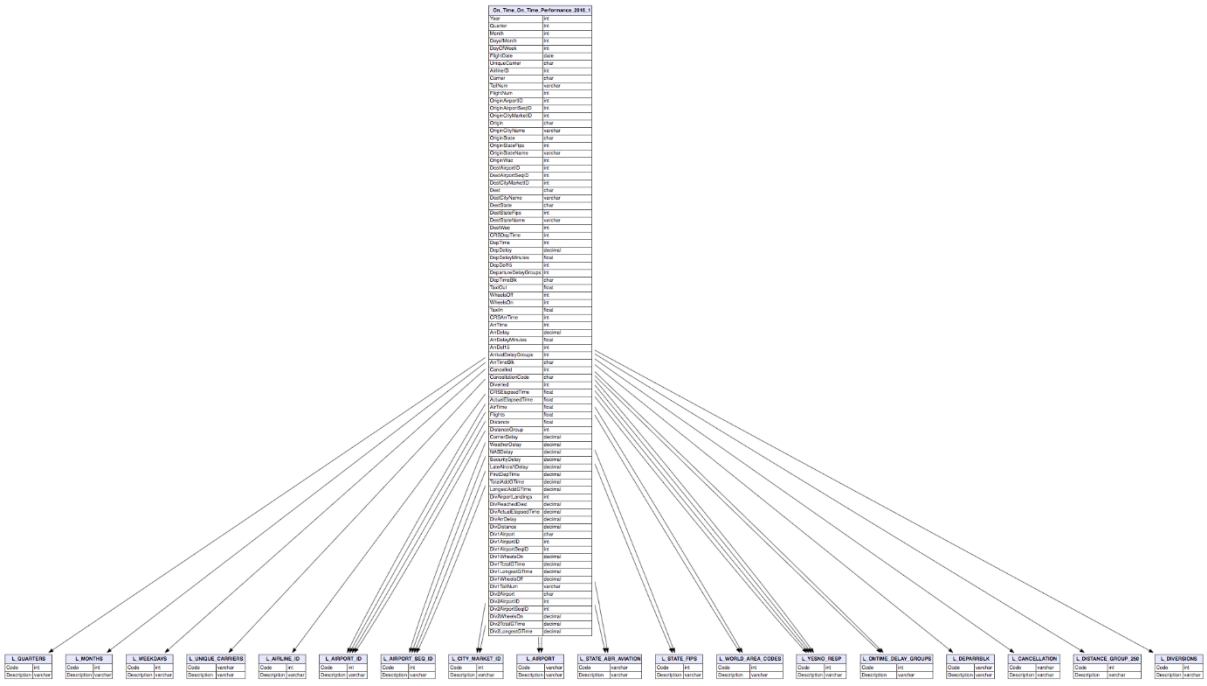
Développement d'application Cloud

Projet

1.1 Jeux de données

Nous avons choisi la base de données « Airline » parmi les différentes bases de données disponibles sur https://relational.fit.cvut.cz/search. Cette base de données est une base SQL contenant tous les vols aériens aux Etats-Unis de **Janvier 2017 à Mars 2017** et dont le schéma relationnel est défini comme suit :

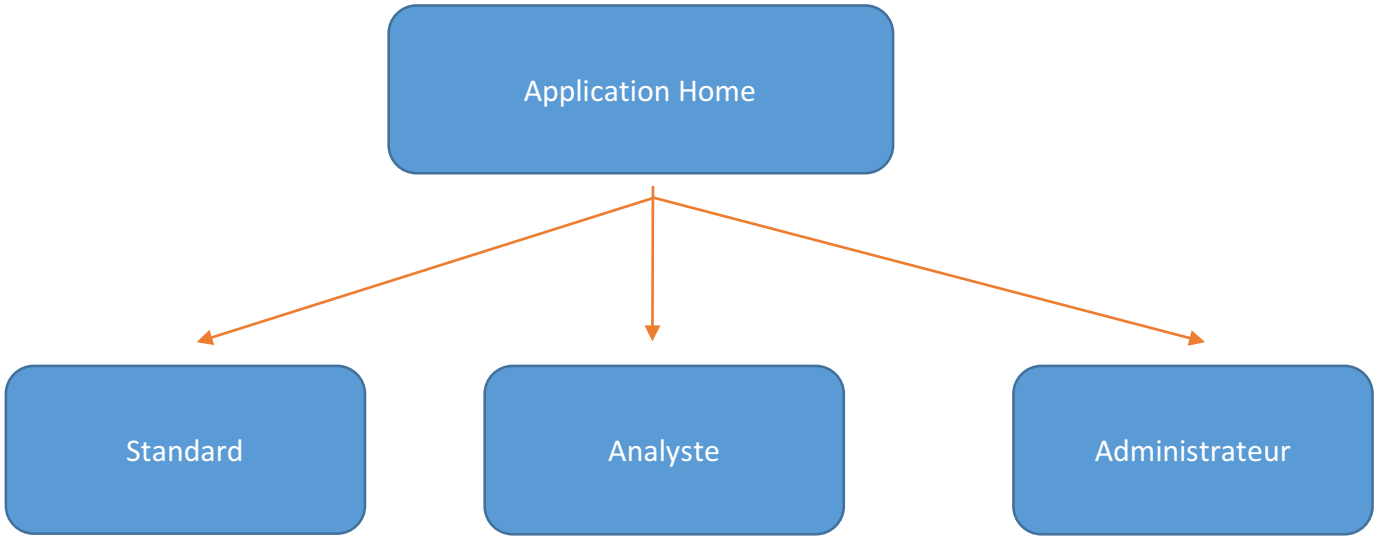


1.2 Spécification des besoins

[Rappel : La base de données comporte des données entre **Janvier 2017 et Mars 2017**]

1. Spécification des besoins pour l'application

L'application doit proposer 3 interfaces à l'utilisateur comme le montre le schéma ci-dessous :



2. Spécification des besoins pour les requêtes

L'application doit requêter la base de données afin de livrer des informations pertinentes à l'utilisateur. Dans le cadre de ce projet et en vue de la base de données choisie, les requêtes seront :

- Pour la partie « Standard » :
  - Obtenir la liste des vols par compagnie aérienne (paramètres : Compagnie, Date)
  - Obtenir la liste des vols entrants et sortants d'un aéroport (paramètres : Aéroport, Date)
  - Obtenir la liste des vols entre 2 aéroports (paramètres : Aéroport de départ, Aéroport d'arrivée, Date)

- Pour la partie « Analyste » :
  - Obtenir le pourcentage d'avoir un retard sur un vol en départ d'un aéroport (paramètre : Aéroport)
  - Obtenir le pourcentage d'avoir un retard sur un vol en arrivée d'un aéroport (paramètre : Aéroport)
  - Obtenir la moyenne des retards en minutes d'un vol en départ d'un aéroport (paramètre : Aéroport)
  - Obtenir la moyenne des retards en minutes d'un vol en arrivée d'un aéroport (paramètre : Aéroport)
  - Obtenir le top 10 des compagnies qui atterrissent le plus dans un aéroport (Paramètres : Aéroport)

- Obtenir le top 10 des compagnies qui décollent le plus d'un aéroport (Paramètres : Aéroport)

- Pour la partie « Administrateur » :

Récupération de l'historique des logs.
Chaque requête effectuée sur la base de donnée est stockée avec ses propriétés (date, temps de réponse (ms), utilisateur, requête).

Depuis la vue administrateur une requête « getLog » permet de récupérer l'historique complet.
De plus 3 tableaux regroupent par utilisateurs (Standard, Analyste, Administrateur) les requêtes ayants été effectuées, avec le temps moyen de réponse.

1.3 Dénormalisation

1.3.1 Schéma


```
{
    "_id" : ObjectId("5a2e90805498680c08fe6631"),
    "FL_DATE" : ISODate("2017-01-01T00:00:00.000+0000"),
    "UNIQUE_CARRIER" : "AA",
    "AIRLINE_ID" : NumberInt(19805),
    "CARRIER" : "AA",
    "FL_NUM" : NumberInt(307),
    "ORIGIN" : "DEN",
    "ORIGIN_CITY_NAME" : "Denver, CO",
    "ORIGIN_STATE_ABR" : "CO",
    "ORIGIN_STATE_NM" : "Colorado",
    "DEST" : "PHX",
    "DEST_CITY_NAME" : "Phoenix, AZ",
    "DEST_STATE_ABR" : "AZ",
    "DEST_STATE_NM" : "Arizona",
    "DEP_TIME" : NumberInt(1135),
    "DEP_DELAY" : -10.0,
    "DEP_DEL15" : 0.0,
    "WHEELS_OFF" : NumberInt(1153),
    "WHEELS_ON" : NumberInt(1321),
    "ARR_TIME" : NumberInt(1328),
    "ARR_DELAY" : -17.0,
    "ARR_DEL15" : 0.0,
    "CANCELLED" : 0.0,
    "DIVERTED" : 0.0,
    "AIR_TIME" : 88.0,
    "FLIGHTS" : 1.0,
    "DISTANCE" : 602.0,
    "CARRIER_DELAY" : null,
    "WEATHER_DELAY" : null,
    "NAS_DELAY" : null,
    "SECURITY_DELAY" : null,
    "LATE_AIRCRAFT_DELAY" : null,
    "AIRLINE_NAME" : "American Airlines Inc.: AA"
}
```

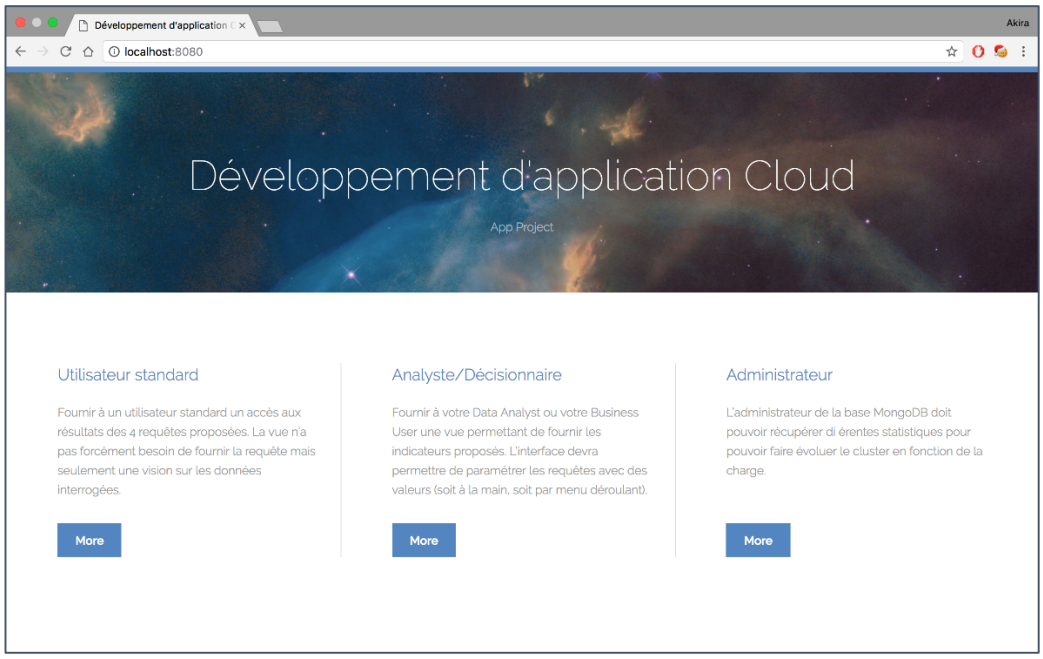
1.3.2 Transformation

Nous souhaitions rajouter le champ contenant le nom de la compagnie dans notre liste, celle-ci contenant déjà le code ID de la compagnie. Nous avons alors dû créer un nouveau champ « AIRLINE NAME », que nous remplissons pour chaque document avec le label correspondant en utilisant le code dans une table spécifiée. Voici le code qui a été exécuté pour effectuer cette modification :

```
db.FLIGHTS.update({},
{$set: {AIRLINE_NAME: null}},
{multi: true});

db.FLIGHTS.find().forEach(function (flightInfo) {

    var doc2 = db.L_AIRLINE_ID.findOne({ Code : flightInfo.AIRLINE_ID },
{ Description: 1 });

    if (doc2 != null) {
        flightInfo.AIRLINE_NAME = doc2.Description;
        db.FLIGHTS.save(flightInfo);
    }
});

Le client MongoDB nous a donc retourné le résultat suivant :
WriteResult({ "nMatched" : 1349131, "nUpserted" : 0, "nModified" : 1349131 })
```

1.4 Vues

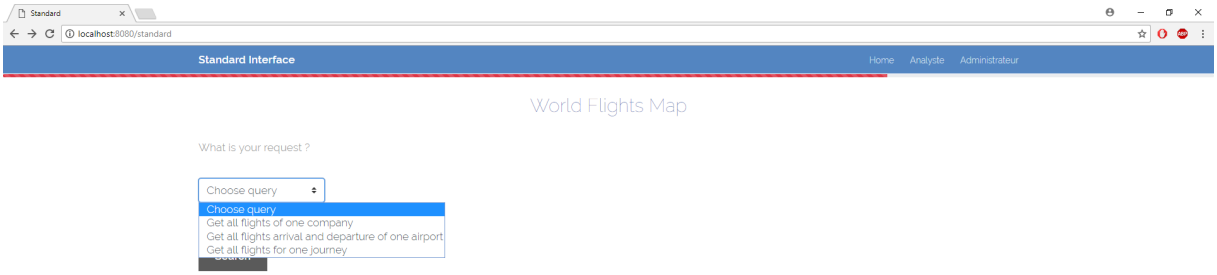
Vue Home (index.html)



Cette vue est la page d'accueil de l'application. Elle laisse à l'utilisateur le choix d'accéder au 3 différentes interfaces que propose l'application.

[Rappel : La base de données comporte des données entre **Janvier 2017 et Mars 2017**]

Vue Standard (standard.html)

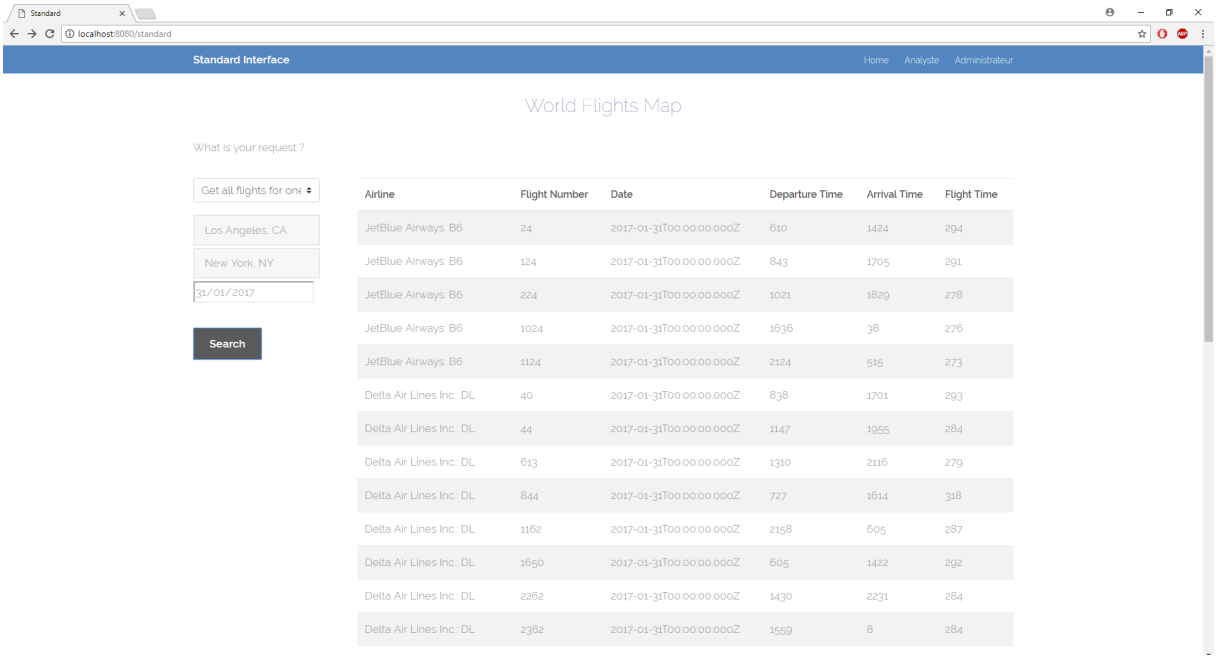


Cette vue est la vue affichant à l'utilisateur les résultats de requêtes de type « Standard ».

Veuillez attendre que la barre de chargement est disparue pour débuter les requêtes. Dans un premier temps, deux requêtes sont effectuées de manière transparente pour l'utilisateur : « company » et « airports » qui retournent la liste de toutes les compagnies aérienne et la liste de tous les aéroports.

Sur cette vue, 3 requêtes peuvent être effectuées par l'utilisateur :

- "Get all flights of one company"
- "Get all flights arrival and departure of one airport"
- "Get all flights for one journey"



En fonction de la requête sélectionnée, les champs à compléter par l'utilisateur changent. Dans le cas de la requête journey par exemple trois champs doivent être complétés : la ville de départ, celle d'arrivé et la date du voyage. Ces champs sont des listes déroulantes obtenues à partir des deux requêtes « company » et « airports » effectuées au préalable.
Utiliser des listes déroulantes permet d'empêcher l'utilisateur de rentrer des valeurs string et ainsi de limiter les réponses vides, les crashs et les attaques sur la base de données.

Vous trouverez ci-dessous le code correspondant à chacune des requêtes :

- "Get all flights of one company"

```
case 'company_flights':
  var company = req.query.for;
  var date = new Date(req.query.on);
  mongoClient.connect(url_db, function(err, db) {
    db.collection(flights_collection).aggregate([
      {
        $match: {
          AIRLINE_NAME:company,
          FL_DATE:date,
        }
      },
      {
        $project: {
          "Airline" : "$CARRIER",
          "Flight Number" : "$FL_NUM",
          "Departure" : "$ORIGIN_CITY_NAME",
          "Arrival" : "$DEST_CITY_NAME",
          "Date" : "$FL_DATE",
          "Departure Time" : "$DEP_TIME",
          "Arrival Time" : "$ARR_TIME",
          "Flight Time" : "$AIR_TIME",
        }
      }
    ]).toArray(function(err, result) {
      if (err) throw err;
      res.setHeader('Content-Type', 'application/json');
      res.send(JSON.stringify(result));
      db.close();
    });
  });
  break;
```

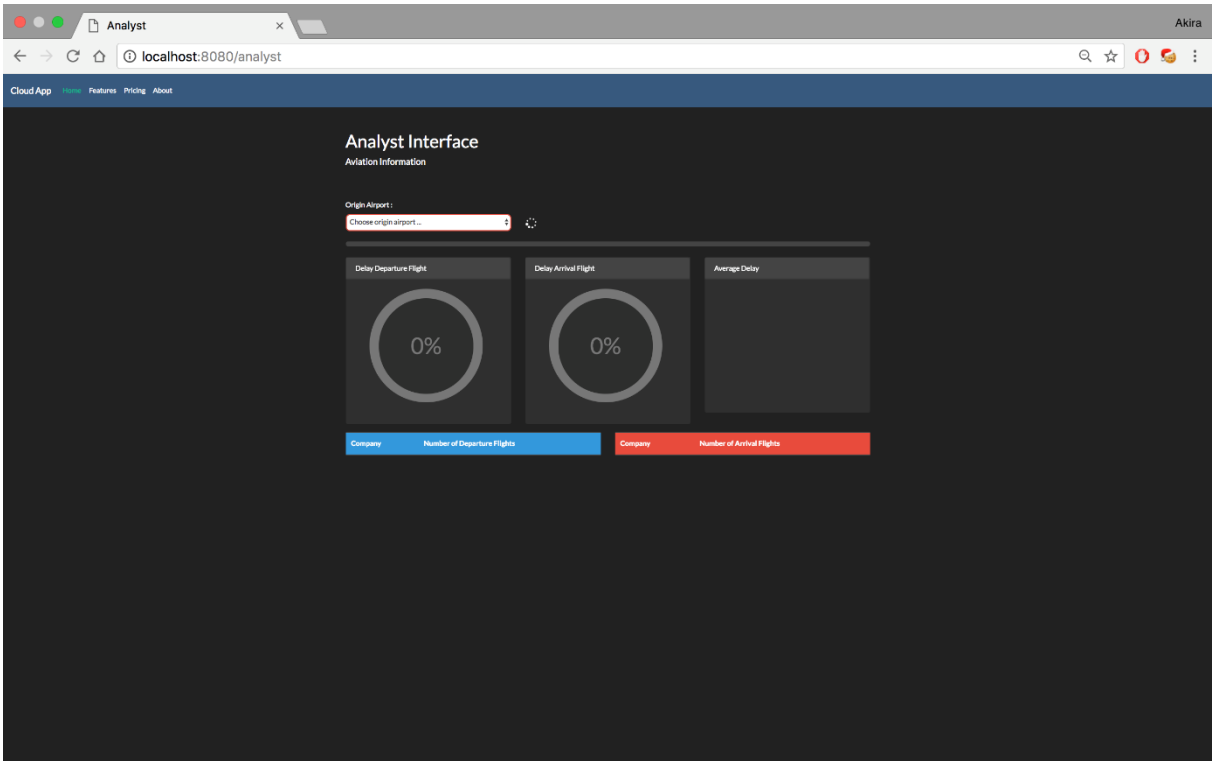
- “Get all flights arrival and departure of one airport”

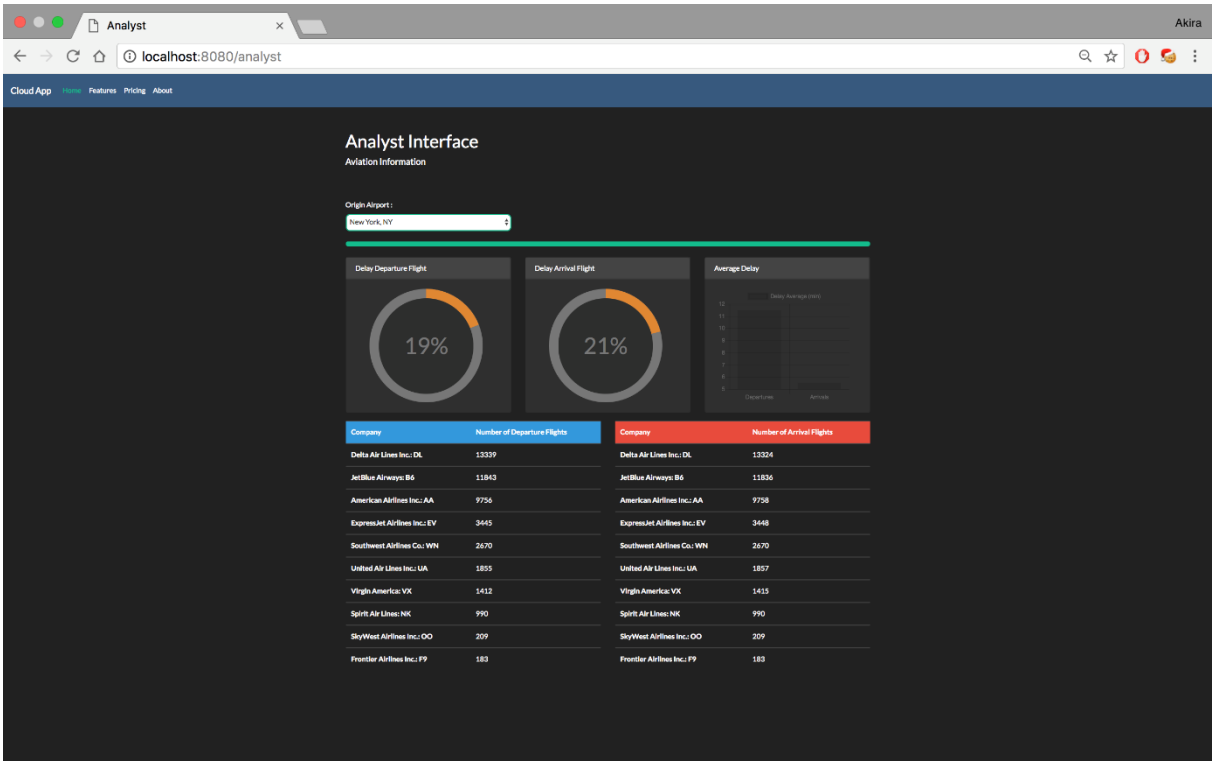
```
case 'airport_flights':
  var airport = req.query.for;
  var date = new Date(req.query.on);
  mongoClient.connect(url_db, function(err, db) {
    db.collection(flights_collection).aggregate([
      {
        $match: {
          $or: [ { ORIGIN_CITY_NAME: airport} , { DEST_CITY_NAME: airport } ],
          FL_DATE:date,
        }
      },
      {
        $project: {
          "Airline" : "$AIRLINE_NAME",      //A CHANGER
          "Flight Number" : "$FL_NUM",
          "Date" : "$FL_DATE",
          "Departure Time" : "$DEP_TIME",
          "Arrival Time" : "$ARR_TIME",
          "Flight Time" : "$AIR_TIME",
        }
      }
    ]).toArray(function(err, result) {
      if (err) throw err;
      res.setHeader('Content-Type', 'application/json');
      res.send(JSON.stringify(result));
      db.close();
    });
  });
  break;
```

- “Get all flights for one journey”

```
case 'journey' :
  var origin_city = req.query.from;
  var dest_cty = req.query.to;
  var date = new Date(req.query.on);
    mongoClient.connect(url_db, function(err, db) {
      db.collection(flights_collection).aggregate([
        {
          $match: {
            ORIGIN_CITY_NAME: origin_city,
            DEST_CITY_NAME: dest_cty,
            FL_DATE: date,
          }
        },
        {
          $project: {
            "Airline" : "$AIRLINE_NAME",
            "Flight Number" : "$FL_NUM",
            "Date" : "$FL_DATE",
            "Departure Time" : "$DEP_TIME",
            "Arrival Time" : "$ARR_TIME",
            "Flight Time" : "$AIR_TIME",
          }
        }
      ]).toArray(function(err, result) {
        if (err) throw err;
        res.setHeader('Content-Type', 'application/json');
        res.send(JSON.stringify(result));
        db.close();
      });
    });
    break;
```

Vue Analyste (analyst.html)





Cette vue est la vue affichant à l'utilisateur les résultats de requêtes de type « Analyste ». Sur cette vue, 5 requêtes sont effectuées :

- **Liste déroulante des aéroports :**

```
exports.getAirports = function(req,res) {
  // Return list of airports city name
  mongoClient.connect(url_db, function(err, db) {
    db.collection(flights_collection).aggregate([
      {
        $group:{
          _id:'$ORIGIN_CITY_NAME'
        }
      },
      { $sort : { _id : 1 } }
    ]).toArray(function(err, result) {
      if (err) throw err;
      res.setHeader('Content-Type', 'application/json');
      res.send(JSON.stringify(result));
      db.close();
    });
  });
}
```

- **Proportion d'avions ayant du retard au départ de l'aéroport choisi + Moyenne de ces retards :**

```
exports.getAvgDelayDep = function(req,res) {
  var airport = req.query.airport;
  mongoClient.connect(url_db, function(err, db) {
    db.collection(flights_collection).aggregate([
      {
        $match: {
          "DEP_DEL15" : { "$exists": true, "$ne": "NULL" },
          "DEP_DELAY" : { "$exists": true, "$ne": "NULL" },
          ORIGIN_CITY_NAME : airport
        }
      },
      {
        $project: {
          _id : "$ORIGIN_CITY_NAME",
          "DepDel15" : '$DEP_DEL15',
          "DepDelay" : '$DEP_DELAY'
        }
      },
      {
        $group: {
          _id : "$_id",
          "avg_Delay": { $avg: "$DepDel15"},
          "avg_DelayTime": { $avg: "$DepDelay"},
        }
      }
    ]).toArray(function(err, result) {
      if (err) throw err;
      res.setHeader('Content-Type', 'application/json');
      res.send(JSON.stringify(result));

      db.close();
    });
  });
}
```

- **Proportion d'avions ayant du retard à l'arrivée de l'aéroport choisi + Moyenne de ces retards :**

```
exports.getAvgDelayArr = function(req,res) {
  var airport = req.query.airport;
  mongoClient.connect(url_db, function(err, db) {
    db.collection(flights_collection).aggregate([
      {
        $match: {
          "ARR_DEL15" : { "$exists": true, "$ne": "NULL" },
          "ARR_DELAY" : { "$exists": true, "$ne": "NULL" },
          DEST_CITY_NAME : airport
        }
      },
      {
        $project: {
          _id : "$DEST_CITY_NAME",
          "ArrDel15" : '$ARR_DEL15',
          "ArrDelay" : '$ARR_DELAY',
        }
      },
      {
        $group: {
          _id : "$_id",
          "avg_Delay": { $avg: "$ArrDel15"},
          "avg_DelayTime": { $avg: "$ArrDelay"},
        }
      }
    ]).toArray(function(err, result) {
      if (err) throw err;
      res.setHeader('Content-Type', 'application/json');
      res.send(JSON.stringify(result));

      db.close();
    });
  });
}
```

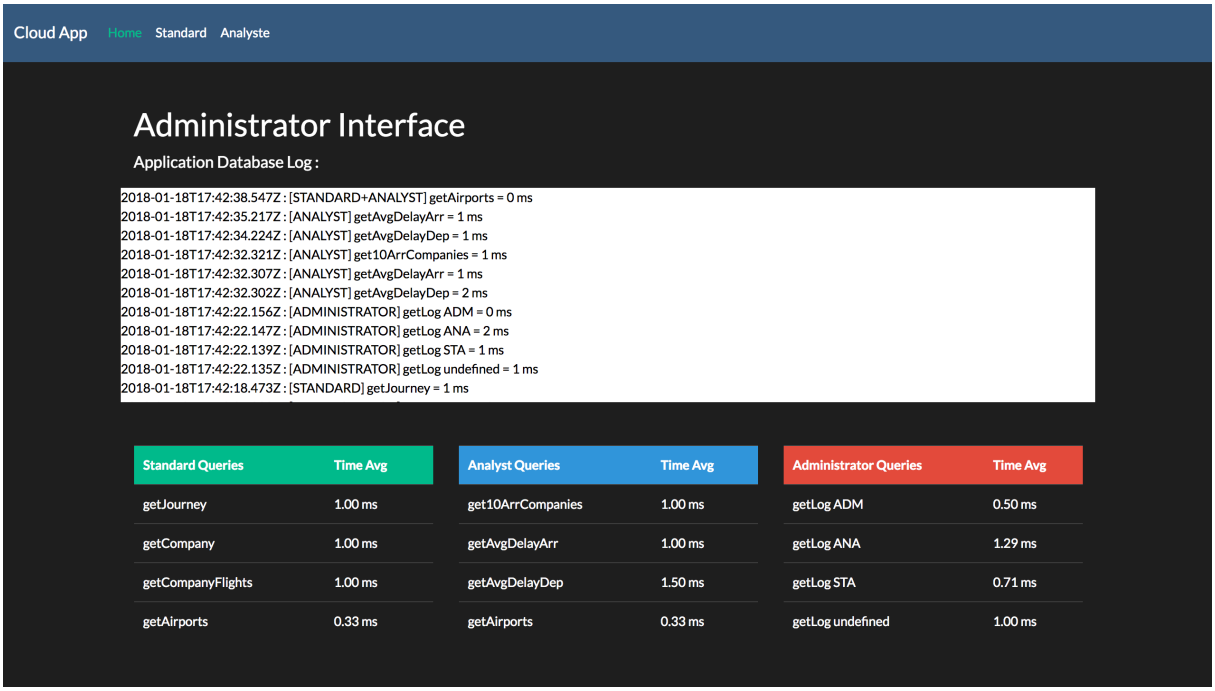
- **Top 10 des compagnies ayant le plus de vol en départ de cette aéroport :**

```
exports.get10DepCompanies = function (req,res) {
  var airport = req.query.airport;
  mongoClient.connect(url_db, function(err, db) {
    db.collection(flights_collection).aggregate([
      {
        $match: {
          ORIGIN_CITY_NAME : airport
        }
      },
      {
        $project: {
            AIRLINE_NAME : 1
        }
      },
      {
        $group: {
          _id: "$AIRLINE_NAME",
          count : {$sum : 1}
        }
      },
      {
        $sort: {
          count : -1
        }
      },
      {
        $limit: 10
      }
    ]).toArray(function(err, result) {
      if (err) throw err;
      res.setHeader('Content-Type', 'application/json');
      res.send(JSON.stringify(result));
      db.close();
    });
  });
}
```

- **Top 10 des compagnies ayant le plus de vol en arrivée de cet aéroport :**

```
exports.get10ArrCompanies = function(req,res) {
  var airport = req.query.airport;
  mongoClient.connect(url_db, function(err, db) {
    db.collection(flights_collection).aggregate([
      {
        $match: {
          DEST_CITY_NAME : airport
        }
      },
      {
        $project: {
            AIRLINE_NAME : 1
        }
      },
      {
        $group: {
          _id: "$AIRLINE_NAME",
          count : {$sum : 1}
        }
      },
      {
        $sort: {
          count : -1
        }
      },
      {
        $limit: 10
      }
    ]).toArray(function(err, result) {
      if (err) throw err;
      res.setHeader('Content-Type', 'application/json');
      res.send(JSON.stringify(result));
      db.close();
    });
  });
}
```

Vue Admin



Cette vue est composée en 4 principaux blocs :

- Un bloc de log des requêtes effectuées sur l'application (affichant : date, type, nom et temps de la requête)
- Un tableau affichant la moyenne du temps de réponse par requête pour la partie « Standard »
- Un tableau affichant la moyenne du temps de réponse par requête pour la partie « Analyste »
- Un tableau affichant la moyenne du temps de réponse par requête pour la partie « Administrateur»

Pour générer ces données, nous avons fait le choix d'exécuter une requête de type « insert » dans une nouvelle collection (« LOG_ADM ») de notre base de données à la suite de chaque requête effectuée sur l'application, la fonction a été définie comme ci-dessous :

```
function insertToLog(req, res, date, query, type, time) {
  var new_log = {
    "date" : date,
    "query" : query,
    "type" : type,
    "time" : time
  };

  mongoClient.connect(url_db, function(err, db) {
    if (err) throw err;
    db.collection(log_collection).insertOne(new_log, function(err,res) {
      if(err) throw err;
      console.log("1 document inserted");
      db.close();
    });
  });
}
```

Cette nouvelle collection est alors requêtée afin d'afficher les différentes informations sur les requêtes comme suit :

```
exports.getLog = function(req,res) {
  var date;
  var t = req.query.type;
  if(t==undefined) {
      date = new Date();
      mongoClient.connect(url_db, function(err, db) {
        db.collection(log_collection).find(
        ).sort({date: -1}).toArray(function(err, result) {
          if (err) throw err;
          res.setHeader('Content-Type', 'application/json');
          res.send(JSON.stringify(result));
          db.close();
        });
      });
      insertToLog(req, res, date, "getLog "+t, "ADMINISTRATOR", new Date() - date);
    } else {
      date = new Date()
      mongoClient.connect(url_db, function(err, db) {
        db.collection(log_collection).aggregate([
          {
            $match: {
              type : {$regex : ".*"+t+".*"}
            }
          },
          {
            $project: {
              query : 1,
              time : 1
            }
          },
          {
            $group: {
              _id : "$query",
              "time_avg" : { $avg : "$time"}
            }
          }
        ]).toArray(function(err, result) {
          if (err) throw err;
          res.setHeader('Content-Type', 'application/json');
          res.send(JSON.stringify(result));
          db.close();
        });
      });
      insertToLog(req, res, date, "getLog "+t.substring(0,3), "ADMINISTRATOR", new Date() - date);
    }
}
```