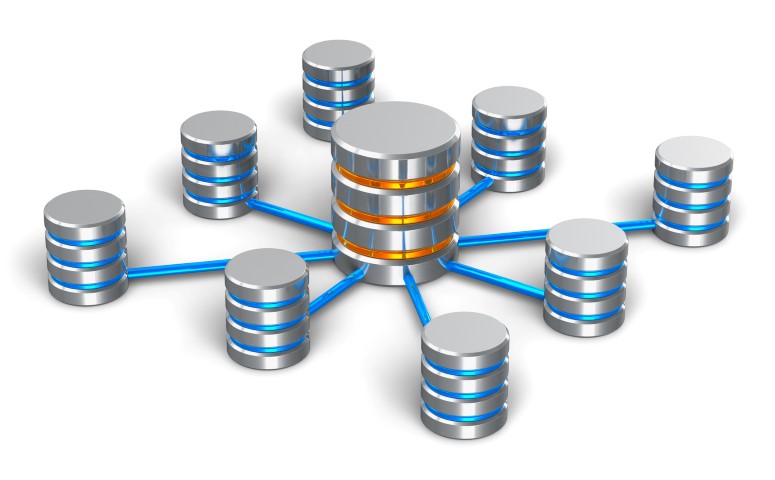
**جامعة دمشق**

**كلية الهندسة المعلوماتية**

Initial report for:  
**Distributed DBs Synchronization System**

****

**السنة الرابعة**

**هندسة البرمجيات و نظم المعلومات**

**اعداد الطلاب:**

**محمد مصطفى ضياء علاء الدين بشر التوبة محمد منار الكل**

**الجدول المحتويات**

* **الفصل الأول: المقدمة .......................................................................... 3**
  + 1.[1 تمهيد (توصيف المشكلة)](#تمهيد)
  + [2.1الهدف من المشروع](#الهدف_من_المشروع)
  + [3.1 المخطط الزمني لتنفيذ المشروع](#المخطط_الزمني_للمشروع)
* **الفصل الثاني : توصيف النظام ................................................................ 5**
  + - [1.2تطبيق المخدم](#تطبيق_المخدم)
    - [2.2تطبيق الزبون](#تطبيق_الزبون)
    - [3.2 مثال على سيناريو الاستخدام](#مثال_على_سيناريو_العمل)
    - [4.2 الشريحة المستفيدة من النظام](#الشريحة_المستفيدة)
    - [5.2 المتطلبات الغير وظيفية](#المتطلبات_الغير_وظيفية)
* **الفصل الثالث: الدراسة المرجعية و النظرية .................................................. 8**
  + [1.3 OpenMRS Wiki](#open_wiki)
  + المشروع الثاني...
* **الفصل الرابع: تحليل النظام .................................................................. 11**
  + [1.4النموذج الاجرائي المستخدم](#النموذج_الاجرائي)
  + [2.4 مخطط حالة الاستخدام](#المخططات_التحليلية)
  + [3.4 مخطط الصفوف](#المخططات_التحليلية)
  + [4.4 مخطط الكيانات العلائقي ERD](#المخططات_التحليلية)
* **الفصل الخامس: تصميم النظام ............................................................... 14**
  + [1.5المخطط التصميمي للصفوف](#مخطط_الصفوف_تصمصمي)
  + [2.5 المخطط التصميمي لقاعدة المعطيات](#مخطط_كيانات_تصميمي)
  + [3.5 المخطط التسلسلي](#مخطط_تسلسلي)
* **الفصل السادس: التحقيق البرمجي .......................................................... 16**
  + التقنيات المستخدمة
  + [Restful API](#restfulAPI)
  + JAVA2EE

**المقدمة**

* 1. **تمهيد:**

تعتمد الكثير من برمجيات اليوم في خدماتها على توفير مركز بيانات موحد يتم تخزين فيه البيانات بشكل مركزي بحيث يمكن الوصول اليها من أي منصة و من أي مكان , فأصبحت التطبيقات اليوم لا تعتمد على قاعدة المعطيات المحلية فقط و انما تقوم بإرسال البيانات عبر الشبكة لتخزينها و توفير إمكانية الوصول اليها من مستخدمين اخرين عبر تطبيقات أخرى , ولم تعد البيانات محصورة في حاسوب وحيد او عند مخدم وحيد و انما يتم تناقلها و نسخها و معالجتها على عدة منصات مختلفة منها المخدمات و منها الحواسيب الشخصية و منها الأجهزة المحمولة و غيرها... , فنجد ان تكرار البيانات Redundancy له أهمية عالية من اجل الاستخدامات التالية:

* النسخ الاحتياطي: فمن المهم تكرار المعطيات للتأكد من عدم فقدانها عند حاجتها و هي من المتطلبات الهامة الغير وظيفية لأي منتج برمجي يعتمد معالجة البيانات الحساسة
* السرعة: تتكرر معطيات مخدمات الانترنت المركزية في العالم و يتم وضع عدة نسخ من نفس البيانات في عدة أماكن في العالم و سبب ذالك هو توفير الوصول السريع للبيانات دون الحاجة لمرور الطلب على مسارات طويلة و مسافات طويلة للوصول لمركز المعطيات data center الذي يحوي المعلومات المطلوبة
* التوفرية: قد نضطر بعض الأحيان الى توفير البيانات محليا (دون الاتصال بالإنترنت) و ذلك بسبب عدم توفر الاتصال الدائم بالإنترنت و الحاجة الى وصول الدائم للبيانات و من هنا تأتي أهمية إيجاد نسخة محلية من المعطيات يمكن الوصول اليها دون الحاجة للاتصال بالإنترنت

وللأسباب المذكورة سابقا قمنا بتطوير نظام يقوم بمزامنة البيانات لعدة قواعد معطيات موجودة على عدة منصات مختلفة (عدة حواسيب)بحيث تكون جميعها تحوي نفس البيانات.

**2.1- الهدف من المشروع:**

توفير لمستخدمي قواعد معطيات على منصات متفرقة طريقة لمزامنة محتواها بشكل آلي (دون تدخل المستخدم أو المطور) بحيث يجعل جميع قواعد المعطيات المشاركة بالخدمة تحوي نفس البيانات , ويمكن أن يتم مزامنة عدة قواعد معطيات مع بعضها البعض بشرط أن تكون جميعها تملك نفس البنية, ولا يشترط أن تتم مزامنتها جميعا مع المخدم في نفس الوقت وإنما كل DB على حدا , ويوجد مخدم يخزن البيانات التي يتم مزامنتها و يتأكد أن جميع الحواسيب المشاركة تحوي هذه البيانات , وهذا المخدم هو المسؤول عن جميع عمليات إدارة المزامنة.

**1,3- المخطط الزمني للمشروع:**

في تاريخ 7\10\2017 اجتمع أعضاء الفريق الأربعة و تم وضع المخطط الزمني الكامل للمشروع منذ لحظة البدء الى إيجاد النظام على ارض الواقع جاهز للتسليم , وفق الخطة التالية:

* من 7\10\2016 وحتى 7\11\2016 تحديد فكرة المشروع و دراسة صلاحيتها كمشروع للسنة الرابعة
  + ينتج عن هذه المرحلة قائمة بأفكار المشاريع الممكن تنفيذها و يرفق مع كل فكرة مدى فعاليتها و قابلية استثمارها في السوق
* من 7\11\2016 حتى 7\12\2016 تحديد خدمات النظام الأساسية التي يقدمها (المتطلبات) و مجال العمل و الشريحة المستفيدة
  + ينتج عن هذه المرحلة فكرة المشروع المعتمدة و اهم الوظائف المراد تحقيقها
* من 7\12\2016 حتى 15\1\2017 تحديد مشاريع سابقة تملك نفس الفكرة العامة و دراستها و انشاء تلخيص عنها (مشروعين على الأقل)
  + ينتج عن هذه المرحلة: أسماء مشاريع مشابهة للمشروع المراد تنفيذه و تقرير بحثي يوصف هذه المشاريع
* من 15\1\2017 حتى 15\2\2017 حصر المتطلبات الوظيفية و غير الوظيفية المطلوبة من النظام و تصنيفها
  + ينتج عن هذه المرحلة: مخطط حالات الاستخدام للنظام و قائمة بالمتطلبات الوظيفية و غير الوظيفية مع شرحها
* من 15\2\2017 وحتى 15\3\2017 تحليل النظام و إيجاد صيغة لإسقاط المتطلبات بمنتج برمجي قابل للتنفيذ
  + ينتج عن هذه المرحلة: النموذج الاجرائي المراد استخدامه process model , و مخططات تحليلية تصف النظام
* من 15\3\2017 وحتى 15\4\2017 تحديد التقنيات الأفضل لاستخدامها في بناء النظام و تصميم prototypes للعمليات المتوقع القيام بها باستخدام هذا النظام
  + ينتج عن هذه المرحلة: قائمة بأسماء التقنيات المستخدمة لتنفيذ المشروع و بيئات البرمجة التي سيتم استخدامها في تطوير النظام
* من 15\4\2017 وحتى 15\5\2017 تحديد البنية التصميمية للمشروع عن طريق بناء المخططات التصميمية و أهمها مخطط الصفوف و تحديد جميع الاجرائيات المستخدمة وعملها
  + ينتج عن هذه المرحلة: المخططات التصميمية للمشروع
* من 15\5\2017 وحتى 1\7\2017 تحقيق المشروع برمجيا (التكويد)
  + ينتج عن هذه المرحلة: منتج برمجي يحقق الوظائف المرجوة منه
* من 1\7\2017 و حتى تسليم المشروع: الاختبار و التصحيح
  + ينتج عن هذه المرحلة: المنتج البرمجي المتكامل الخالي من الأخطاء (قدر الإمكان) و تقرير نهائي يعد بمثابة توثيق للمشروع.

**الفصل الثاني: توصيف النظام**

* نظامنا البرمجي يتكون من قسمين:
  + تطبيق على المخدم (Central server)
  + تطبيق محلي يكون موجود على المنصات الموزعة (أي الحواسيب الأخرى التي تريد المزامنة)

1.3.1- تطبيق المخدم:

هو تطبيق ويب (web service) مسؤول عن إدارة عملية المزامنة حيث انه يستقبل البيانات الجديدة من احد الحواسيب ويضعها في جميع الحواسيب الأخرى في مكانها المناسب و يخزن نسخة منها عنده ويقوم بمعالجة التضاربات الممكن حدوثها نتيجة تعدد الحواسب التي تتواصل مع بعضها, و من التضاربات الممكن حدوثها:

* + تعديل حاسوبين لنفس البيانات في نفس الجدول
  + إضافة حاسوب لتسجيلة تحوي مفتاح أساسي مطابق لتسجيلة أخرى موجودة
  + التعديل في بنية قاعدة المعطيات المحلية في احد الحواسيب
  + وغيرها...
* يوجد آليات لمعالجة التضاربات:
  + إعطاء أهمية للبيانات حسب تاريخ إضافتها (مثلا المضافة حديثا أهم من القديمة)
  + إعطاء أهمية للبيانات حسب الحاسوب الذي يضيفها ,أي إعطاء أولويات للحواسيب و تكون البيانات الأهم هي بيانات الحاسوب ذو الأولوية الأعلى
  + من ممكن دمج البيانات لنفس السطر في حال أمكن ذالك (وحسب رغبة المستخدم)
  + يوفر النظام للمستخدم اليات لتحديد هذه الخيارات عند إنشاء قاعدة المعطيات المركزية في المخدم
* و عند حدوث أي حدث يعيد المخدم للحاسوب الهدف معلومات متعلقة بالحدث (استثناءات, أخطاء , نجاح العملية , المدة , تضاربات ,...) و يسجلها في سجل الأحداث
* يتيح المخدم لمسؤول قواعد المعطيات أدوات تمكنه من إنشاء قاعدة المعطيات المركزية و تحديد خصائصها و ووضع كلمة مرور للتأكد من هوية الحواسيب المتصلة بهذه القاعدة , كما يعرض للمسؤول بيانات عن حالة القاعدة حاليا و الحواسب المتصلة بها و حالتهم و تسلسل العمليات الحاصلة عليها و غيرها...
* الهدف من قاعدة المعطيات المركزية هي تخزين نسخة من البيانات جميعها والتأكد من أن كل قواعد المعطيات المتصلة بها أصبحت تحوي هذه البيانات و تكون بنية هذه القاعدة هي نفس بنية جميع القواعد المتصلة بها

2.3.1- تطبيق الزبون

هو المسؤول عن إرسال المعطيات إلى المخدم حيث عند إعداده يتم ربطه بقاعدة المعطيات المحلية و ربطه بالمخدم و يقوم (عند طلب المستخدم، أو بشكل آلي) بتحديد المعطيات الجديدة التي تم التعديل عليها و رفعها إلى المخدم و يتم تحديد المعطيات الجديدة بإحدى الآليات التالية:

* + حفظ تاريخ آخر عملية مزامنة تمت و طلب استعلام من محرك قاعدة المعطيات المحلية DBMS لإعطائنا كل البيانات التي تم التعامل معها بعد تاريخ آخر مزامنة
  + مراقبة عمل القاعدة المحلية و تخزين جميع العمليات و أرقام الأسطر التي يتم التعامل معها و استخدام هذه المعلومات عند بدء المزامنة
* يتيح التطبيق المحلي للمستخدم تحديد قاعدة المعطيات المحلية المراد الربط بها و يحدد قاعدة المعطيات المركزية على المخدم المراد الاتصال بها.
  + كما يقدم الإحصاءات و العمليات التي تتم و الحالة و رد المخدم و غيرها...
    1. مثال على سيناريو العمل:

بفرض أن المسؤول يريد مزامنة حاسوبين يحويان قاعدتي معطيات متشابهتين ,فيقوم بالتالي:

1. يقوم بإنشاء قاعدة معطيات جديدة للمزامنة
   1. يعطيها اسم و ينشئ الجداول المشتركة بما تحويه من أعمدة و خصائص
   2. يحدد كلمة المرور الخاصة بهذه القاعدة ليتم الاتصال بها من قبل الحواسيب الأخرى
   3. حدد خصائص المزامنة و طريقة معالجة التصادمات و غيرها...
2. يحمل التطبيق على الحاسوب الذي يحوي قاعدة المعطيات المحلية , ويستخدمه للتالي:
   1. يحدد قاعدة المعطيات المحلية المراد مزامنتها
   2. يضع اسم قاعدة المعطيات المركزية المراد المزامنة معها و يدخل كلمة المرور الخاصة بها
   3. يحدد طريقة المزامنة و توقيتها (عند الاتصال بالشبكة , كل مدة محددة , عند الساعة المحددة ,يدويا,...) و يحدد آلية معالجة التصادم على قاعدة المعطيات المحلية

يتم فعل البند الثالث من اجل كل حاسوب زبون يريد المزامنة ضمن هذه القاعدة

**4,3,1- الشريحة المستفيدة من النظام:**

يستفيد من نظام مزامنة قواعد المعطيات المؤسسات التي تملك نقاط عمل موزعة جغرافيا و مرتبطة بنفس مخزن البيانات و تحتاج لدمج البيانات التي تتعامل معها بين الحين و الاخر(مثل تخزين منتسبين جدد , تسجيل احداث لا تعتمد على احداث لم يتم مزامنتها, تعديل البيانات المبني على الأولويات -أي يتم اعتماد التحديث ذو التاريخ الاحدث او التحديث الحاصل من قبل الزبون ذو الأولوية الأعلى- , مراكز الدراسات و الإحصاء, ...) و يشترط توفر الانترنت بكل نقطة ولكن ليس بشكل دائم و انما من الممكن العمل على النظام في حال عدم توفر اتصال و تخزين الاحداث الحاصلة على البيانات محليا و عند توفر الاتصال يتم المزامنة, ولكن **يجب الانتباه** الى ان الانقطاع عن الانترنت يعني عدم الدراية بما يحدث من تبدلات و تغيرات في قاعدة المعطيات البعيدة و بالتالي اذا كان الوصول للمعطيات مطلوب بشكل اني (بالزمن الحقيقي) من اجل كل نقاط العمل معا (مثل البنوك و مراكز الخدمات التفاعلية) فلم يعد مقبول وظيفيا العمل بدون انترنت.

**5,3,1- المتطلبات الغير وظيفية الموجودة بالنظام:**

* Availability : يوفر للزبون الولوج الى قاعدة المعطيات حتى في حال عدم وجود اتصال بالانترنت
* Performance: يتم تبادل المعطيات مع المخدم باستخدام تقنية restful API التي تعتمد على تغليف المعطيات كJSON مما يزيد من سرعة تبادل البيانات مع المخدم
* Portability : تم تصميم النظام على شكل مكتبة java .jar والتي يمكن استخدامها على المنصات المختلفة
* Reusability : وهي الصفة التي تتصف بها الصفوف في النظام , فمثلا تم استخدام نفس صفوف الاتصال بقواعد المعطيات JDBC عند المخدم و الزبون
* Compatibility : يدعم النظام تعدد محركات قواعد المعطيات فمن الممكن ان يكون احد الزبائن يعمل على SQL مثلا و الاخر يعمل على MySQL و المخدم يعمل على محرك اخر , تم تصميم النظام ليقبل دعم محركات جديدة بسهولة (مثلا دعم Oracl and SQLlit)

**الفصل الثالث: الدراسة المرجعية والاعمال المماثلة**

## OpenMRS Wiki - 1.2 :

هو عبارة عن برنامج و منصة تواصل. كبرنامج يقدم خدمة لنظام سجلات طبية الكترونية مصمم من اجل الدول النامية. ولأنه مشروع مفتوح المصدر تم الاعتماد عليه و زيادة بعض المواصفات ليصبح منصة للمعلومات الطبية تستخدم في كل قارة.

تم عمل البرنامج لان هناك العديد من الناس في الدول النامية يعانون من امراض خطيرة مختلفة مثل التهاب الكبد الوبائي والايدز .. الخ وبسبب قلة المعلومات والمصادر والتعليم يؤدي ذلك لارتفاع نسبة الوفيات و لذلك ظهرة الحاجة الى نظام سجلات طبي فعال و لكن العديد من هذه الدول تستخدم الأوراق التي تكون معرضة للتلف او الضياع او التعقيد في البحث عن المعلومات و من هنا ظهرت الحاجة النظام الالكتروني متزامن و ذلك لأنه في هذه الدول غالبا ما تكون خدمة الانترنت غير متوفرة.

في البداية كانت الفكرة المبدأية للمشروع تُعنى بمرضى التهاب الكبد الوبائي في كينيا ورواندا و لكن هؤلاء المرضى قد يكون لديهم امراض أو مشاكل أخرى لذلك اتجه إلى دعم العديد من الأمراض و الاهتمام بالصحة الإنجابية و العديد من المشاكل الأخرى ليصبح نظام سجلات طبي متعدد الأغراض يقدم خدمات للاعتناء بالمرضى و تجميع الملاحظات و المراقبة و تحويل هذه المعلومات الى ملخصات مفيدة و تقارير التي ستحسن الفعالية للناس الذين سيستخدمون التطبيق.

التطبيق هو web Application مبني بلغة ال java و يستخدم قاعدة معطيات mysql و لكن يمكن استخدامه مع قواعد معطيات أخرى باستخدام تقنية ال Hibernate ،كما أنه يدعم أكثر من نظام تشغيل مثل Linux و Windows و Mac OS X، بدأ ب data model بسيط ثم غلفوه من خلال API ثم قاموا ببناء web-based application يستخدم هذا ال API .يقوم ال API بإخفاء العمليات التي تحصل في ال data model و ضمان أن كل التطبيقات التي تستخدم ال API تعمل وفق نفس القواعد لتنظيم السجلات الطبية الالكترونية.

التطبيق مبني بطريقة ليدعم ال modules و ذلك لأنه يمكن تغير سلوك النظام بناءً على ما المتطلبات المحلية للزبون في منطقته.

التطبيق مبني بتعاون ضخم من شركات و منظمات مختلفة و ذلك بالاعتماد على تشارك الأفكار و إعطاء الآراء في المشاكل البرمجية و الطرق المختلفة للبناء و الاستخدام و هو مستخدم في العديد من الدول مثل الولايات الأمريكية المتحدة و الأرجنتين و الكونغو و العديد من دول شرق و غرب اسيا و وسط افريقيا و يمكن استخدامه فرديا او من خلال منظمات او حكومات و شركات ربحية او غير ربحية.

كما قلنا فإن التطبيق يقوم على مزامنة البيانات و ذلك بين وسائط التخزين المحلية و المخدمات المركزية و تتم عملية المزامنة بعدة طرف في التطبيق منها master-master replication و conflict resolution و يمكن تطبيق عمليات import , export , transfer على قاعدة المعطيات.

في المرحلة الأولى من التطبيق تم بناء مزامنة قاعدة المعطيات بافتراض أن الزبون لن يقوم سوى بإدخال المرضى و مراقبتهن من دون تغير في قيم الحقول المتواجدة أي أنه لن يكون هنالك تصادمات و بالتالي سيكون هنالك قاعدة معطيات واحدة متكاملة اسمها master مع مراعاة أن معالجة التصادمات ستتم في المراحل القادمة من التطبيق.

في المرحلة الأولى تتم المزامنة ب:

1. العيادة او المستوصف او المستشفى (يمثل الابن) تبدأ مع نسخة مطابقة لقاعدة معطيات الأب (master)
2. يتم في هذه المرحلة إعطاء ID خصوصي للابن واحضار البنية الأساسية لقاعدة المعطيات
3. اختيار طريقة ارسال المعلومات من الأب الى الابن يقوم بعدها بإرسال معلومات عن مخدم الأب الى الابن وارسال المعلومات المطلوبة
4. بعد ذلك يقوم الابن بتمكين والسماح بالمزامنة ومن هذه المرحلة سيقوم الابن بتتبع التغيرات التي تحدث في قاعدة معطيات الأب

مرحلة ارسال المعطيات من الابن للأب :

1. تجميع التغيرات الجاهزة للإرسال إلى مخدم الأب وعرض هذه السجلات المرسلة عند الابن.
2. عند الابن :
3. اذا تم النقل باستخدام ملف فيقوم بتوليد ملف نقل مزامن و حفظه محليا.
4. اذا تم النقل باستخدام الويب و في حال عدم مطابقة معلومات الأب يرسل رسالة خطأ.
5. اذا تم النقل باستخدام الويب و في حال عدم مطابقة معلومات الأب يقوم بفتح قناة اتصال بينهما و يرسل المحتوى المنقول المزامن إلى مخدم الأب لمعالجته
6. الموافقة على النقل المتزامن من قبل الأب:
7. اذا تم النقل باستخدام ملف فيقوم برفع الملف المنقول المزامن يتم تجهيزه للمعالجة.
8. اذا تم النقل باستخدام الويب قبول الاتصال من الابن و استقبال المعطيات منه لمعالجتها.
9. لمعالجة المعطيات المستقبلة من الابن يتم قراءة كل السجلات المرسلة و وضعهم في قاعدة المعطيات للأب بنفس الترتيب.
10. ارسال و معالجة رسالة موافقة المزامنة:
11. عند الأب: من أجل كل سجل تم ارساله للمزامنة يتم توليد سجل يعبر عن حالة السجل المرسل (هل تم وضعه بنجاح أم تم تخطيه).
12. توليد رسالة استجابة للمزامنة و تجهيزها للإرسال.
13. عند الابن: استقبال رسالة الاستجابة ثم من أجل كل سجل فيها تحديث حالة السجل.

بعض المشاكل التي واجهها المشروع :

1. Global data record (identity tracking) : هل من الضروري مزامنة المفاتيح الرئيسية لقاعدة المعطيات أم لا ؟ و ما مقدار الضرر الذي سيحدث اذا تم اهمال المزامنة بين المفاتيح الرئيسية ؟
2. Global Ordered record history: بما أن التعامل مع الأبناء و السجلات بالاعتماد على الزمن و أن الأبناء يكونون في مناطق مختلفة فيزيائيا هذا يؤدي إلى اختلاف المنطقة الزمنية و بالتالي هذ الاختلاف سيؤثر على عملية المزامنة التي ستحصل مع الأب.

# المراجع :

1. <https://wiki.openmrs.org/display/docs/Home>
2. <https://en.wikipedia.org/wiki/OpenMRS>

2.2 -

**الفصل الرابع: تحليل النظام**

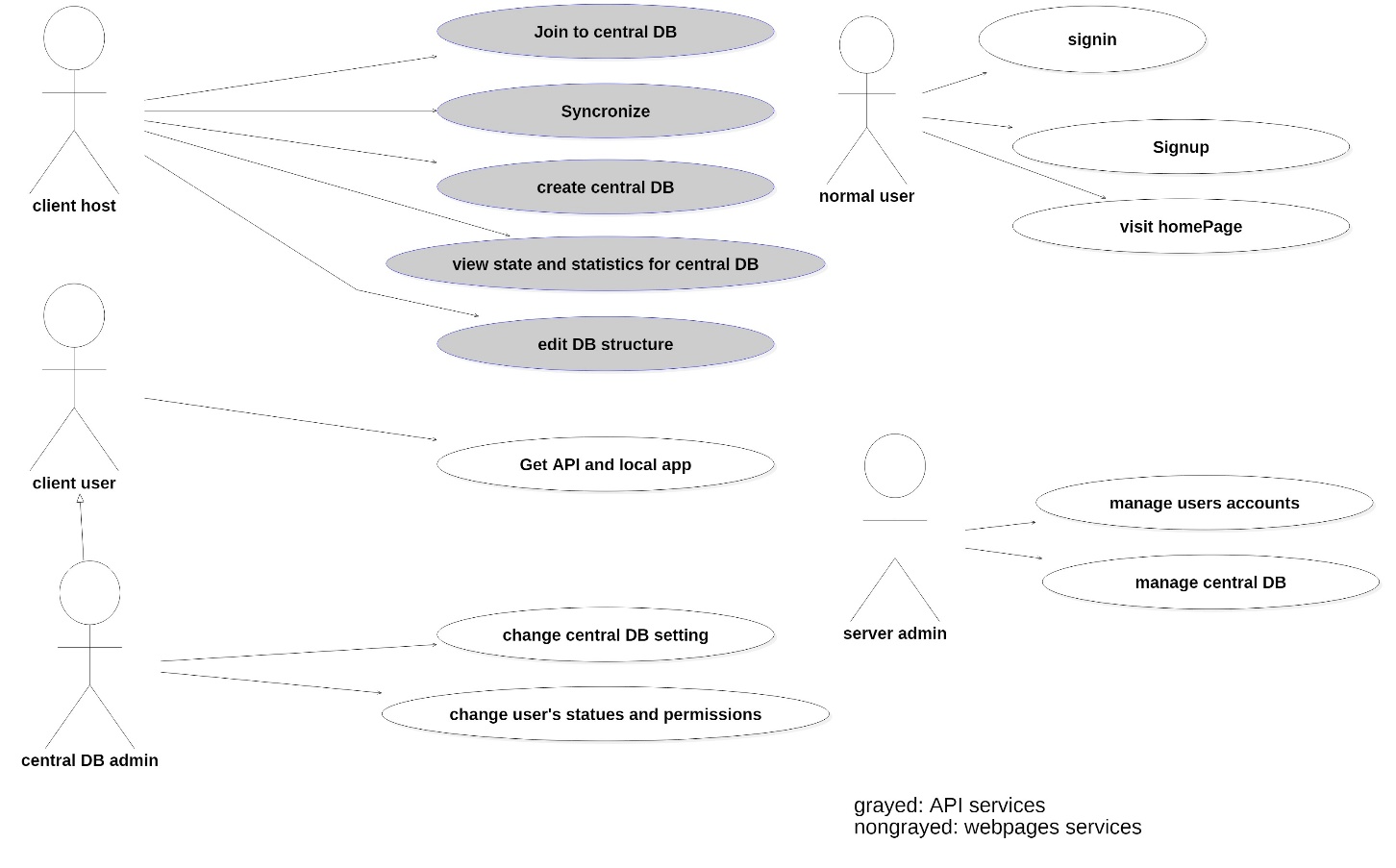
**1.4النموذج الاجرائي المستخدم:**

تم استخدام النموذج الاجرائي التزايدي incremental process model وذلك لما يوفره من مرونة في إضافة المتطلبات , فنحن بحاجة لنموذج يمكننا من تصميم النظام بطريقة واضحة لجميع أعضاء الفريق مع مكانية توزيع المهام بينهم دون حدوث تضارب و الحصول على توافق بين الوحدات modules التي يتم بناءها على التفرع من قبل أعضاء الفريق , يتصف النظام الذي نقوم ببنائه بانه لا يمكن تجزئته لوظائف مستقلة فان كل اجراء في صف ما يعتمد في بنيته على عدة إجراءات من صفوف اخرى وبالتالي لبناء النظام (الطور الأول على الاقل) لا بد من توضيح بنية النظام كاملة و تحليلها و تصميمها و التأكد من صحتها قبل البدء بالبناء البرمجي (implementation ) للتأكد من ان النظام يحقق الوظائف المطلوبة منه.

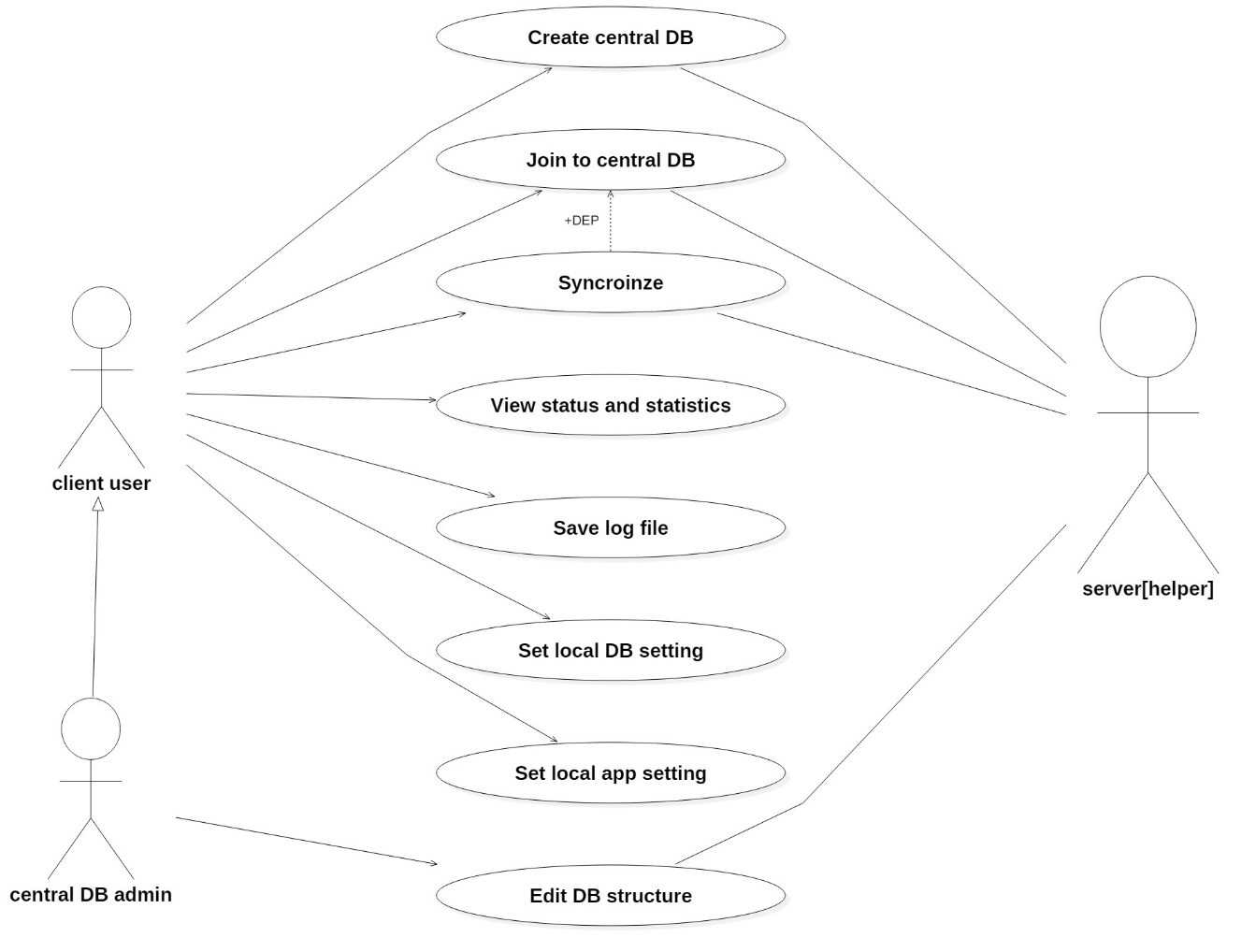
**المخططات التحليلية:**

* 1. **مخطط حالة الاستخدام:**

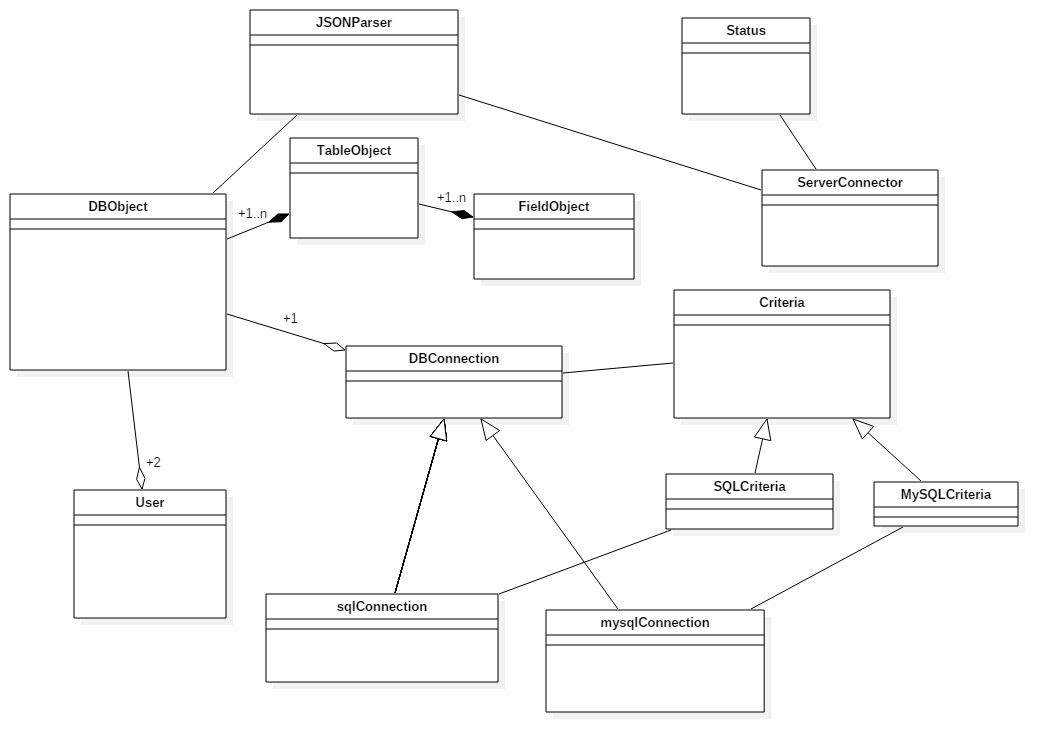
عند المخدم:

****

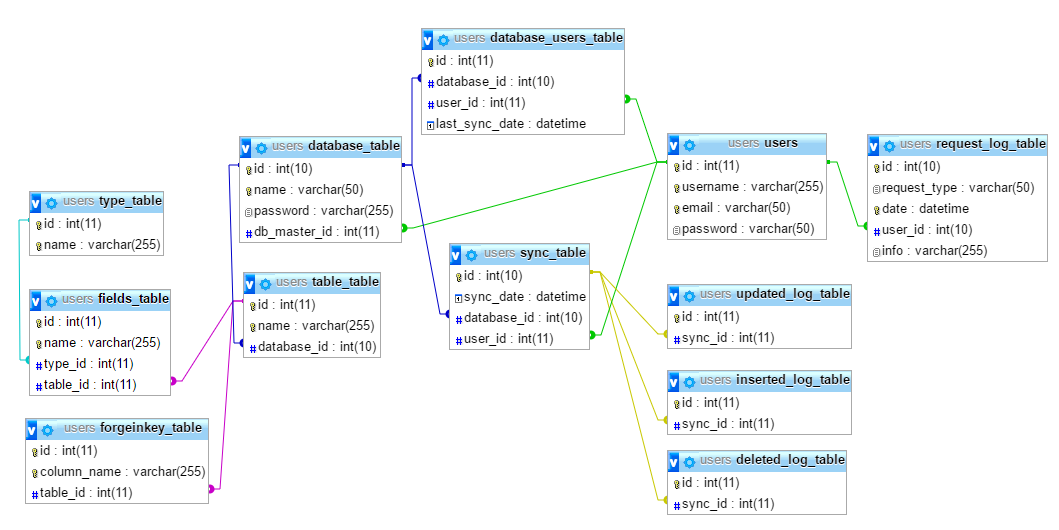
عند الزبون:



**3,4 - مخطط الصفوف:**

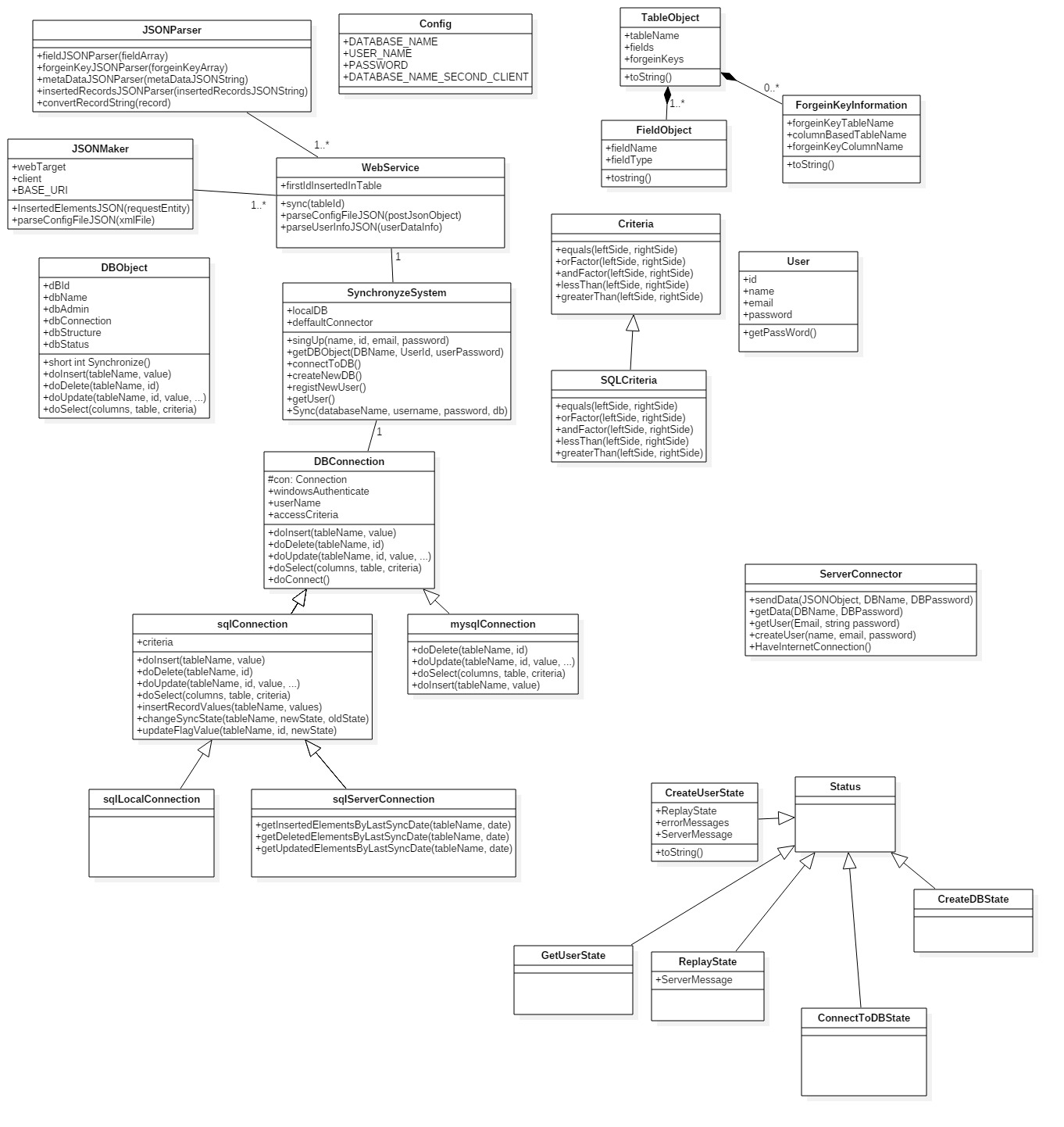


* 1. **مخطط الكيانات العلائقي ERD**

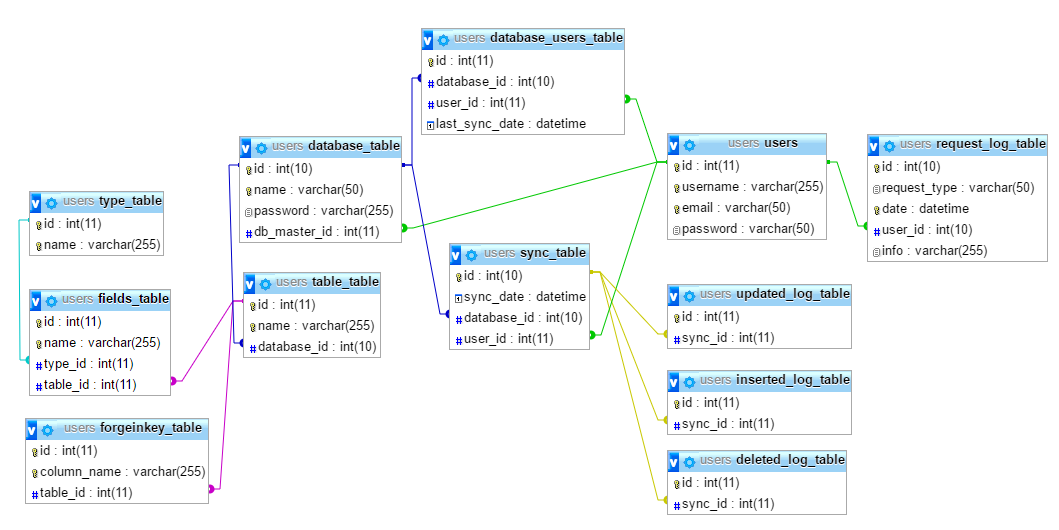


**الفصل الخامس: تصميم النظام**

**1,5 - مخطط الصفوف:**

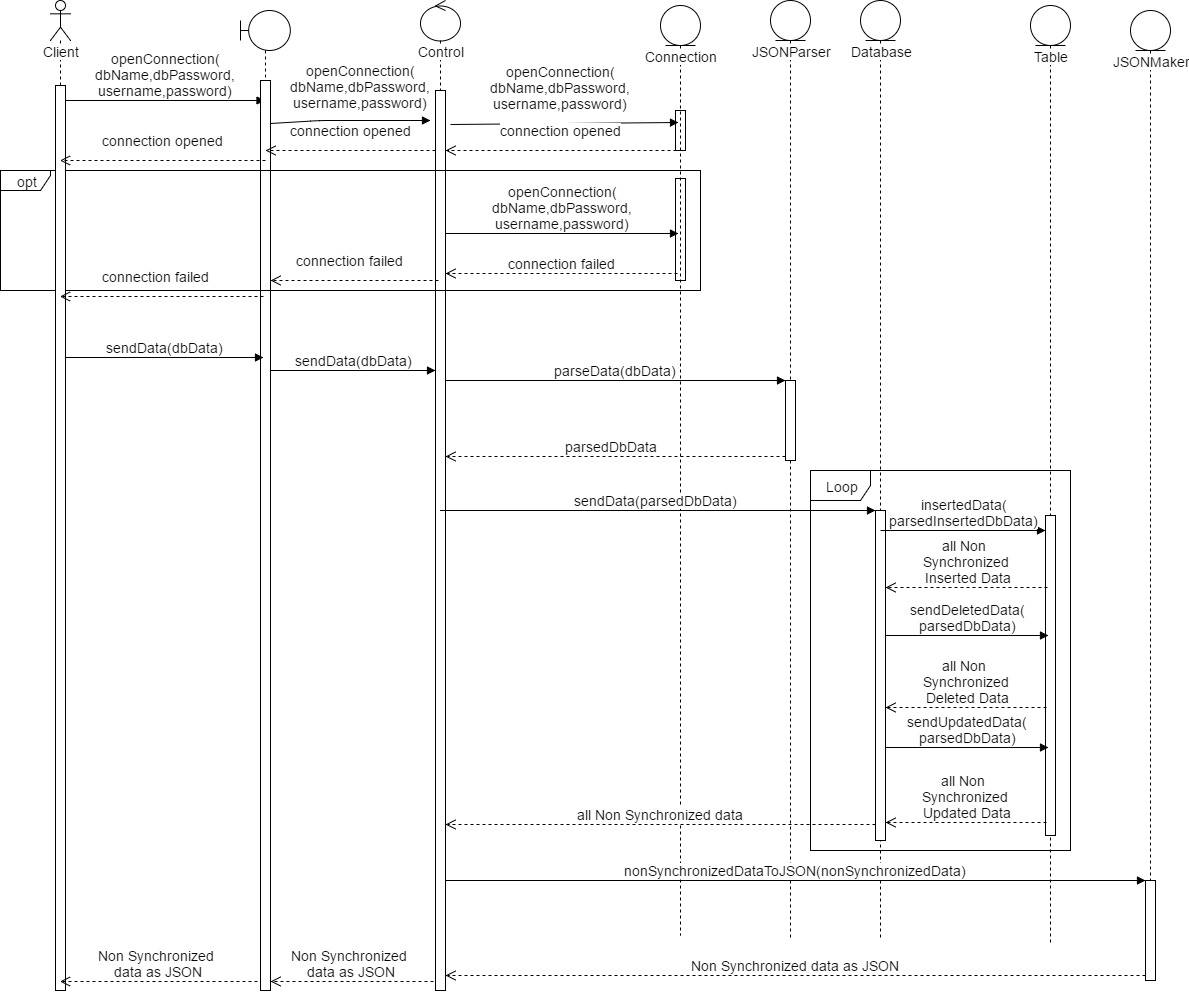


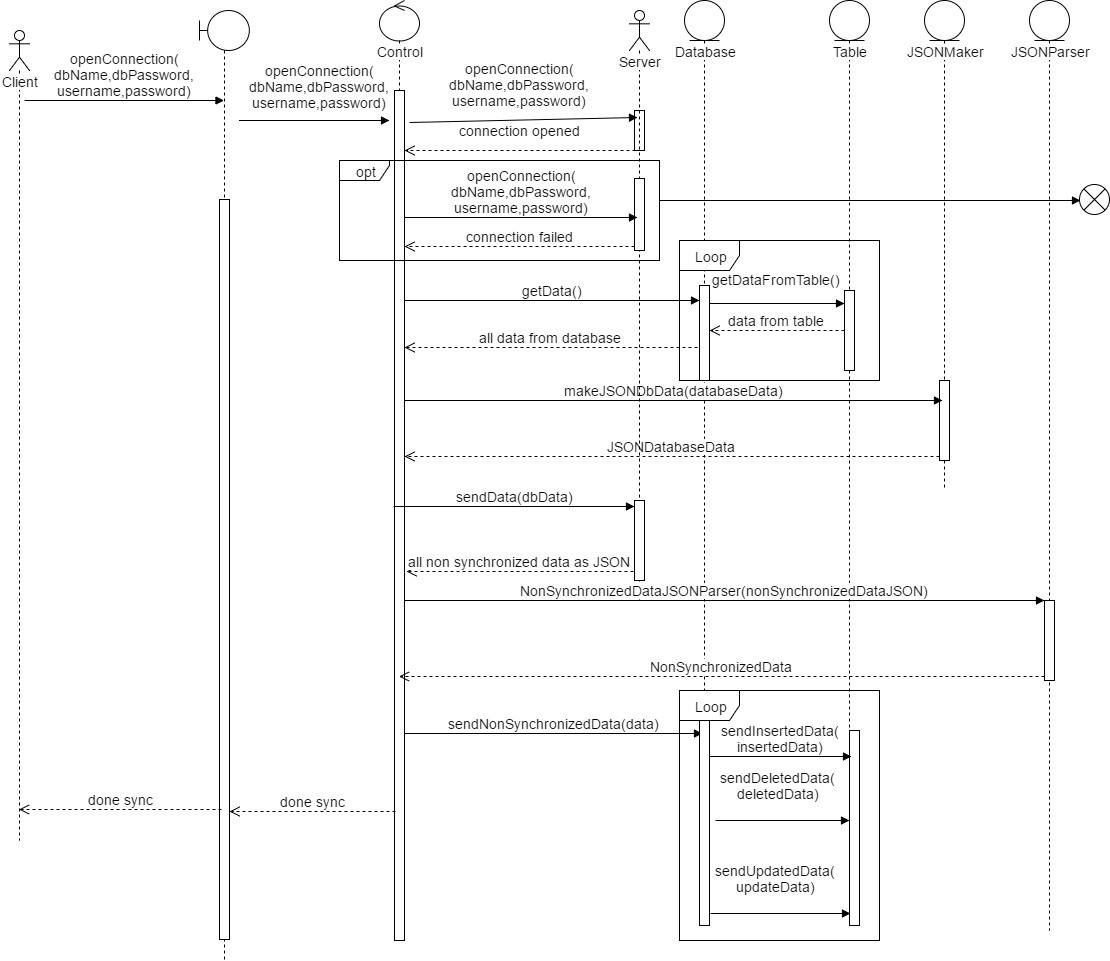
**2,5 مخطط الكيانات العلائقي ERD**



**3.5 المخطط التسلسلي**

عند المخدم:

****

عند الزبون:****

**الفصل السادس: التحقيق البرمجي**

سنقوم بتطوير تطبيق الزبون و المخدم بلغة الجافا باستخدام الأدوات التالية:

* JAVA2EE platform
* Resful API
* JDBC
* NetBeans IDE
* SQL server
* MySQL

## ما هو restful api ؟

لنوضح ما هو rest api يجب ان نوضح مفهوم ال api (application programming interface ) :

ال api هو عبارة عن كود يسهل عملية الاتصال بين برنامج معين و برنامج اخر عن طريق طلب مركب و رد مركب معياريين يفهمها البرنامجين .

كمثال على ذلك المطبخ والزبون في المطعم و هناك قائمة من الطلبات والردود المتعارف عليها في المطعم فيكون ال api هو النادل الذي يقوم بايصال الطلبات و الردود من و الى المطبخ والزبون .

REST API (**Representational state transfer)** :

هو API ولكن له قيود محددة يجب ان يتبعها مما يجعل منه API REST , من هذه القيود :

* استخدام بروتوكولات معيارية و اغلب REST API تستخدم بروتوكول HTTP و تعتمد اعتماد كاملا على توابع الHTTP مثل POST , GET
* STATELESS يعني كلا من CLIENT و SERVER لا يخزنون اي معلومات مؤقته لاتصال , ولكن اي طلب يحوي معلومات عن ماهية الطلب و صاحب الطلب .

مما يجعل له عدة خصائص منها :

* يمكن استخدامه في معظم البيئات و لغات البرمجة لاعتماده على بروتوكول HTTP المعياري .
* سرعة في الاداء و وثوقية عالية.

المرجع :

<https://en.wikipedia.org/wiki/Representational_state_transfer>