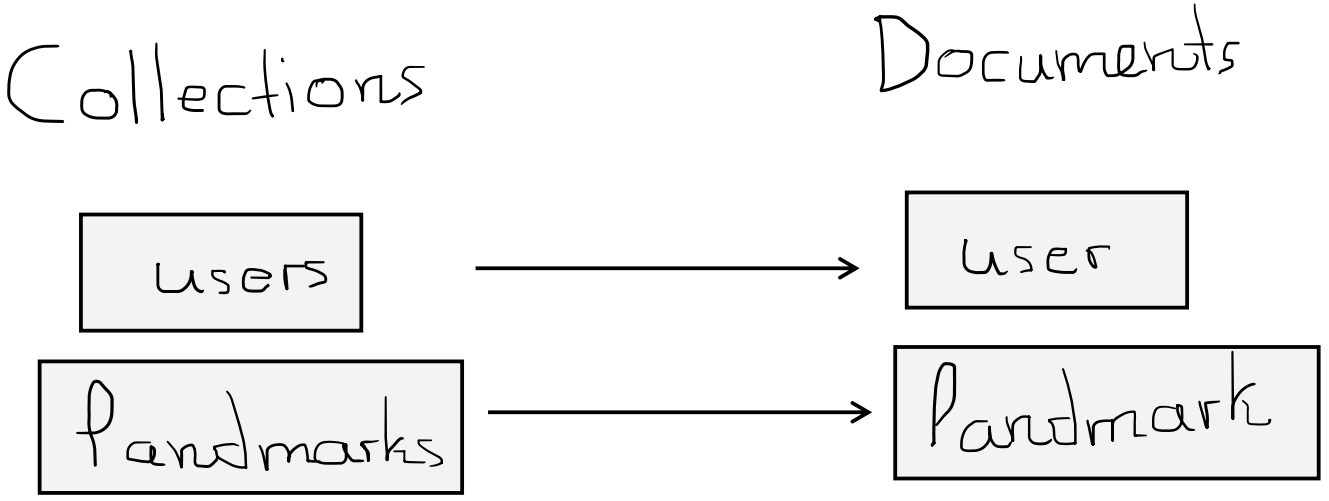
MongoDB

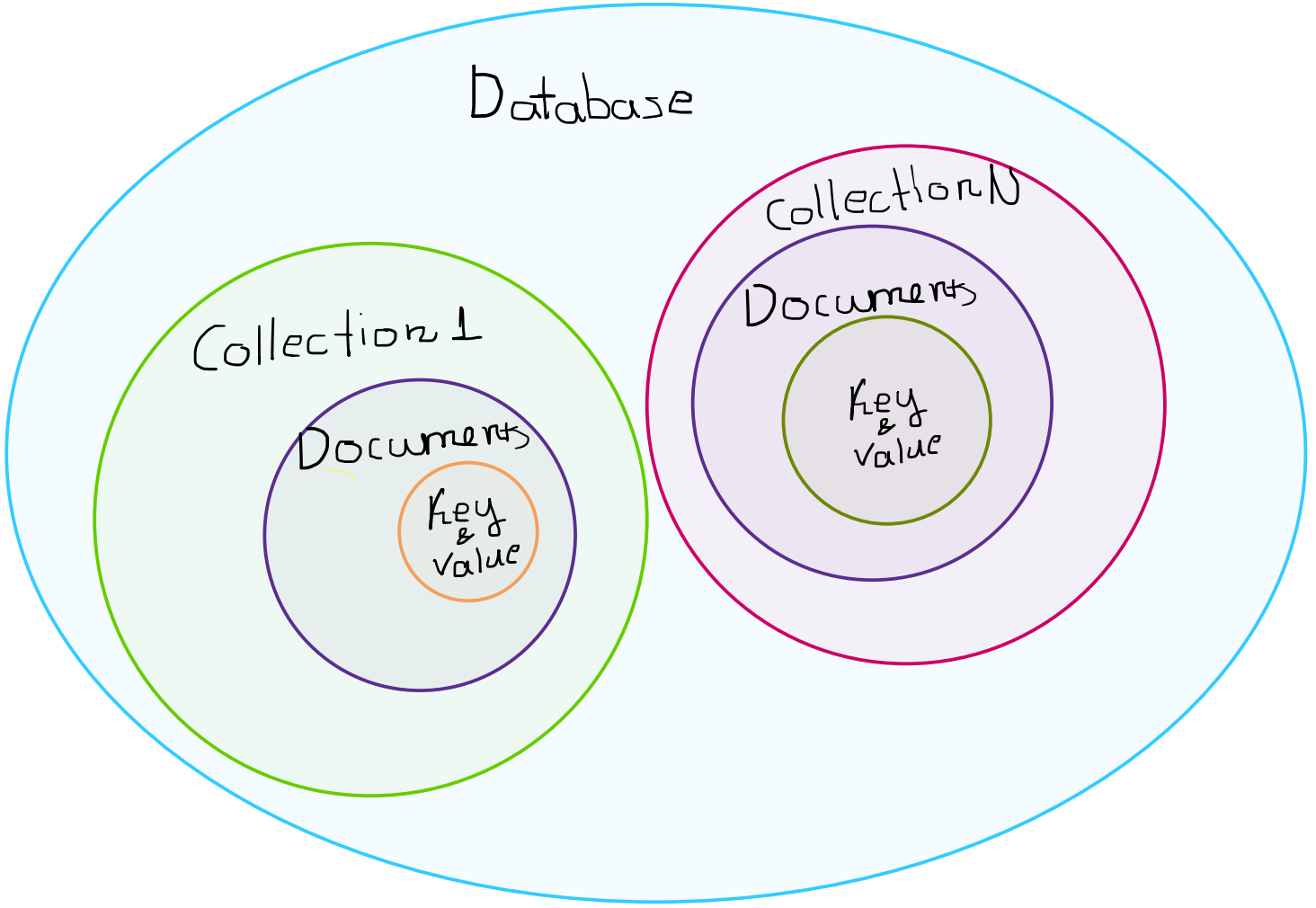
- Η MongoDB είναι μία document-oriented database

- Collections σαν tables
- Documents σαν rows

- Ένα έγγραφο(document) περιλαμβάνει την αποθήκευση μια εγγραφής, για παράδειγμα για ένα χρήστη ή ένα αξιοθέατο

MongoDB




Database
Collection 1
Documents
Key & Value
CollectionN
Documents
Key & Value

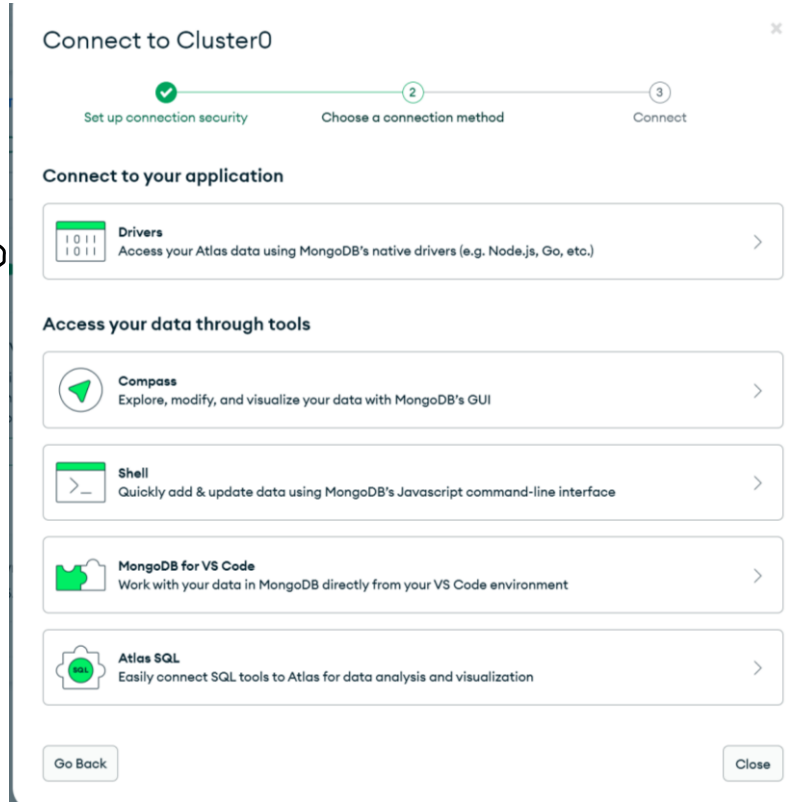
Εισαγωγή

- Θα δούμε τώρα πως μπορούμε να συνδέσουμε μια βάση δεδομένων mongoDB στην εφαρμογή μας και πως να κάνουμε διάφορες ενέργειες με αυτή

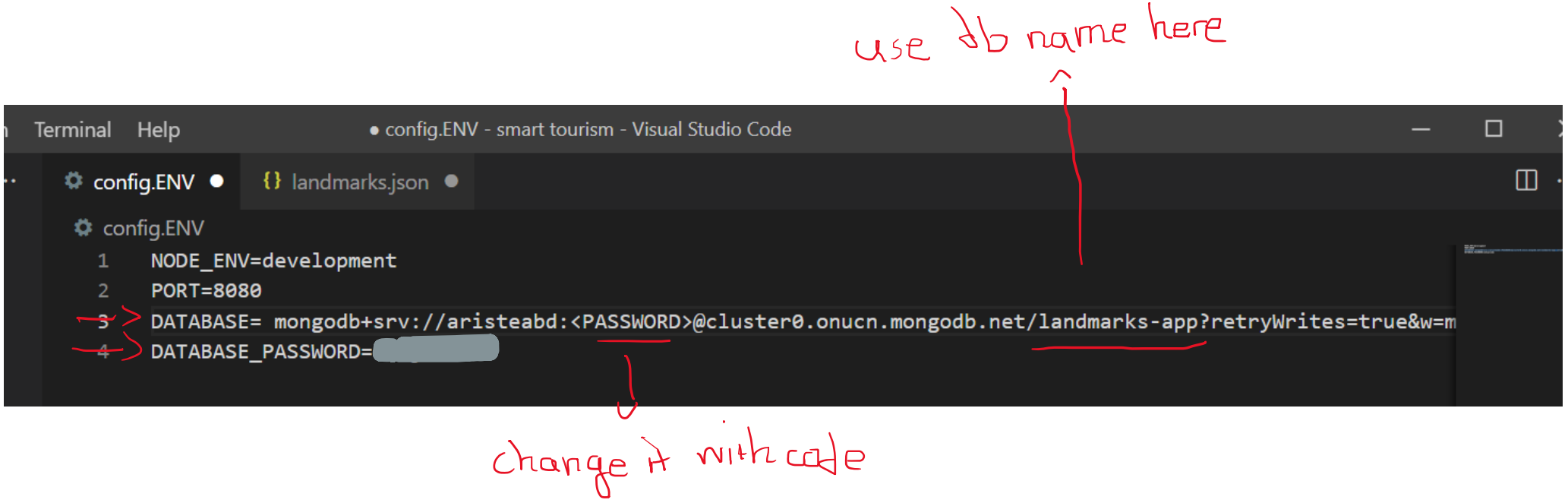
- Θα δούμε ακόμη μια βιβλιοθήκη : Mongoose : https://mongoosejs.com/

Connect to db

- Εδώ χρησιμοποιώ το cloud Atlas mongoDB
- click drivers-> copy connection string



Config.env for remote db



Mongodb

- Το επόμενο βήμα είναι να εγκαταστήσουμε ένα λογισμικό που μας επιτρέπει μέσα από την node.js εφαρμογή μας να επικοινωνούμε και να αλληλοεπιδρούμε με την mongodb βάση δεδομένων

- Υπάρχουν διάφορες επιλογές
  - Mongoose
  - Mongoskin
  - Mongoskin
  - Mongolian

Mongodb

https://www.npmtrends.com/mongolia-vs-mongoose-vs-mongoskin

Mongoose

- Η Mongoose αποτελεί μια Object Data Modeling (ODM) βιβλιοθήκη βασισμένη στο Node.js για τη MongoDB.
- Ένα ODM είναι ένα μοντέλο δεδομένων που αντιμετωπίζει τα σύνολα δεδομένων ως "αντικείμενα", εκχωρώντας ιδιότητες και τιμές
- Έτσι είναι ευκολότερος ο χειρισμός τους
- Με απλά λόγια, η Mongoose επιτρέπει στους προγραμματιστές να χρησιμοποιούν την MongoDB πολύ ευκολότερα

Mongoose

- mongoose **schema** : μοντελοποιούμε τα δεδομένα, περιγράφοντας τη δομή τους, πιθανές default τιμές κλπ.

- mongoose **model**: Στην συνέχεια χρησιμοποιούμε αυτό το «σχήμα» και δημιουργούμε ένα μοντέλο, μια διεπαφή δηλαδή με τη βάση για **crud** λειτουργίες

Connect to db

- Install mongoose in project : npm install mongoose
- Server.js
- Use module mongoose
  - const mongoose = require('mongoose’);
- Define variable for db and use password in it
  - const mydb = process.env.DATABASE.replace('<PASSWORD>',process.env.DATABASE_PASSWORD);

https://www.npmjs.com/package/mongoose

```
mongoose
  .connect(mydb, {
    useNewUrlParser: true,
    useCreateIndex: true,
    useFindAndModify: false,
    useUnifiedTopology: true
  })
  .then(() => console.log('DB connection successful!'));
```

A query also has a .then() function-> can be used as a promise.

- These are some options for dealing with deprecation warnings
- Περισσότερα στο https://mongoosejs.com/docs/connections.html
- //process.env.DATABASE_LOCAL αντι για mydb για σύνδεση με τοπική βάση δεδομένων

Εισαγωγή

- Θα δούμε τώρα πως στη πράξη πως μπορούμε να φτιάξουμε ένα mongoose shema & mongoose model για την εφαρμογή μας.

- Mongoose Schema-> ορίζει το ***σχήμα*** των **εγγράφων** (documents) μέσα σε μια συλλογή (collection)

- mongoose model -> τα μοντέλα δρουν σαν *σχεδιαγράμματα ή* ***κλάσεις*** που χρησιμοποιούμε για να **δημιουργούμε documents**

- Τα **Mongoose models** είναι υπεύθυνα για την **αναζήτηση**, τη **δημιουργία**, την **ενημέρωση** και την **αφαίρεση** εγγράφων από τη βάση δεδομένων MongoDB.

Εισαγωγή

- Άρα χρησιμοποιούμε τα **μοντέλα** για να :
  - Δημιουργούμε documents
  - Για να **αναζητούμε**, **ενημερώνουμε**, **διαγράφουμε** αυτά τα documents από τη βάση δεδομένων.

Mongoose models are created from schemas

Δημιουργούμε “Σχήμα”

- Παραδείγματα

```
//use javascript data types
const landmarksSchema = new mongoose.Schema({
  type: String,
  name: String,
  description: String,
  ratingsAverage: Number,
  ratingsQuantity: Number
  });
```

```
  //define more options about a schema
const landmarksSchema = new mongoose.Schema({
  type: {
    type: String,
    lowercase: true,
    required: true
  },
  name: String,
  description: String,
  ratingsAverage: Number,
  ratingsQuantity: Number
  });
```

https://mongoosejs.com/docs/guide.html

All Schema Types

- Υπάρχουν πολλές επιλογές τις οποίες είναι καλό να γνωρίζουμε:

The permitted SchemaTypes are:

- String
- Number
- Date
- Buffer
- Boolean
- Mixed
- ObjectId
- Array
- Decimal128
- Map

Mongoose plugins : μπορούν να προσθέσουν custom SchemaTypes όπως int32.

- SchemaTypes : έχουν επιπλέον τις παρακάτω επιλογές.

- default- Determines the default value
- required: {Boolean}

All Schema Types

- Υπάρχουν πολλές επιλογές τις οποίες είναι καλό να γνωρίζουμε:

- https://mongoosejs.com/docs/schematypes.html

Παραδείγματα

```
const blogSchema = new Schema({
   title:  String
   author: String,
   body:   String,
   comments: [{ body: String, date: Date }],
   date: { type: Date, default: Date.now },
   hidden: Boolean
});
```

1 δημιουργούμε «Σχήμα»

```
//use javascript data types
const landmarksSchema = new mongoose.Schema({
  type:{
    type: String,
    required:[true, "A landmark must be landmark or musuem"]// specify error string
  },
  name: String,
  description: String,
  ratingsAverage: Number,
  ratingsQuantity: Number
  });
```

2 δημιουργούμε «Μοντέλο»

```
//schema
const landmarkSchema = new mongoose.Schema({
  type:{
    type: String,
    required:[true, "A landmark must be landmark or musuem"]// specify error string
  },
  name: String,
  description: String,
  ratingsAverage: Number,
  ratingsQuantity: Number
  });
//model out of  schema
const Landmark = mongoose.model("Landmark",landmarkSchema);
```

- `const Landmark = mongoose.model("Landmark",landmarkSchema);`

- Η πρώτη παράμετρος είναι το **μοναδικό όνομα της συλλογής (collection)** για την οποία δημιουργούμε ένα μοντέλο

- Η δεύτερη παράμετρος είναι ο ορισμός του σχήματος που είδαμε παραπάνω

Το όνομα της συλλογής που θα αποθηκευτεί στη MongoDB θα είναι `landmarks`. Άρα όλα μικρά και πληθυντικός

3 δημιουργούμε «Έγγραφο»

```
});
//model out of  schema
const Landmark = mongoose.model("Landmark",landmarkSchema);

//lets create a new document out of the model we have builled above
const testLandmark= new Landmark({
  type: "landmark",
  name: "Acropolis",
  description: "The Acropolis is the strongest and most important monument of
  ratingsAverage: 4.9,
  ratingsQuantity: 97
});
```

4 αποθηκεύουμε το «Έγγραφο»

- Με το που σώσουμε το πρώτο έγγραφο, αν **δεν υπάρχει η συλλογή στη βάση δημιουργείται**

```
//save it to Landmarks collection and we have also access to it
testLandmark.save().catch(err=>{
  console.log("ERROR: "+ err);
});
```

//call the **save method ->** save the document to the database.
//save here will return a promise -> we can then consume this promise

Note that…

- Αν δημιουργήσουμε ένα post request με ένα πεδίο που δεν είναι ορισμένο στο «σχήμα», απλά θα το αγνοήσει και θα δημιουργήσει ένα νέο έγγραφο με όλα τα πεδία που υπάρχουν στο «σχήμα».

- Με άλλα λόγια δε θα βγάλει error
- Θα δημιουργήσει το document

To be continued…

https://www.mongodb.com/docs/atlas/driver-connection/?tck=docs_driver_nodejs