

الجمهورية العربية السورية

المعهد العالي للعلوم التطبيقية والتكنولوجيا

قسم النظم المعلوماتية

مشروع أعدّ

لنيل درجة الإجازة في هندسة النظم المعلوماتية

نظام مشاركة ملفات داخلي من نوع ند لند

**P2P Intranet File Sharing System**

إعداد

**محمد بشار دسوقي**

إشراف

**م. حسين خير**

**د. نسيان سابا**

أيلول 2016



# إهداء

إلى من أخذ بيدي من بداية الطريق ... والدتي

إلى من لم ييخل بشيء من أجل دفعي في طريق النجاح ... والدي

إلى من رافقني في مشواري وكان سنداً لي ... عبدالله

إلى الذين لم ييخلوا وقدموا لنا الكثير ... أساتذتي

إلى من جعلهم الله أخواي ... أحمد، أنس، أيمن، خالد، شهم، عبدالرحيم، فؤاد، لؤي، محمد.



# شكر

أولاً نُقدم أسمى آيات الشكر والامتنان والتقدير والمحبة إلى الذين مهدوا لنا طريق العلم والمعرفة، الهيئة التدريسية في المعهد العالي للعلوم التطبيقية والتكنولوجيا، التي لا تتوانى في تقديم أي جهد لرفع السوية العلمية لطلاب المعهد وأُخصّ بالتقدير والشكر **الدكتور غسان سابا** صاحب الفضل الكبير في إتمام هذا العمل، و**المهندس حسين خيو** لما قدمه من دعم لنا.



## ملخص

تعتمد شبكات الند للند على التعاون بين المستخدمين والتشارك في الموارد، وقد شاع استخدام هذا النوع من الشبكات في كثير من التطبيقات، من أشهرها مشاركة الملفات. تقسم تطبيقات مشاركة الملفات، المعتمدة على بنية الند للند إلى منظومات مركزية (تعتمد على وجود مخدم مركزي لتنظيم العمل) ومنظومات موزعة.

نقدّم في هذا العمل تطبيق ند لند مركزي يُستخدم من أجل مشاركة الملفات والمحادثة بالإضافة إلى مزيد من الخدمات الإضافية التي تساهم في تشجيع المستخدمين على استخدام التطبيق بشكل فعال.

**الكلمات المفتاحية:** الند للند – مشاركة الملفات – منظومة مركزية – مخدم مركزي.

## Abstract

The advantage of peer-to-peer (P2P) paradigm relies on two main concepts: cooperation among users and resource sharing. There are many applications based on peer-to-peer paradigm, but the most popular one is the file sharing. We can classify the file sharing application into centralized systems (having a central server), and decentralized systems.

In this project, we have implemented a centralized peer-to-peer application for file sharing, chatting, and another additional services that encourage application's user to be active user.

**Keywords:** Peer-to-peer, file sharing, centralized system, directory server.

## فهرس المحتويات

ملخص .....	
I..... فهرس المحتويات	
V ..... قائمة الأشكال	
VII..... قائمة الجداول	
1 ..... مقدمة	
2 ..... الهدف من المشروع	
2 ..... مخطط تنظيم التقرير	
3 ..... الفصل الأول	
4 ..... 1.1- المتطلبات الوظيفية	
5 ..... 2.1- المتطلبات غير الوظيفية	
7 ..... الفصل الثاني	
8 ..... 1.2- مفهوم شبكة الند للند	
8 ..... 2.2- تصنيف شبكات الند للند	
10 ..... 3.2- النموذج اللامركزي في مشاركة الملفات من نوع ند لند	
11 ..... 4.2- النموذج المركزي في مشاركة الملفات من نوع ند لند	
12 ..... 5.2- مقارنة بين النموذج المركزي واللامركزي في مشاركة الملفات من نوع ند لند	
13 ..... 6.2- بروتوكول Napster	
16 ..... 7.2- بروتوكول BitTorrent	
16 ..... 1.7.2- آلية عمل البروتوكول	
17 ..... 2.7.2- ملف torrent	
18 ..... 3.7.2- أهمية بروتوكول BitTorrent	



19	4.7.2- طرق الاتصال بين الأنداد .....
19	5.7.2- تحميل ومشاركة ملفات Torrent .....
19	8.2- المشاكل الموجودة في أنظمة مشاركة الملفات الحالية والتي من نوع ند لنند .....
20	9.2- الدروس والعبر المستفادة من الدراسة المرجعية .....
21	الفصل الثالث .....
22	1.3- نموذج الإجرائية المعتمد .....
22	2.3- الخطّة الزمنية .....
25	الفصل الرابع .....
26	1.4- مقدمة .....
27	2.4- مخططات حالات الاستخدام .....
27	أولاً- النظام من جهة الزبون .....
30	ثانياً- النظام من جهة المخدم .....
30	3.4- الوصف النصي لحالات الاستخدام ومخططات التسلسل .....
30	1.3.4- تسجيل الدخول إلى النظام .....
32	2.3.4- بدء محادثة .....
34	3.3.4- تحميل ملف .....
36	4.3.4- مشاركة ملف جديد .....
38	4.4- مخطط المفاهيم (مخطط الصفوف الأولى) .....
38	1.4.4- مخطط المفاهيم الخاص بالنظام P2P HiastShare .....
39	2.4.4- مخطط المفاهيم الخاص بنظام إدارة المخدم المركزي .....
41	الفصل الخامس .....
42	1.5- التصميم الرئيسي .....
43	2.5- المكونات الرئيسية للنظام P2P HiastShare .....

44	3.5- آلية التواصل في النموذج المركزي
44	1.3.5- تواصل ند- مخدم Peer- Server
49	2.3.5- تواصل ند- لند P2P
50	3.3.5- التواصل بين تطبيق الزبون HiastShare وتطبيق المخدم المركزي
51	4.5- مخطط قاعدة معطيات المخدم المركزي
53	5.5- تصميم نظام إدارة المخدم المركزي
54	6.5- تأثير التصميم المتبع على توسيع النظام
55	الفصل السادس
56	1.6- بيئة العمل وأدوات التنفيذ لنظام HiastShare
56	مقدمة:
56	1.1.6- تنفيذ نواة اتصال المخدم CommandServer
57	2.1.6- تنفيذ نواة اتصال الزبون CommandClient
57	3.1.6- بروتوكول النقل للمستخدم
57	4.1.6- استخدام المقابس ارتباطية التوجه Connection-oriented Sockets
58	5.1.6- مشكلة التوقف Blocking في التطبيقات الشبكية
58	6.1.6- إدارة النيايب
59	7.1.6- إرسال واستقبال البيانات
60	8.1.6- تنفيذ التطبيق HiastShare
63	1.8.1.6- إدارة بدء تشغيل التطبيق startup
63	2.8.1.6- إنشاء نسخة واحدة فقط للتطبيق single-instance
64	9.1.6- تنفيذ تطبيق المخدم المركزي
64	2.6- بيئة العمل وأدوات التنفيذ لنظام إدارة المخدم المركزي
64	1.2.6- بيئة العمل

64	..... Asp.Net MVC 4 Framework
65	..... 2.2.6- أدوات التطوير
65	..... Entity Framework
66	..... 3.6- بعض الحلول التقنية المفيد ذكرها
67	..... 4.6- الخاتمة وآفاق مستقبلية
69	..... المصطلحات
71	..... المراجع

## قائمة الأشكال

الشكل 1	بنية الشبكة الصافية	9
الشكل 2	بنية الشبكة المهجنة	9
الشكل 3	بنية شبكة Napster مع آلية طلب ملف فيها	12
الشكل 4	بنية رسالة بروتوكول Napster	13
الشكل 5	الرسائل المتبادلة في بروتوكول Napster	14
الشكل 6	رسالة طلب ملف في بروتوكول Napster	15
الشكل 7	نموذج الإجرئية المعتمد وتوضيح التطوير التزايدى	22
الشكل 8	مخطط حالات الاستخدام -أ	27
الشكل 9	مخطط حالات الاستخدام -ب	28
الشكل 10	مخطط حالات الاستخدام -ج	28
الشكل 11	مخطط حالات الاستخدام -د	29
الشكل 12	مخطط حالات الاستخدام -هـ	29
الشكل 13	مخطط حالات الاستخدام -و	30
الشكل 14	مخطط المفاهيم الخاص بالنظام P2P HiastShare	38
الشكل 15	مخطط المفاهيم الخاص بنظام إدارة المخدم المركزي	39
الشكل 16	معمارية نظام HiastShare	42
الشكل 17	المكونات الرئيسة لنظام P2P HiastShare	43
الشكل 18	بنية رسالة بروتوكول HiastShare من الند إلى المخدم	45
الشكل 19	بنية بروتوكول HiastShare من المخدم إلى الند	45
الشكل 20	مخطط الصفوف لنواة اتصال الزبون	46
الشكل 21	مخطط الصفوف لنواة اتصال المخدم	47
الشكل 22	الكيانات المشتركة بين تطبيق الزبون وتطبيق المخدم المركزي	50
الشكل 23	مخطط قاعدة معطيات المخدم المركزي	52
الشكل 24	بنية MVC وآلية عملها	53
الشكل 25	آلية إنشاء مقبس بين مخدم وزبون	58
الشكل 26	بنية النظام من وجهة نظر زبون-مخدم	59
الشكل 27	واجهة التطبيق الرئيسية	60

- الشكل 28 واجهة إعدادات التطبيق ..... 61
- الشكل 29 واجهة استعراض الملفات المشاركة ..... 61
- الشكل 30 واجهة الطلبات العامة ..... 62
- الشكل 31 واجهة تحميل ملف ..... 62
- الشكل 32 واجهة عرض الرسائل المتروكة عندما كان المستخدم غير نشط ..... 63
- الشكل 33 واجهة المخدم المركزي ..... 64
- الشكل 34 واجهة إدارة تطبيق المخدم المركزي ..... 66

## قائمة الجداول

جدول 1	فوائد ومحددات البنى المختلفة لشبكات الند للند	10
جدول 2	ملخص الرسائل المستخدمة في بروتوكول Napster	15
جدول 3	أهم مصطلحات بروتوكول BitTorrent	18
جدول 4	المهام المنجزة خلال سير المشروع مع الفترات اللازمة	23
جدول 5	أنماط الرسائل المتبادلة بين الند والمخدم في نظام P2P HiastShare	49
جدول 6	أنماط الرسائل المتبادلة بين الند والند الآخر في نظام P2P HiastShare	49

## مقدمة

أصبحت أنظمة مشاركة الملفات من الند للند Peer to Peer من الأمور المعهود بها ولم تعد وليدة عهد، حيث أصبح هناك عدّة أنظمة ومعماريات مألوفة مثل: Napster و Gnutella و Freenet التي خطفت الأنظار على مثل هذا النوع من الأنظمة وأصبحت تتمتع بشعبية واسعة.

إنّ الهدف الرئيسي من هذه الأنظمة هو توزيع الملفات على مختلف عقد الشبكة، وهذا ما يختلف عن الأنظمة التقليدية التي تعتمد طريقة زبون-مخدّم لنقل الملف، حيث تكون جميع الملفات موجودة في عقدة مركزية وحيدة، وتكون جميع التنقلات دائماً بين الزبون والعقدة المركزية فقط، أما في أنظمة نقل الملفات من الند للند يمكن أن تظهر التنقلات بين أي عقدتين في الشبكة حيث تؤدي كل عقدة دور مخدّم وزبون في الوقت نفسه [1].

إن التطبيق المقدم في هذا العمل P2P HiastShare هو تطبيق آخر من تطبيقات مشاركة الملفات من الند للند، لكن لماذا نقدّم تطبيق آخر في هذا المجال في ظل وجود عدد من الأنظمة والمعماريات المعروفة في هذا المجال؟ الجواب هو أن P2P HiastShare يملك بعض الخصائص الفريدة التي تميّزه عن غيره من الأنظمة المشابهة.

**أولاً-** صُمّم التطبيق للاستخدام من قبل مجموعة من الأشخاص تعرف بعضها بعكس الأنظمة الموجودة حيث يستطيع أي شخص الدخول إلى النظام.

**ثانياً-** صُمّم التطبيق ليكون دليل معروضات Catalogue على عكس كثير من الأنظمة التي تعتمد الطلب للسؤال عن ملف معين، حيث يوجد دليل للملفات المشاركة عند كل عقدة في هذا النظام، لذا قد يكون البحث عن ملف غير ضروري لتحميله (مع إمكانية البحث عن ملف معين)، هذا له فائدة عندما يكون المستخدم في حيرة من أمره عن الملفات التي يريد تحميلها ويفضل اختيار ملف من قائمة الملفات على أن يبحث عن شيء معين ربما غير موجود في النظام.

**ثالثاً-** لا يقتصر النظام على مشاركة الملفات بين المستخدمين بل يمكن لأي مستخدم بدء محادثة مع أي مستخدم آخر داخل إلى النظام online أو ترك رسالة له في حال عدم دخوله offline.

**رابعاً-** إمكانية أي مستخدم ترك طلب عام لملف معين يرغب أن يشاركه له أحد المستخدمين الذين يملكون هذا الملف.

**خامساً-** اعتماد نظام تحفيز لمشاركة الملفات عن طريق توزيع نقاط على المستخدم في حال مشاركة ملف مفيد على النظام (قام بتحميله المستخدمون الآخرون) أو حذف نقاط من المستخدم عند تحميله لملف مشارك.

**سادساً-** الوصول إلى بعض الملفات وخاصة ذوو الأحجام الكبيرة من الشبكة الداخلية يوفر في استخدام عرض الحزمة المخصص للإنترنت.

من هنا تأتي الحاجة إلى بناء تطبيق مصمّم وفق الاعتبارات السابقة قادر على تلبية احتياجات المستخدمين في تبادل الملفات بشكل فعال بالإضافة إلى جميع الخدمات المساهمة في إنجاح هذه العملية.

## الهدف من المشروع

يهدف هذا المشروع إلى تصميم نظام مشاركة ملفات يعمل ضمن الشبكة الداخلية Intranet يعتمد على بنية الند لند للاستفادة من ميزات هذه البنية، وتنفيذه برمجياً، مع إتاحة خدمات أخرى مثل الدردشة وترك طلبات عامة بالإضافة إلى اقتراح نظام تحفيز لمشاركة الملفات.

ولإنجاز هذا الهدف جرى العمل وفق ما يلي:

- 1- دراسة مرجعية عن بروتوكولي Napster و Bit Torrent وعن طرق التحفيز وتحسين الأداء المستخدمين فيهما.
- 2- تصميم كلاً من قواعد المعطيات والتطبيق والبروتوكولات المطلوبة بشكل تزايد.
- 3- تطوير النظام حسب المتطلبات الموضحة لاحقاً.
- 4- بناء منظومة إدارة للنظام تسمح بمراقبة كافة المناقالات التي تجري خلاله ولها صلاحيات كاملة عليها.
- 5- الاختبار والتوثيق.

## مخطط تنظيم التقرير

نقدّم في هذا العمل مراحل بناء النظام من خلال تقسيمه لعدة فصول ندرج فيها بتوضيح مراحل إنجاز وتحقيق النظام، نبدأ بالفصل الأول الذي نطرح فيه مجموعة المتطلبات الوظيفية وغير الوظيفية للمشروع، ومن ثم في الفصل الثاني نقدم دراسة مرجعية حول بنية شبكات الند لند وأهم الأنظمة التي تستفيد من ميزات هذه البنية كما نقدّم في هذا الفصل دراسة عن بروتوكولي Napster و Bit Torrent، ثم نوضح في الفصل الثالث مجموعة المهام المطلوب تنجزها ضمن فترة المشروع مع التوزيع الزمني لهذه المهام، في حين يتضمن الفصل الرابع تحليلاً للنظام باستخدام UML، نعرض فيه مخططات لحالات استخدام النظام بالإضافة إلى المخططات التسلسلية. بعد دراسة وتحليل النظام نتقل في الفصل الخامس لتصميم النظام من خلال عرض حواريات العمل المصممة من أجل تحقيق وظائف النظام. نتقل في الفصل السادس لتوضيح كيفية تنفيذ النظام بأجزائه المختلفة، حيث نعرض مراحل تنجز النظام ونوضح بعض الأمثلة والاختبارات.



## الفصل الأول

### متطلبات المشروع

## SRS (Software Requirements Specifications)

نقدم في هذه الفصل تفصيلاً لمتطلبات الزبون الوظيفية وغير الوظيفية.

## 1.1- المتطلبات الوظيفية

نهدف من خلال هذا المشروع إلى تصميم وتطوير نظام مشاركة ملفات يعمل ضمن الشبكة الداخلية Intranet من نوع ند لند، بشكل يضمن نقل ملفات كبيرة نسبياً (كتب إلكترونية وأنظمة OS وتحديثات عليها وبرامج ويندوز وآنرويد المختلفة وبرامج مشهورة وغيرها) ضمن انترانت داخلية من نوع P2P مع إدارة مركزية مشابهة لنظام Napster وفق المتطلبات التالية:

1. الدخول إلى النظام Login عن طريق مخدم مركزي.
2. إرسال لائحة بجميع المستثمرين النشطين وغير النشطين Online or Offline إلى المستثمر الداخل إلى النظام تتحدث هذه اللائحة خلال زمن التشغيل Run-Time بمجرد تبديل حالة مستثمر من نشط إلى غير نشط أو العكس.
3. يشارك المستثمر الملفات التي يريد مع الجميع public حيث يرسل للمخدم اسم الملف وحجمه ولحمة عنه.
4. إمكانية البحث عن الملفات المشارك عليها عن طريق المخدم المركزي الذي يعيد أسماء المستثمرين الذين شاركوا الملف وحالتهم (نشط أو غير نشط)، حيث يستطيع الزبون الاتصال بحواسيب المستثمرين وطلب الملف منها.
5. إمكانية المستثمر استعراض الملفات التي حملها أو التي شارك بها والتعديل عليها.
6. يقوم النظام بجعل المستثمر الذي يحمل ملف بذرة seeder لهذا الملف (أي يصبح الملف مشارك عليه للجميع من حاسبه) وذلك لتخفيف العبء عن الحاسب الأصلي خاصة إذا كان الملف مطلوب بكثرة.
7. اعتماد نظام تحفيز يعتمد على توزيع العلامات لحث المستثمرين على المشاركة، حيث يتم توزيع علامات على المستثمر عندما يشارك أو عندما يتم التحميل من حاسبه أو حذف علامات عندما يحمل ملف من مستثمر آخر، ويأخذ بعين الاعتبار توزيع علامات بدائية Initial Bonus تراعي كون مجموع العلامات الكلي لجميع المستثمرين يتزايد باستمرار.
8. يسمح النظام بال دردشة Chatting بين المستثمرين النشطين Online أو ترك رسالة نصية إذا كان المستثمر المطلوب غير نشط Offline.
9. يسمح النظام للمستثمر بتعميم طلب للجميع يصف فيه ملف ما غير مشارك به (الرسالة من الشكل: يرجى ممن لديه ملف File Name المشاركة به على النظام...).
10. يتيح النظام إيقاف الاتصال والإعادة في أي وقت لاحق مع تحديث البذور باستمرار.
11. إمكانية عرض Top 10 and Bottom 10 للمستثمرين ذوو العلامات العليا والدنيا.
12. يتيح النظام للمستثمر معرفة مجموع نقاطه الحالية.

## 2.1- المتطلبات غير الوظيفية

### الأمان Security:

المصادقة Authentication: تتم كافة العمليات في التطبيق على مستوى التحقق من المستخدم verified user عن طريق اسم المستخدم وكلمة المرور.

### الواجهة User Interface:

1. يجب أن تكون واجهة التطبيق سهلة وواضحة يستطيع الأشخاص العاديين استخدامها.

2. يبنى التطبيق كتطبيق مكتبي Desktop Application.

### التوسع Scalability:

التوسع على مستوى التطبيق Application Scalability: القدرة على إضافة مجتزئات برمجية جديدة بسهولة كون التطبيق خدومي فيمكن إضافة خدمات جديدة حسب الحاجة أو الطلب.

### الأداء Performance:

يجب أن يكون أداء النظام جيداً لكي يتمكن من الاستجابة على الطلبات بشكل سريع.

### التوافر Availability:

يجب على المخدم المركزي التوافر الدائم لتخدم المستخدمين في أي وقت.



## الفصل الثاني

### الدّراسة المرجعيّة

### Survey

نقدم في هذا الفصل دراسة حول بنية شبكات الند لند وبعض أنظمة مشاركة الملفات التي تستفيد من هذه البنية ثم نوضح بروتوكولي Napster و Bit Torrent وأخري نلخص الدروس المستفادة من هذه الدراسة.

## 1.2- مفهوم شبكة الند لند

أخذ مفهوم شبكات الند لند peer to peer networks بالتطور حديثاً رغم أنه مفهوم قديم. يعود السبب في ذلك إلى انتشار صناعة الأفلام والموسيقى وحاجة المستخدمين إلى مشاركة مثل هذه الأنواع من الملفات بين بعضهم البعض. يُعرف هذا المفهوم بشكله البسيط: "شبكة الند لند هي شبكة من العقد nodes لها نفس الصلاحيات والمسؤوليات وتستطيع كل عقدة في الشبكة إنشاء قناة اتصال مع أي عقدة أخرى ضمن هذه الشبكة" [2]. يعدّ هذا المفهوم بديلاً عن المفهوم التقليدي للمعمارية زيون-مخدّم التي تقتصر على وجود مخدّم وحيد يخدّم العديد من المستخدمين. عند العودة إلى تاريخ شبكات الند لند وتتبع ظهورها، نجد أنه في ستينيات القرن الماضي كان التنفيذ الأولي للإنترنت وهو (ARPANET) كشبكة من نوع الند لند حيث جميع العقد متساوية الصلاحيات والمسؤوليات. وبالتالي قد يتناسب تعريف هذه الشبكات مع أكثر من سيناريو، يعدّ مخدّم نطاق الأسماء مثال جيّد لتوضيح المزج بين شبكات الند لند التقليدية والنموذج الهرمي للوصول إلى مالك المعلومة، يوجد لذلك تعريف أكثر دقة [3]: "يمكن أن يُطلق على الشبكة الموزعة شبكة الند لند، إذا كانت تتشارك في مواردها سواء الماديّة (مثل الطابعات، ومساحة التخزين، وعرض الحزمة وغيرها)، أو المنطقية (كالخدمات) بهدف إنجاز مهام محددة مثل التشارك بالملفات، وتبادل الرسائل الفورية instant messaging والمعالجة الموزعة distributed computing".

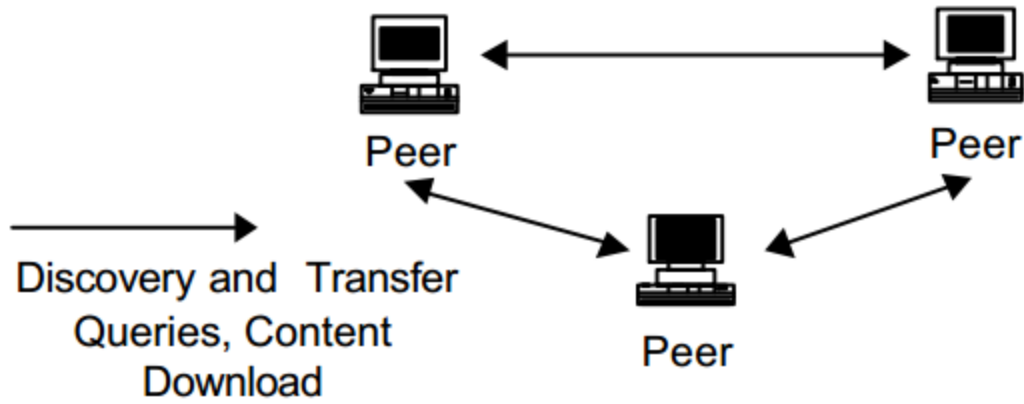
## 2.2- تصنيف شبكات الند لند

يمكن أن تصنف شبكات الند لند تبعاً لاستخداماتها إلى [4]:

- مشاركة الملفات
- اتصالات هاتفية
- بث الوسائط المتعددة (صوت، فيديو)

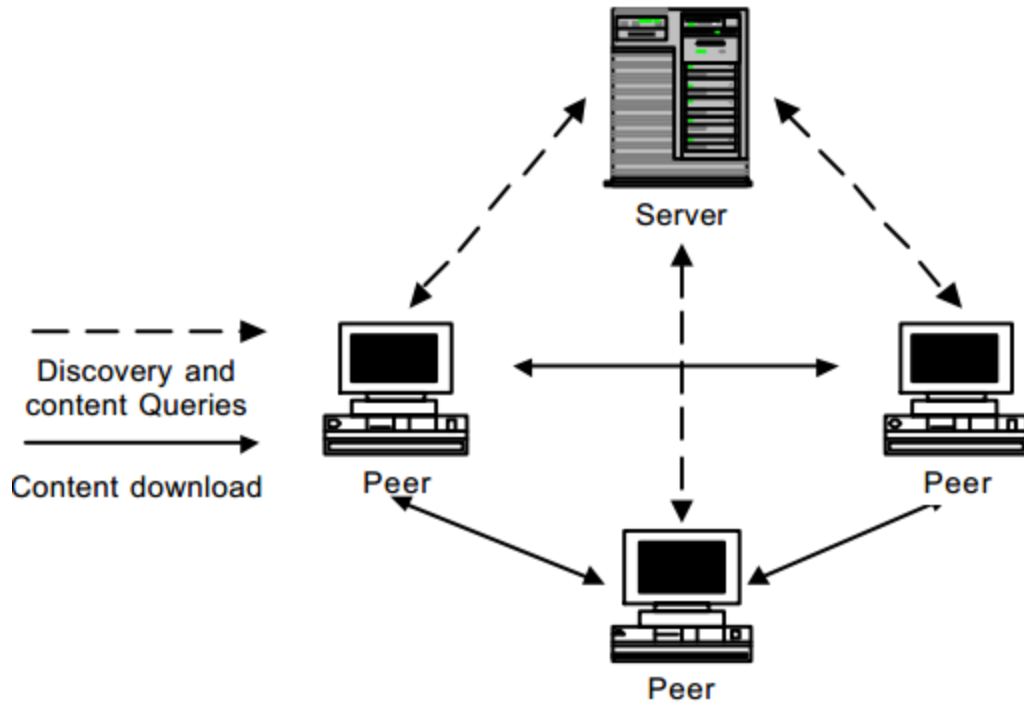
ويمكن أن تصنف تبعاً لدرجة مركزيتها إلى [5]:

- **بنية صافية pure P2P architecture**: نقول عن شبكة أنها صافية إذا تكونت من عناصر يعمل كل منها كمخدم وزبون في نفس الوقت دون وجود مخدّم مركزي مسؤول عن إدارة الشبكة وتنظيمها. وتمتاز هذه الشبكة بالوثوقية وانخفاض التكلفة بسبب عدم الحاجة إلى تجهيزات بمواصفات مميزة [5]. يوضح الشكل (1) بنية الشبكة الصافية.



الشكل 1 بنية الشبكة الصافية

- البنية الهجينة **hybrid P2P architecture** أو **Server-Mediated**: تشمل وجود مخدّم مركزي للتحكم بالشبكة وبناءها. يوضح الشكل (2) بنية الشبكة الهجينة.



الشكل 2 بنية الشبكة الهجينة

تتلخص فوائد ومحددات البنية الصافية والبنية المهجينة في الجدول (1).

بنية الند لند	الفوائد	الحدوديات	أمثلة
البنية الصافية	- لا يوجد مخدّم مركزي.	- استهلاك أكثر	Guntella -
	- سماحية عالية للأخطاء.	- لموارد الشبكة.	Freenet -
	- معمارية بسيطة	- محدودة التوسعية.	
البنية المهجينة	- استهلاك أقل لموارد الشبكة.	- معتمدة على مخدّم مركزي.	Napster -
	- قابلة للتوسع.	- أقل سماحية للأخطاء	

جدول 1 فوائد ومحددات البنى المختلفة لشبكات الند لند

تطرقنا من خلال بحثنا عن البنية الأنسب لتطبيقنا P2P HiastShare إلى دراسة بعض معماريات أنظمة مشاركة الملفات بتصنيفاتها المختلفة. سنناقش في الفقرات اللاحقة في هذا الجزء من الدراسة الأنظمة المختلفة من حيث معماريتها، ثم سنقارن بينهم من حيث الأداء و الموارد المطلوبة وسماحية الخطأ والقدرة على التوسع.

## 3.2- النموذج اللامركزي في مشاركة الملفات من نوع ند لند

في هذا النموذج من نماذج مشاركة الملفات من نوع ند لند، يكون لدى جميع العقد في الشبكة نفس الصلاحيات ونفس المسؤوليات. يكون التواصل بين العقد متناظر بمعنى أن كل عقدة من الشبكة تلعب دور زبون ومخدّم بنفس الوقت ولا يوجد أي علاقة سيّد-تابع بين هذه العقد. يمكن لأي عقدة في الشبكة أن تلعب دور زبون بالنسبة للعقدة التي تحمّل منها الملفات وفي نفس الوقت يمكن أن تلعب دور مخدّم بالنسبة للعقد التي ترفع لها الملفات. فالبرمجيات التي تعمل على كل عقدة تحوي كلاً من وظائف الزبون والمخدّم معاً.

لا تستخدم الشبكات اللامركزية في مشاركة الملفات من نوع ند لند مخدّم مركزي لتتبع جميع الملفات المشاركة في الشبكة وهذا يعد نقطة قوة من ناحية الإتاحة وبنفس الوقت يعد نقطة ضعف من ناحية الفعالية والتوسعية. يوجد فهرس يحوي معلومات عن الملفات المشاركة عند كل عقدة أي يخزن محلياً عند كل عقدة. وبالتالي لإيجاد ملف مشارك في هذا النموذج، تسأل العقدة جميع العقد المتصلة معها والذين بدورهم يسألوا جميع العقد المتصلين معهم ليحصلوا على معلومات تدل على الملف المطلوب<sup>1</sup>. تُطبّق الأنظمة المختلفة سياسات مختلفة في آلية إيجاد الملف في سياق هذا النموذج.

<sup>1</sup> جاءت هذه الآلية كبديل أفضل عن البث الشبكي المتعدد IP Multicast، لعدم فعالية هذه الآلية في شبكات الانترنت Internet، والتسبب في كثير من مشاكل إيقاف الخدمة DOS سواء كانت أخطاء برمجية Bugs أو في نية الهجوم على الشبكة، في الشبكات الداخلية Intranet [5].

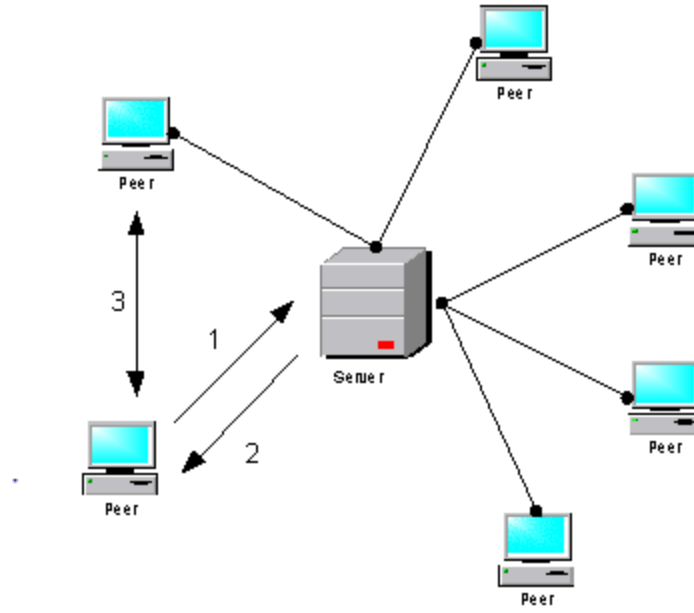


يعتمد نجاح الأنظمة اللامركزية لمشاركة الملفات بشكل كبير على نجاح آلية إيجاد الملفات المستخدم في هذا النظام. إن كلاً من التطبيقين Freenet و Gnutella يستخدمون هذا النموذج. يوضح الشكل (1) بنية شبكة هذا النموذج.

## 4.2- النموذج المركزي في مشاركة الملفات من نوع ند لند

يوجد في هذا النموذج مخدّم مركزي (أو مجموعة مخدّمات مركزية cluster) مسؤولة عن إدارة وتنظيم اتصال العقد [5] حيث يحوي هذا النموذج كلاً من النظام الصافي pure P2P ونظام مخدّم-زبون في نفس الوقت، لذلك يسمّى بالنموذج الهجين لمشاركة الملفات hybrid file-sharing system. يكون تبادل الملفات بين العقد كما في البنية الصافية pure بينما يكون البحث عن الملفات عن طريق المخدّم المركزي. يستخدم كل من التطبيقين Napster و OpenNap هذا النموذج. يحوي المخدّم المركزي في هذه التطبيقات دليل الملفات المشاركة من قبل كل عقدة مسجّلة في الشبكة. في كل مرة يسجّل المستخدم دخوله أو خروجه من شبكة Napster، يجري تحديث الدليل الموجود لدى المخدّم المركزي ليحذف أو يضم الملفات التي يشاركها المستخدم. يسجّل المستخدم دخوله إلى شبكة Napster عن طريق الاتصال بالمخدّم المركزي. لكي يحصل المستخدم على ملف معين يرسل طلب إلى المخدّم المركزي المتصل به، يبحث عندها المخدّم في قاعدة البيانات الموجودة لديه، حيث تحوي هذه القاعدة على معلومات الملفات المشاركة من قبل مستخدمي الشبكة المتصلين حالياً، ثم ينشئ قائمة بالملفات التي توافقت مع طلب المستخدم. تُرسل هذه القائمة كجواب على طلب المستخدم. يختار المستخدم الملف المرغوب من القائمة، ثم يفتح اتصال HTTP مباشر مع العقدة (المستخدم) التي تملك الملف. يُرسل الملف مباشرة من عقدة إلى عقدة أخرى (دون وجود وسيط). وبعبارة أخرى، فإن ملف الموسيقى MP3 لا يُخزّن أبداً على المخدّم المركزي، فالمخدّم لديه فقط معلومات تدل على الملف وليس الملف نفسه.

في النموذج المركزي المستخدم في تطبيق Napster، تحفظ معلومات الملفات المشاركة في النظام على المخدّم المركزي. فلا يوجد حاجة لسؤال كل مستخدم في الشبكة لاكتشاف وجود الملف. يستطيع الفهرس المركزي في Napster تحديد موقع الملفات في النظام بسرعة وفعالية. يجب على كل المستخدم أن يسجّل في المخدّم المركزي كي يستطيع دخول شبكة Napster. لذلك يضم الفهرس المركزي جميع الملفات المشارك بها في النظام من جميع المستخدمين المسجلين دخولهم Logged-on users. يوضح الشكل (3) بنية شبكة هذا النموذج وآلية طلب الملف [6].



الشكل 3 بنية شبكة Napster مع آلية طلب ملف فيها.

- 1- ترسل العقدة (المستخدم) طلب inquiry إلى المخدم.
- 2- يعالج المخدم الطلب ويجيب عليه بإرسال لائحة بالملفات المتاحة.
- 3- تحمّل العقدة الملف مباشرة من العقدة المشاركة للملف المطلوب.

## 5.2- مقارنة بين النموذج المركزي واللامركزي في مشاركة الملفات من نوع ند لند

من ناحية النموذج المركزي: إن آلية اكتشاف الملفات في النموذج المركزي أكثر فعالية وشمولية بسبب وجود فهرس مركزي يحوي معلومات عن الملفات المشاركة. يكون كذلك المخدم المركزي نقطة دخول أي عقدة إلى النظام، حيث يعتبر عنق الزجاجة في النظام، وبالتالي فإن فشل المخدم المركزي يؤدي إلى فشل الشبكة بأكملها. أمّا من ناحية المعالجة والتخزين، فنحتاج وجود مخدم فعال قادر على معالجة كافة طلبات البحث من مختلف العقد وقادر على استيعاب معلومات كافة الملفات المشاركة في النظام. وبالتالي تعتمد التوسعية في النموذج المركزي على قوة معالجة وتخزين المخدم المركزي.

من ناحية النموذج اللامركزي: تكون مسؤولية إيجاد الملف موزعة على كافة عقد الشبكة، فإذا خرجت إحدى العقد من الشبكة تبقى عملية البحث عن الملف قائمة بين العقد الباقية. لا يوجد كذلك في هذا النموذج نقطة فشل وحيدة تؤدي إلى فشل كامل الشبكة. لذلك فإن هذا النموذج هو الأفضل من ناحية القوة والانتاحية Availability مقارنة مع النموذج المركزي. كذلك، تكون فهرسة الملفات محلياً عند كل عقدة، وبالتالي تتطلب عملية إيجاد الملف بحث في كامل الشبكة، لذلك تصبح عملية إيجاد الملف في النموذج اللامركزي أقل فعالية مع توسع الشبكة وتسبب باستهلاك أكثر موارد الشبكة.

## 6.2- بروتوكول Napster<sup>2</sup>

تعتمد معمارية بروتوكول Napster [7] على النموذج المركزي Centralized Model في نظام مشاركة الملفات من نوع الند للند، إذ تحوي شبكة Napster على مخدم مركزي، لذا يقسم بروتوكول Napster إلى:

- تواصل ند - مخدم Peer to Server
- تواصل ند - لند P2P

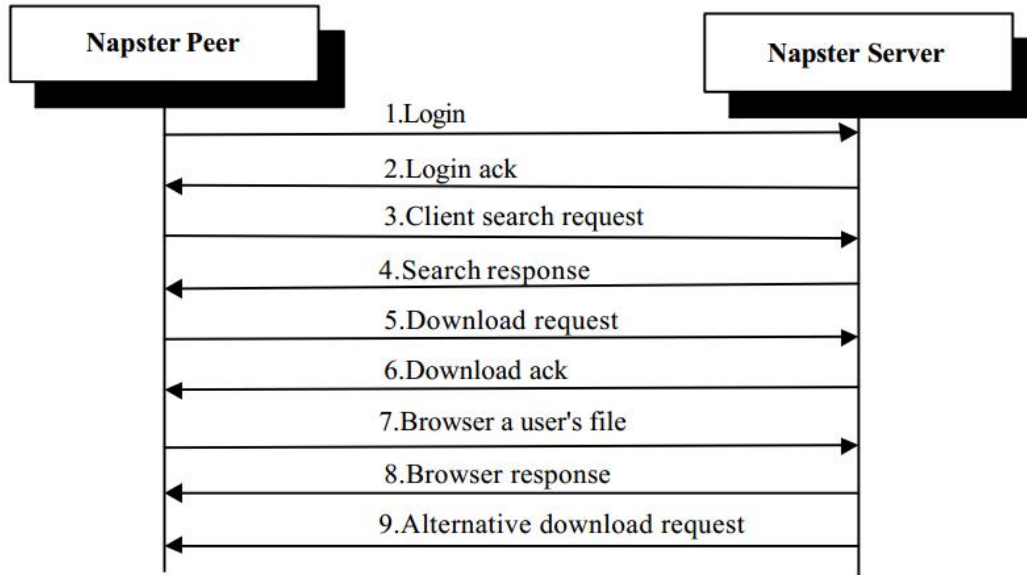
تتكون كل رسالة من وإلى مخدم Napster من ثلاثة حقول "الطول Length"، "النمط Function"، "الحمل Payload". يكون طول حقل "الطول" و "النمط" 2 Bytes.

- يصف حقل "الطول" طول "الحمل" في الرسالة (بالبايت).
- يصف حقل "النمط" نوع أو نمط الرسالة.
- يكون "الحمل" في الرسالة مرمز بترميز ASCII.

<Length> (2 Bytes)	<Function> (2 Bytes)	<Payload> (n Bytes)
-----------------------	-------------------------	------------------------

الشكل 4 بنية رسالة بروتوكول Napster

<sup>2</sup> في الحقيقة إن تطبيق Napster ليس مفتوح المصدر Open Source، لكن بُني تطبيق مشابه (OpenNap) له نفس الوظائف حاول تقليد بروتوكول Napster عن طريق الهندسة العكسية reverse-engineering (Eduard, 2002)



الشكل 5 الرسائل المتبادلة في بروتوكول Napster

يبدأ الزبون (النند) جلسته بتسجيل الدخول على مخدّم Napster، مما يسمح للمخدّم بحفظ قائمة بجميع المتصلين. لتسجيل الدخول على المخدّم يبدأ الزبون بإرسال رسالة "تسجيل دخول login" للمخدّم عن طريق إدخال الاسم وكلمة المرور، ويتنصت الزبون مباشرة على بوابة خاصة لبروتوكول Napster، عندها يرد المخدّم برسالة "قبول الدخول login ack" للإعلان عن نجاح تسجيل الدخول. يقوم الزبون أيضاً بالبحث عن الملف الذي يحتاج أن يحمله فيرسل أولاً رسالة "client search request" أو رسالة "Browse a user's file" الى المخدّم، عندئذٍ يرد المخدّم برسالة "Search response" ورسالة "Browser response" على الترتيب، هذه الرسائل تحوي معلومات مثل اسم المستخدم الذي يشارك الملف، واسم الملف، وحجم الملف. يمكن عندها طلب تحميل الملف بإرسال رسالة "Download request" إلى المخدّم. المخدّم سيرد برسالة "Download ack" حيث تحوي معلومات أكثر تفصيلاً عن الملف المحتمل. فإذا احتوت الرسالة "Download ack" على معلومات مفادها أن البوابة التي يتنصت عليها الزبون هي 'zero 0' عندها يرسل الزبون (طالب الملف) رسالة إلى المخدّم "Alternative download request" لكي يرسل بدوره رسالة إلى الزبون النند (الذي يستضيف الملف) تفيد بأن يفتح الاتصال هو ويرسل الملف الذي لديه. في هذه الحالة ينتظر الزبون (طالب الملف) حتى يتصل نذّه (مستضيف الملف) به على البوابة المخصصة. أما في الحالة التي لا يكون النند فيها محمي بجدار حماية يمكن للزبون الطالب إنشاء اتصال TCP للبوابة المشار إليها في رسالة "Download ack"، عندها ينبغي على النند الموافقة على فتح الاتصال بإرسال رسالة تحوي محرف واحد وهو '1' (ASCII 29). عندما يقرأ الزبون الطالب هذا المحرف سيرسل مباشرة

رسالة طلب بالملف الذي يريده (هذه الرسالة ترسل إلى الند) - أولاً ترسل "GET" بحزمة Packet واحدة، ثم يُرسل رسالة على هذه الصيغة:

<Nick>	<Filename>	<Offset>
--------	------------	----------

الشكل 6 رسالة طلب ملف في بروتوكول Napster

حيث:

- <Nick>: هي اسم المستخدم.
- <Filename>: هي الملف المراد تحميله.
- <Offset>: إزاحة تمثل رقم البايت التي يبدأ التحميل عنده.

عندها يرد الند المستضيف برسالة (الى الند الطالب) تحوي حجم الملف أو رسالة خطأ مثل "Invalid Request" أو "File not shared"، يلخص الجدول (2) الرسائل المستخدمة في بروتوكول Napster.

جدول 2 ملخص الرسائل المستخدمة في بروتوكول Napster

نوع الرسالة	وصف الرسالة
Login	تحتوي هذه الرسالة اسم المستخدم وكلمة المرور لتسجيل الدخول على مخدّم Napster. وأيضاً يتم فيها تحديد البوابة الخاصة وسرعة الاتصال لدى الزبون.
Login Ack	يرسلها المخدّم للزبون عند نجاح عملية تسجيل الدخول.
Client Search Request	يرسل الزبون هذه الرسالة إلى المخدّم للبحث عن ملف معين.
Search response	يرسل المخدّم معلومات عن الملف الذي يتم البحث عنه في حال وجود الملف المطلوب.
Download request	يطلب الزبون من المخدّم تحميل الملف بإرسال اسم المستخدم واسم الملف ومعلومات المضيف لهذا الملف.
Download ack	يرسل مخدّم Napster هذه الرسالة للزبون مع تفاصيل الملف المحمّل، حيث تحتوي عنوان IP Address الند

الذي يستضيف الملف، رقم البوابة التي يتنصت عليها المستضيف، وأيضاً تحوي سرعة اتصال المستضيف.	
يرسل الزبائن هذه الرسالة إلى المخدم لطلب عرض الملفات المشاركة من قبل مستخدم معين.	<b>Browse a user's files</b>
يرد المخدم على الرسالة السابقة بهذه الرسالة التي تحوي على تفاصيل الملفات المشاركة من قبل مستخدم معين.	<b>Browse response</b>
ترسل هذه الرسالة من الزبون إلى المخدم عندما يمتلك الزبون الملف المطلوب لكن لديه جدار حماية يمنع الزبون الند من فتح اتصال TCP معه.	<b>Alternate download request</b>

## 7.2- بروتوكول BitTorrent

هو أحد بروتوكولات الند للند لمشاركة الملفات عبر الانترنت، حيث يسمح بروتوكول [8] Bit Torrent بتبادل الملفات بين المستخدمين دون وجود وسيط عدا عن المتتبع Tracker وهو برنامج مستضاف عادةً على مخدم خاص، يقوم المتتبع بتنسيق عملية الاتصال ما بين الأنداد Peers. يقوم كل ند في نفس اللحظة بتحميل Download ورفع Upload البيانات إلى الأنداد الآخرين.

### 1.7.2- آلية عمل البروتوكول

يسمح بروتوكول Bit Torrent بتبادل الملفات كبيرة الحجم عبر الإنترنت دون التعرض لمشاكل كثرة الطلبات والضغط التي تتعرض لها المخدمات العادية، كما أنه يسمح للأجهزة ذات الموارد الضعيفة (كالهواتف الذكية مثلاً) بالمشاركة في عملية تبادل البيانات ذات الحجم الكبيرة.

تقوم فكرة هذا البروتوكول على تقسيم الملف الكبير إلى عدد محدد من الأجزاء الصغيرة المتساوية في الحجم تسمى قطع (Pieces)، ولها أحجام قياسية (32KB, 64KB, 128KB, 256KB, 512KB, 1MB, 2MB)، كما أن هذه القطع محددة بدورها بأقسام أصغر يمثل كل قسم 16kB تسمى Sub pieces. ثم يتم توزيع هذه القطع بين المستخدمين بشكل عشوائي، وكلما زاد عدد المستخدمين الذين يتبادلون الملف نفسه كلما زادت الفرص بالحصول على سرعة أكبر، والحصول على الملف كاملاً. لا بد من وجود وسيط بين المستخدمين، يقوم على تنسيق عملية التبادل بين المستخدمين، ويسمى المتتبع Tracker، ويعمل المتتبع على تسجيل معلومات عن عدد المستخدمين الكلي (الأنداد Peers)، عدد المستخدمين الذين يملكون الملف كاملاً (مزودو الملف Seeders)، عدد المستخدمين الذين مازالوا يحملون الملف ولما يكتمل (الحاملون leechers)، عدد مرات اكتمال تحميل الملف، كما يقوم المتتبع بتسجيل عناوين الأنداد ليسهل عليهم الاتصال ببعضهم.

ليكن ملف بحجم 3520MB، في البداية يجب إنشاء ملف بلاحقة torrent. باستخدام أحد البرامج المختصة بالتعامل مع هذه التقنية مثل µTorrent وأثناء إنشائه يتم تحديد حجم القطعة وليكن 4MB، عندئذٍ سيجزأ الملف إلى  $3520/4=880$  قطعة وكل قطعة تتألف من  $(1024*4)/16=256$  قسماً. لو كان حجم الملف 3521MB عندئذٍ:

$3521/4=880.25$  إذاً سيجزأ الملف إلى 881 قطعة وكل قطعة بحجم 4MB مما يؤدي إلى زيادة حجم التحميل بمقدار 4MB تقريباً أي أنك ستقوم بتحميل 3524MB ولكن الملف الأصلي يبقى كما هو دون أي تعديل. وأيضاً أثناء إنشاء الملف يجب تحديد عنوان المتتبع.

للبدء بمشاركة هذا الملف يجب توزيع الملف ذو اللاحقة torrent. على المستخدمين الراغبين بالحصول على الملف الأصلي، يمثل المستخدم الذي يملك الملف الأصلي في البداية المزود الأولي للملف (Initial Seeder)، ثم يبدأ بقية المستخدمون بالاتصال به وبدء عملية التحميل، وكل منهم في هذه الحالة يسمى (مُحمِّل Leecher) ويتم توزيع القطع بشكل عشوائي بين المستخدمين مما يسمح لكل مستخدم بتحميل القطع غير المتوفرة لديه من مستخدم آخر يملكها حتى لو لم يكن كلا المستخدمين قد أكمل تحميل الملف كاملاً. عند اكتمال تحميل الملف لدى أحد المستخدمين يتحول دوره من مُحمِّل leecher إلى مزود للملف Seeder، وتقاس "صحة" ملف Torrent بعدد المزودين المشتركين في الحشد الكلي Swarm (أي بعدد نسخ الملف الكاملة المتوفرة).

تسمح هذه التقنية بنشر نسخ كثيرة من الملفات الكبيرة خلال وقت قصير وبدون ضغط على المزود الأول للملف (الذي ربما يقوم بتوزيع الملف مرة واحدة فقط)، كما يمكن أن يعيش ملف Torrent لسنوات طالما هناك من يزوده.

## 2.7.2 - ملف torrent.

يحتوي هذا الملف على قسمين أساسيين، الأول هو قسم الإعلان Announce وفيه يحفظ عنوان المتتبع، والثاني هو قسم المعلومات Info، ويحتوي على أسماء الملفات وحجمها، وعدد القطع الكلي، بالإضافة إلى تليبيد Hash خاص بكل قطعة باستخدام التابع SHA-1، البرنامج العميل BitTorrent Client يبحث في هذين القسمين لدى إضافة ملف Torrent إليه ليبدأ عملية المشاركة.

يقوم العميل بعد الانتهاء من تحميل كل قطعة بالقيام باختبار الاكتمال مستفيداً من التليبيد Hash المسجلة في ملف Torrent، وذلك لضمان التحميل الخالي من الأخطاء، وفي حال فشل الاختبار يتم إعادة تحميل هذه القطعة من جديد، وهذا أحد أسباب زيادة حجم التحميل عن الحجم المحدد للملف.

### 3.7.2- أهمية بروتوكول BitTorrent

يختلف بروتوكول Bit Torrent عن طرق البروتوكولات الأخرى (FTP و HTTP) في عدة نقاط أساسية:

- ينشئ Bit Torrent عدة طلبات صغيرة الحجم عبر بروتوكول TCP متصلاً بعدة أُنْدَاد، بينما في الطرق العادية الأخرى يتم إنشاء طلب وحيد عبر بروتوكول TCP موجه إلى طرف وحيد.
- يحتمل بروتوكول Bit Torrent بطريقة عشوائية أو بطريقة "النادر-أولاً"، مما يرفع مستوى التوافرية، بينما الطرق الأخرى تحتمل بطريقة تسلسلية.

هذه النقاط تجعل بروتوكول Bit Torrent يستهلك القليل من موارد الموزعين، ويعطي توافرية عالية للملفات المشاركة، كما يتخلص من مشاكل الضغوطات العالية التي تتعرض لها المخدمات العادية. إلا أن لهذا البروتوكول بعض نقاط الضعف. نظرياً: قد يستغرق الوصول للسرعة القصوى للتحميل بعض الوقت، نظراً للوقت الذي تستغرقه عملية الاتصال بالأُنْدَاد، كما يمكن أن تتغير السرعة أثناء عملية التحميل صعوداً وهبوطاً تبعاً لحالة الاتصال بالأُنْدَاد. بينما في الطرق الأخرى يتم الوصول إلى السرعة القصوى في وقت قصير، كما تبقى السرعة ثابتة طوال فترة التحميل.

جدول 3 أهم مصطلحات بروتوكول BitTorrent

المصطلح	التعريف
محمّل Leecher	يطلق على الند الذي مازال يحمل الملف ولما يكتمل بعد
مزود ملف Seeder	يطلق على الند الذي أتمّ تحميل الملف كاملاً
ند Peer	وهو الاسم المشترك بين المحمل والمزود
متتبع Tracker	وهو المسؤول عن عملية تنسيق الاتصالات بين الأُنْدَاد
عميل Client	وهو البرنامج الذي يقوم بتنفيذ بروتوكول Bit Torrent
حشد Swarm	ويمثل عدد جميع الأُنْدَاد المساهمين حالياً في عملية المشاركة (عدد المحملون + عدد المزودون)
إعلان Announce	وهي العملية التي يقوم بها العميل، فيعلم المتتبع برغبته في الانضمام إلى الحشد، مزوداً إياه بالمعلومات اللازمة، متوقعاً رداً يتضمن عناوين أُنْدَاد آخرين للاتصال بهم وبدأ عملية المشاركة
قطع Pieces	وهي أجزاء الملف الأصلي، المتساوية الحجم



## 4.7.2- طرق الاتصال بين الأنداد

عن طريق المتتبع Tracker، وهو برنامج مثبّت على مخدّم مثلاً، يقوم بتسجيل عناوين الأنداد المشاركين حالياً بملف Torrent، ويؤمن الاتصال فيما بينهم،

باستخدام تقنية DHT (Distributed Hash Table) التي تعتمد على اللامركزية، بحيث يعتبر كل ند عبارة عن عقدة Node، ويتم تبادل القيم المسجلة في جدول التليبد بين العقد بفعالية عالية، تسمح هذه التقنية بمساهمة أعداد ضخمة من العقد في عملية المشاركة طريقة تبادل الأنداد PEX (Peer Exchange)، وهدفها تقليل الاعتماد على المتتبع، فيقوم ند متصل بند آخر بطلب عناوين الأنداد المتصل بها هذا الأخير لكي يستطيع الأول الاتصال بها أيضاً وهكذا.

تقوم بعض المتتبعات بجعل ملفات Torrent المتتبعة من خلالها معلّمة بعلامة خاص Private، وتسمى متتبعات خاصة، بحيث يتم إلغاء ميزتي DHT و PEX، مما يسمح لهذه المتتبعات بتتبع تفاصيل المشاركة لكل ند.

## 5.7.2- تحميل ومشاركة ملفات Torrent

يقوم المستخدم بالبحث في الأنترنت، عن طريق Google مثلاً، لإيجاد ملفات Torrent التي توافق رغبته، ثم يقوم بتحميلها وفتحها بأحد البرامج الداعمة لصيغة Torrent ومن هذه البرامج (BitTorrent, BitComet, µTorrent) والتي تعرف بالعميل The Client حيث يوصله هذا البرنامج بالمتتبع المسبق تحديده في ملف Torrent فيتسلم قائمة بالأنداد الذين يتبادلون أجزاء ملفات Torrent المحدد. يصبح المستخدم في هذه الحالة ند هو الآخر يتشارك أجزاء الملفات مع أقرانه. كل مجموعة من الأنداد تتشارك في أجزاء ملف Torrent تسمى الحشد Swarm.

## 8.2- المشاكل الموجودة في أنظمة مشاركة الملفات الحالية والتي من نوع ند لند

أصبحت مشاركة الملفات من الند لند شائعة في الآونة الأخيرة، لكن هناك جانب مظلم من الحكاية إذ يوجد بعض المشاكل الشائعة في هذه الأنظمة. وكانت اللامركزية واستقلالية المستخدم التي تؤمن التواصل من الند إلى الند لها النصيب الأكبر في طرح هذه المشاكل [9].

إنّ محتوى الملفات المشاركة من قبل المستخدمين قد تكون مشبوهة، حيث لا توجد سلطة تستطيع التحقق من المحتوى الضار.

أيضاً إنّ جودة خدمة التحميل يمكن أن تختلف تبعاً لجودة الاتصال اللامتجانسة heterogeneous connection qualities. حسب الدراسات فإنّ 35% من مستخدمي Gnutella لديهم حدود على سرعة الرفع Upstream حوالي 100Kbps، و 8% من المستخدمين فقط تبلغ سرعة الرفع عندهم 10Mbps، بينما الباقين الذين يشكلون 22% يملكون أقل من 100Kbps.

ومن المشاكل الشائعة أيضاً في أنظمة مشاركة الملفات هي مشكلة المتقاعس<sup>3</sup> Free Riding، حسب أحد الدراسات فإن 66% من مستخدمي Gnutella لا يشاركون أي ملف على النظام بينما 73% يشاركون أقل من 10 ملفات على النظام، وفي دراسة مشابهة على مستخدمي eDonkey فإن 68% من المستخدمين لا يشاركون ملفاتهم. من هذا المنطلق ينبغي على نظام الند لنـد أن يصمم بطريقة تضمن نمو الشبكة بحيث تشجع المستخدم الفعّال الذي يشارك ملفاته وتثبط عزيمـة المستخدم غير الفعّال.

إنّ معظم الملفات المشاركة على أنظمة مشاركة الملفات هي ملفات موسيقى أو فيديو، وإن معظمها له حقوق ملكية ونشرها يعدّ من انتهاك سياسة الخصوصية فتتعرض هذه الأنظمة إلى الملاحقة القانونية والدعاوي القضائية.

## 9.2- الدروس والعبر المستفادة من الدراسة المرجعية

الدرس الأول الذي نلاحظه من خلال دراستنا أنّه يوجد دائماً تسويات tradeoffs لتصميم بروتوكول شبكة الند لنـد. فتسوية النموذج المركزي هي سرعة اكتشاف الملف والتوسعية وتسوية النموذج اللامركزي هي الإتاحة العالية والمساهمة في الأخطاء.

اعتمدنا في تصميمنا لبناء التطبيق P2P HiastShare على النموذج المركزي لعدّة أسباب أهمها وجود خدمات أخرى بالإضافة إلى خدمة مشاركة الملفات، مثل وجود خدمة المحادثة بين المستخدمين ووجود نظام تحفيز يعتمد على النقاط لتشجيع المستخدمين على المشاركة.

الدرس الثاني الذي تعلمناه من خلال هذه الدراسة أن المستخدمين لأنظمة مشاركة الملفات من نوع ند لنـد لامتجانسين من حيث: سرعة اتصال، الوقت الذي يبقى فيه المستخدم نشط online time، كمية الملفات المشاركة [9]. لذا يجب أن يؤخذ عدم التجانس بين المستخدمين بعين الاعتبار في تصميمنا للتطبيق.

التطبيق P2P HiastShare موجه لمجموعة معيّنة من الناس وليس عام (ضمن شبكة داخلية) لذا المستخدمين يعدّون أكثر تجانساً homogeneous من المستخدمين الآخرين في تطبيقات مشاركة الملفات الأخرى. لذا جاء اختيار التصميم بالشكل الذي يخدم هذه المجموعة من المستخدمين على الشكل الأمثل، وكما ذكرنا سابقاً أنّه يوجد العديد من التسويات في تصميم شبكة الند لنـد، لذا علينا اتخاذ القرار بما يتناسب مع معرفتنا لمستخدمي النظام واحتياجاتهم.

<sup>3</sup> هي مصطلح اقتصادي الأصل للدلالة على الفرد الذي يستفيد من المصادر أو البضائع أو الخدمات دون أن يدفع أجراً لهذه المنفعة. وفي سياق حديثنا عن مشاركة الملفات فمعناه للمستخدم الذي يستفيد من تحميل الملفات للمشاركة على النظام دون أن يشارك هو بملفاته ليستفيد منها الآخرون.

## الفصل الثالث

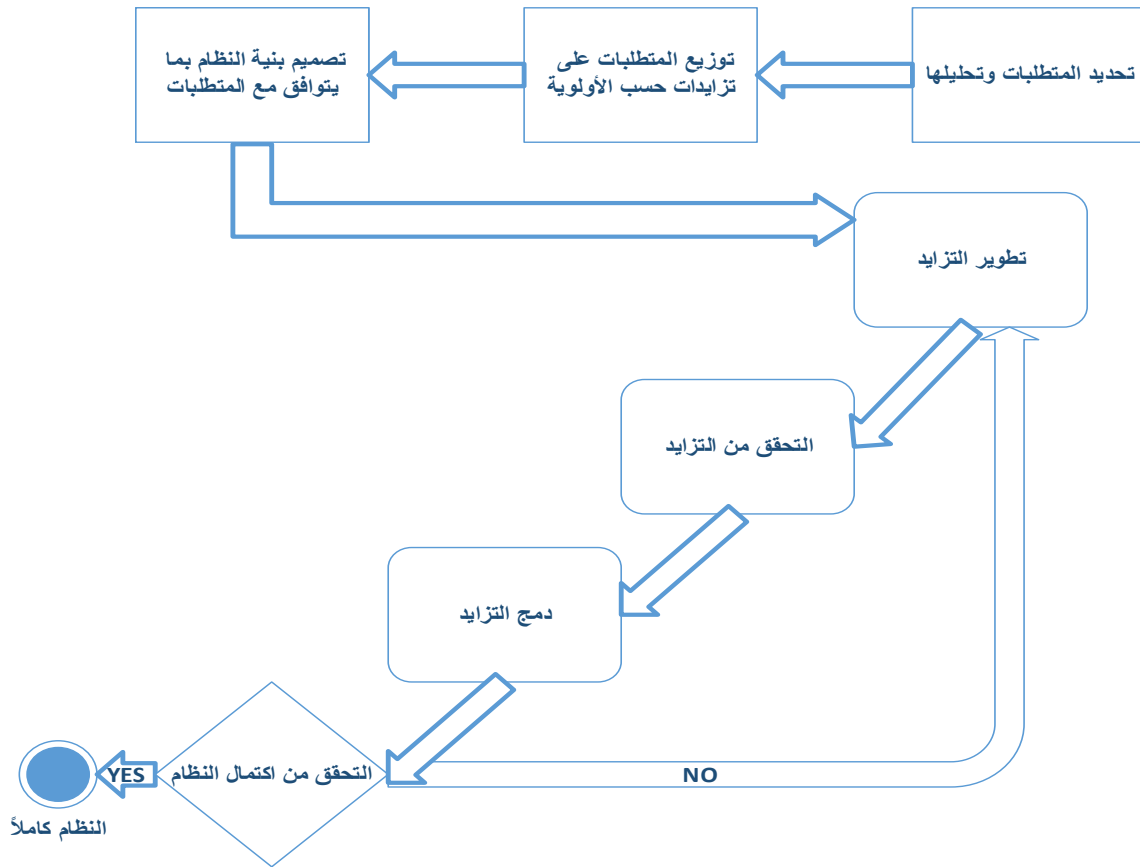
### خطة إدارة المشروع

## Project Management Plan

سنقوم بهذا الفصل بتعريف الاستراتيجية المتبعة في تنفيذ مراحل المشروع، كما سنتحدث عن الخطة الزمنية التي جرى العمل وفقها مع عرض الفترات الزمنية التقريبية التي تتطلبها كل مهمة من المهام.

### 1.3- نموذج الإجرائية المعتمد

اعتمدنا في هذا المشروع النموذج الشلالي Waterfall model بحيث يتم العمل في اتجاه واحد انطلاقاً من جمع المتطلبات والتحليل إلى التصميم والتنفيذ والاختبارات. جزء التطوير والتسليم إلى تزايدات increments حيث يؤمن كل تزايد جزءاً من الوظيفة المطلوبة.



الشكل 7 نموذج الإجرائية المعتمد وتوضيح التطوير التزايدي

إنّ بنية البروتوكول المصمّم وتجرده وظيفياً سمح بفصل الوظائف وتحقيق التطوير التزايدي ففي الخطة الزمنية كما سنرى لاحقاً يظهر واضحاً كيفية إدارة التزايدات وتقسيمها حسب حالات الاستخدام.

### 2.3- الخطة الزمنية

يعرض الجدول (4) المهام والعمليات المطلوبة لتنفيذ المشروع مرفقة بالمدّة التقريبية لإنجاز كل منها، ويهدف ذلك إلى توضيح وإظهار كلفة العمليّات من حيث الجهد والزمن، أيضاً يهدف إلى إعطاء فكرة عن الفترة الزمنية اللازمة لتنفيذ المهام ضمن هذا المشروع.

الفترة	النهائية	البداية	اسم المهمة
1d	5/8/2016	5/8/2016	دراسة مرجعية عن شبكت الند لند
2d	5/10/2016	5/9/2016	دراسة مرجعية عن مشاركة الملفات بشبكة الند لند
7d	5/18/2016	5/11/2016	دراسة عن بروتوكول Napster و Bit Torrent
3d	5/21/2016	5/18/2016	البدء في تحليل النظام ووضع مخطط الحالة
5d	5/24/2016	5/20/2016	إنشاء مخطط التتالي لأهم حالات الاستخدام
4d	5/28/2016	5/24/2016	العمل على تصميم بنية بروتوكول التخاطب بين المخدم المركزي والعقد
14d	6/11/2016	5/28/2016	البدء في تنفيذ نواة المخدم المركزي المسؤولة عن الاتصال بما يتوافق مع البروتوكول المصمم
7d	6/18/2016	6/11/2016	البدء في تنفيذ نواة العقدة (التي تمثل الزبون للمخدم المركزي) المسؤولة عن الاتصال بما يتوافق مع البروتوكول المصمم
3d	6/21/2016	6/19/2016	التأكد من صحة الاتصال بين العقد والمخدم المركزي
5d	6/26/2016	6/22/2016	تصميم قاعدة بيانات المخدم المركزي وتغليفها
7d	7/3/2016	6/27/2016	إنشاء أساسيات واجهة التطبيق
7d	7/10/2016	7/4/2016	إنشاء وظائف تسجيل الدخول والخروج في كل من العقدة ومقابلاتها في المخدم المركزي
14d	7/24/2016	7/11/2016	تنفيذ وظائف الدرسشة chatting في كل من العقدة ومقابلاتها في المخدم المركزي
4d	7/28/2016	7/25/2016	تنفيذ وظائف مشاركة ملف في العقدة ومقابلاتها في المخدم المركزي
5d	8/2/2016	7/29/2016	تنفيذ وظائف ترك رسالة من مستخدم لآخر في العقدة ومقابلاتها في المخدم المركزي
4d	8/7/2016	8/3/2016	تنفيذ وظائف إنشاء طلب عام واستعراض الطلبات العامة في العقدة ومقابلاتها في المخدم المركزي
7d	8/14/2016	8/7/2016	تنفيذ وظائف استعراض آخر الملفات المشاركة على النظام
8d	8/22/2016	8/14/2016	تنفيذ وظائف عرض البذور Seeders لكل ملف مشارك على النظام مع حالة البذور إذا كانت نشطة أو غير نشطة
10d	9/1/2016	8/22/2016	تنفيذ الوظائف اللازمة لعملية تحميل ملف بين عقدتين
7d	9/8/2016	9/1/2016	إنشاء موقع وب لإدارة ومراقبة المخدم المركزي
14d	9/22/2016	9/8/2016	كتابة التقرير النهائي
5d	9/27/2016	9/22/2016	تحسين عملية التحميل
1d	9/28/2016	9/27/2016	إنشاء العرض النهائي الذي يشرح العمل

جدول 4 المهام المنجزة خلال سير المشروع مع الفترات اللازمة



## الفصل الرابع

### تحليل المتطلبات

## SRA (Software Requirements Analysis)

نقدم في هذه الفصل تحليلاً مفصلاً لما ذكرناه في الفصل الأول من متطلبات (وظيفية وغير وظيفية)، حيث نورد مخططات حالات الاستخدام مع الوصف النصي الخاص بكل حالة، ونورد مخططاً أولياً للصفوف (مخطط المفاهيم).

## 1.4- مقدمة

نقوم في هذا الفصل بتوضيح النموذج السلوكي Behavioral model الذي يصف التفاعل مع النظام. يصف هذا الجزء من التحليل تفاعل المستخدمين مع النظام. يُعتبر المخطط الرئيسي الذي يعبر عن هذا الجزء من النمذجة مخطط حالات الاستخدام Use case diagram الذي يُعنى بتجميع متطلبات النظام وتوضيحها بشكل بياني يُجمع حالات استخدام النظام من وجهة نظر المستخدم لا النظام، كما يبيّن هذا المخطط مستخدمي النظام (الفاعلين، Actors، Stakeholders).

نقوم بتوضيح أهم حالات الاستخدام باستخدام سرد نصي Narrative يكون على شكل استمارة مُعرّفة الصيغة بشكل واضح، حيث يحوي هذا السرد على اسم الحالة والهدف منها والفاعلين المشتركين فيها وكافة السيناريوهات المتعلقة بها (السيناريو الناجح، السيناريوهات البديلة وسيناريوهات الخطأ)، كما يحوي على الشروط السابقة واللاحقة لهذه الحالة. يجري رسم مخطط التسلسل في نهاية السرد النصي الذي يعبر عن السيناريو المكتوب ضمن السرد بشكل بياني يحوي على الطلبات التي يرسلها المستخدم للنظام ورد النظام على هذه الطلبات (عن طريق رسائل).

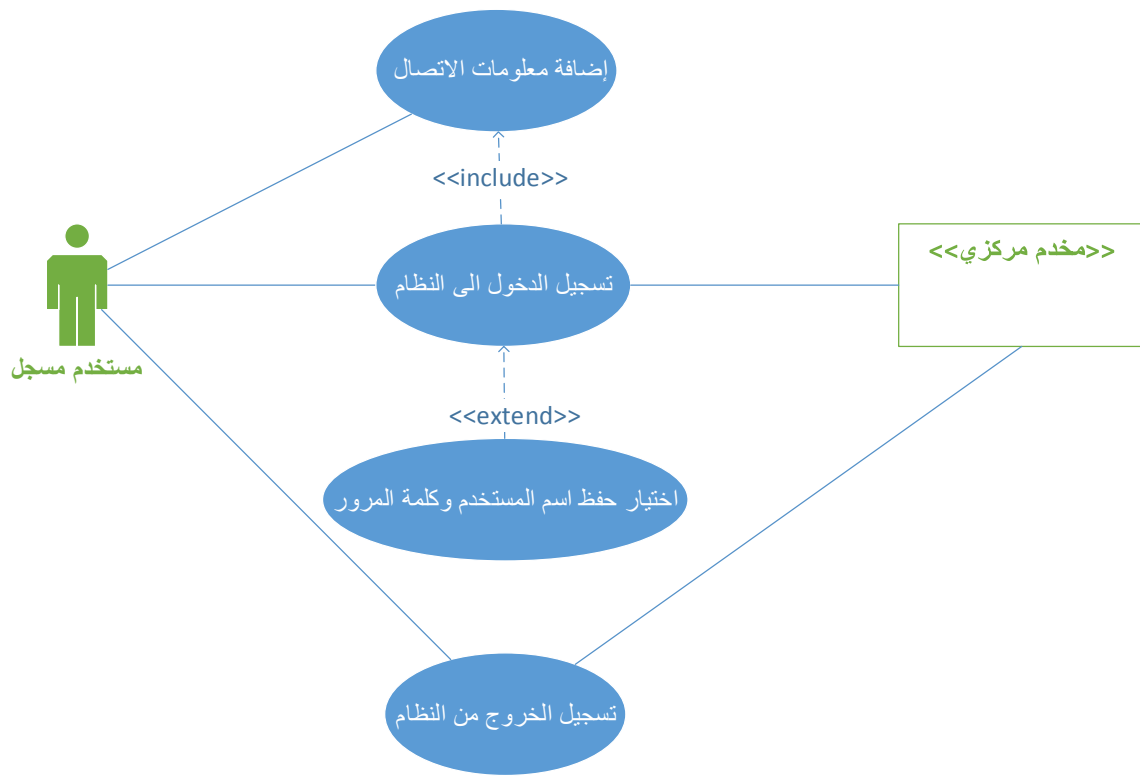


## 2.4- مخططات حالات الاستخدام

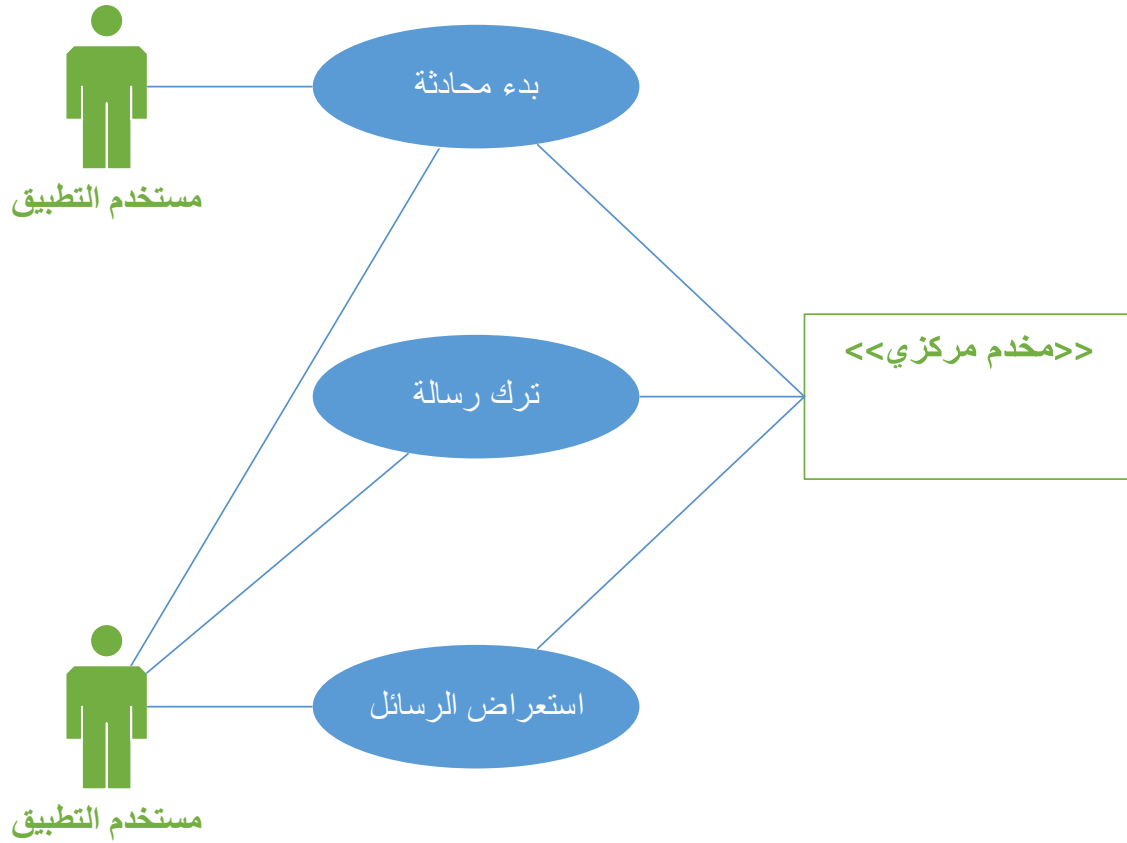
نستعرض في هذا القسم مخطط حالات الاستخدام مقسماً إلى عدة أقسام وذلك بغرض التوضيح فقط.

### أولاً- النظام من جهة الزبون

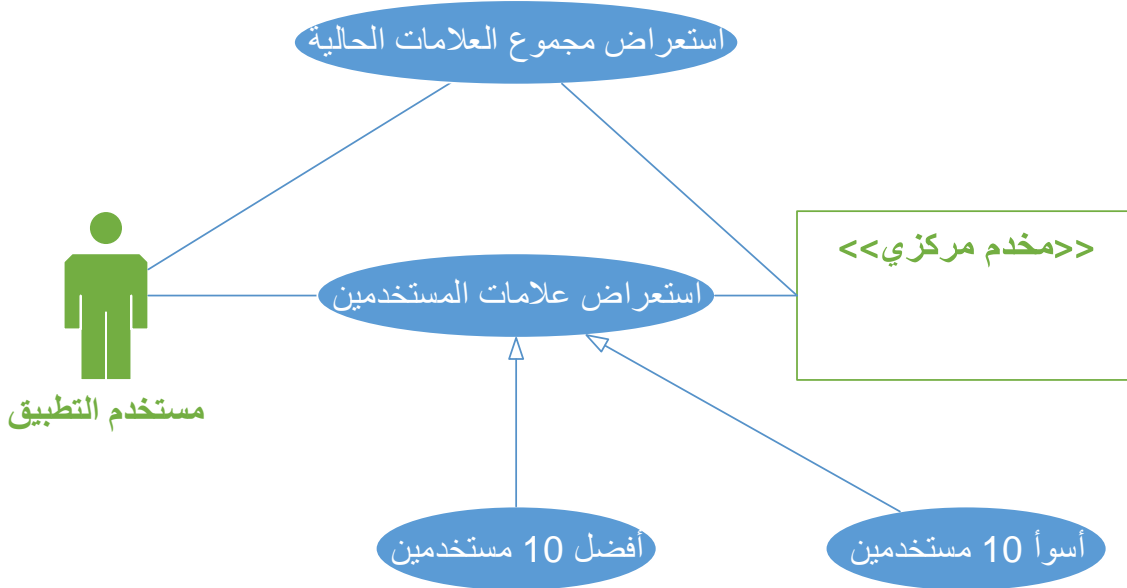
في هذه الحالة نعتبر المخدم المركزي فاعل مشارك participating actor الهدف، والمستثمر فاعل مبادر initiating actor يطلق استخدام النظام من أجل الوصول إلى هدف معين.



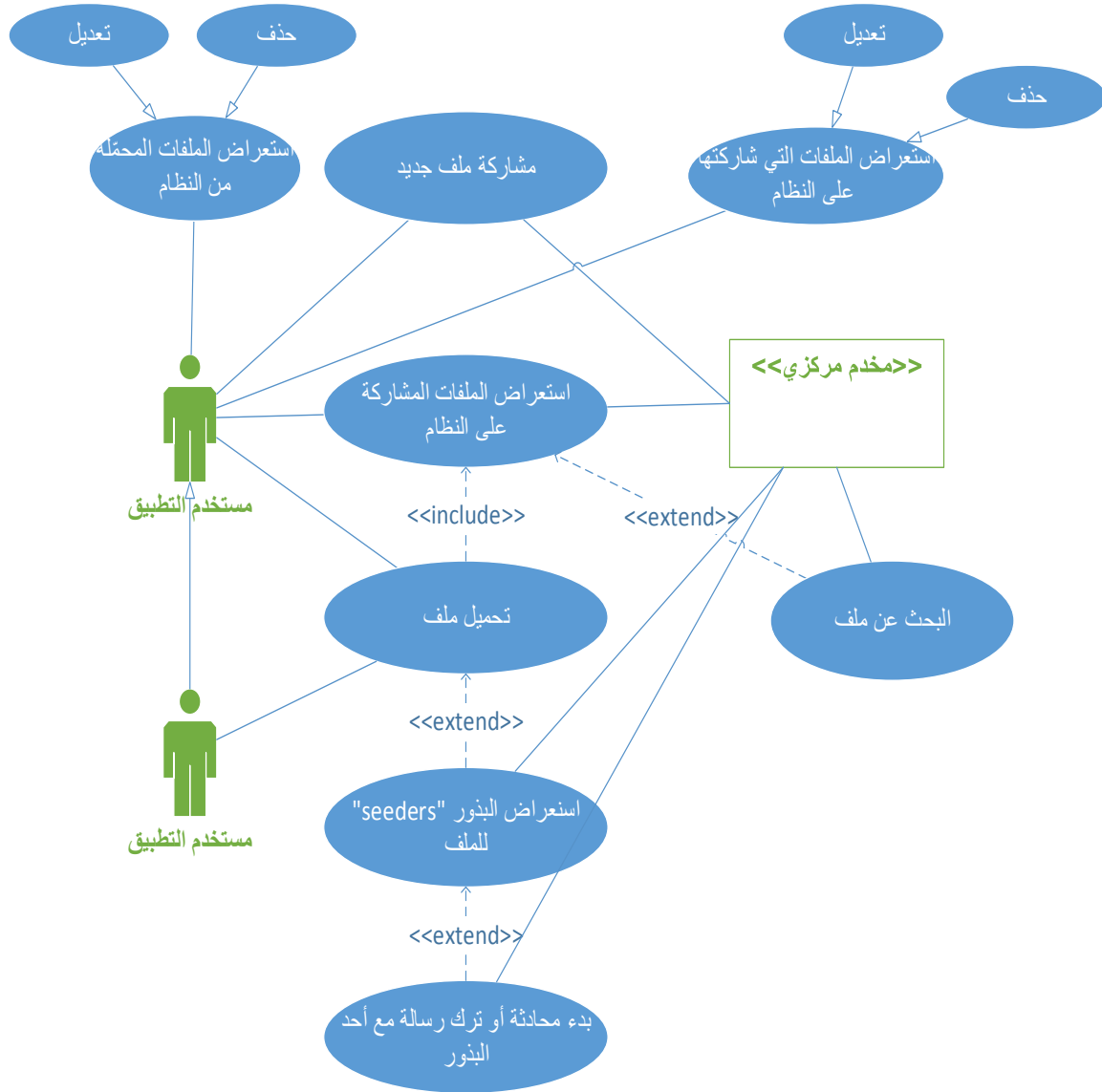
الشكل 8 مخطط حالات الاستخدام - أ



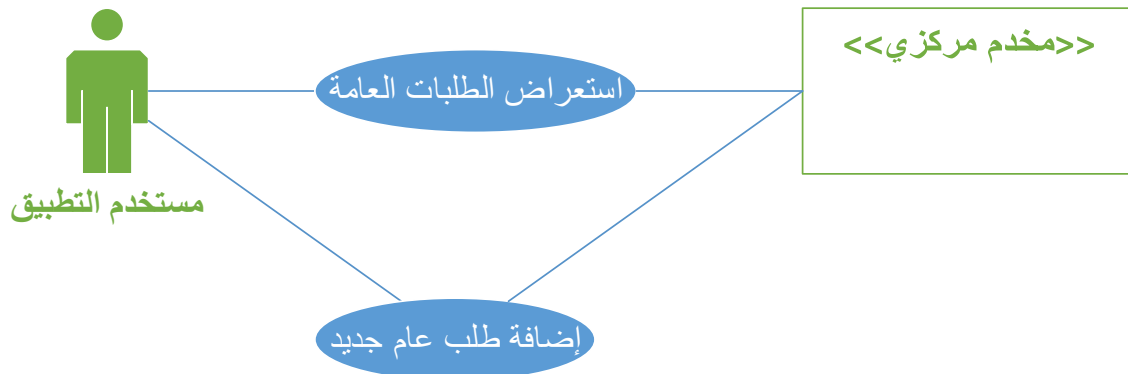
الشكل 9 مخطط حالات الاستخدام - ب



الشكل 10 مخطط حالات الاستخدام - ج



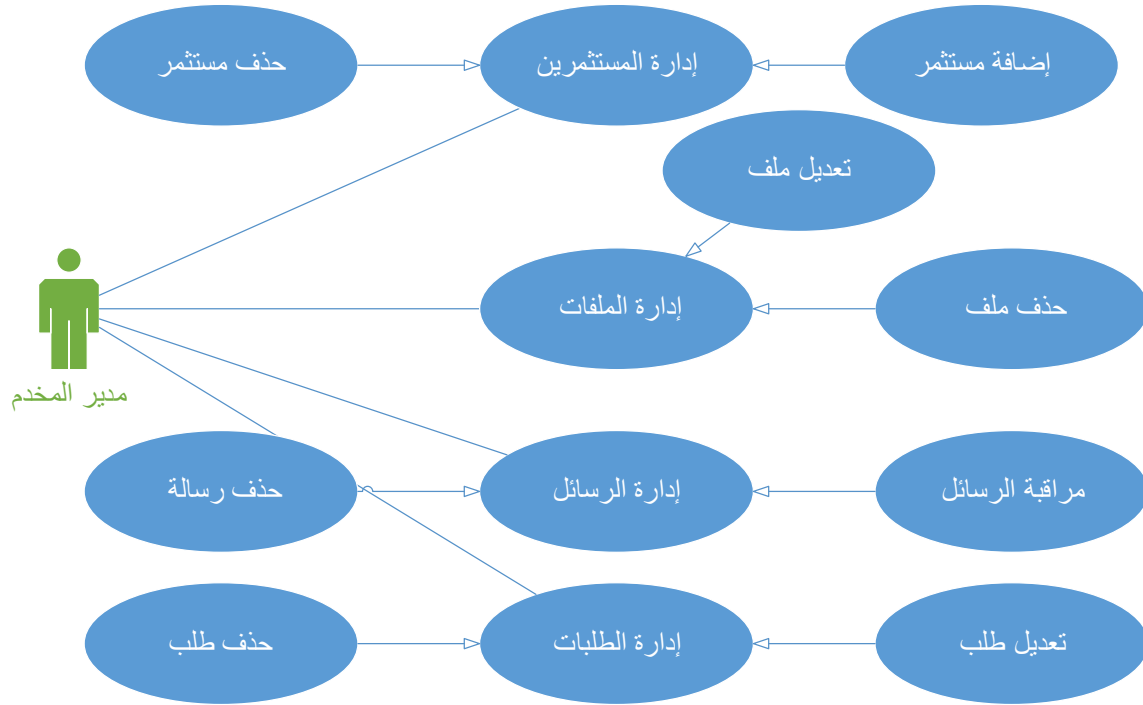
الشكل 11 مخطط حالات الاستخدام - د



الشكل 12 مخطط حالات الاستخدام - د

### ثانياً- النظام من جهة المخدم

نوضح في هذه الحالة، حالات الاستخدام على المخدم المركزي حيث يعتبر مدير المخدم هو الفاعل المبادر initiating actor.



الشكل 13 مخطط حالات الاستخدام - و

## 3.4- الوصف النصي لحالات الاستخدام ومخططات التسلسل

نقدم في هذا القسم الوصف النصي لأهم حالات الاستخدام المبينة في الشكل أعلاه.

### 1.3.4- تسجيل الدخول إلى النظام

النمط: أساسية.

**ملخص:** في هذه الحالة يقوم المستثمر بطلب الدخول إلى النظام ويدخل اسم المستخدم وكلمة المرور الخاصة، فيقوم النظام بالتحقق من صحة هذه المعلومات عن طريق المخدم المركزي، ثم يصبح المستثمر مستخدم للتطبيق قادر على استخدام وظائفه المختلفة.

**الفاعلون:** مستخدم مسجل في النظام.

**الظروف المسبقة:** المستخدم له حساب على النظام.

الظروف اللاحقة: يصبح المستثمر مستخدم داخل إلى النظام له الحق في استخدام وظائفه المختلفة.

السيناريو الأساسي الناجح:

الفاعلون	النظام
1. يطلب المستثمر تسجيل الدخول إلى النظام.	
2. يُظهر النظام قائمة بالحقوق التي يجب على المستثمر إدخالها.	
3. يُدخل المستثمر اسم المستخدم وكلمة المرور.	
4. يطلب النظام من المخدم المركزي التحقق من هوية المستثمر.	
5. يقوم المخدم المركزي بالتحقق من هوية المستثمر ويحجب بصحته.	
6. يقوم النظام بالسماح للمستثمر باستخدام الوظائف المختلفة عليه	

السيناريوهات البديلة والأخطاء:

1. خطأ في معلومات الاتصال مع المخدم المركزي

يبدأ هذا المسار عند النقطة 4 من السيناريو الأساسي الناجح.

5- يقوم النظام بمحاولة الاتصال بالمخدم المركزي للتأكد من هوية المستثمر لكنه لا يستطيع الوصول إلى المخدم.

6- يظهر النظام عبارة تفيد بوجود خطأ في معلومات الاتصال بالمخدم المركزي.

2. المستثمر غير مسجل بالنظام أو يوجد خطأ باسم المستخدم أو كلمة المرور

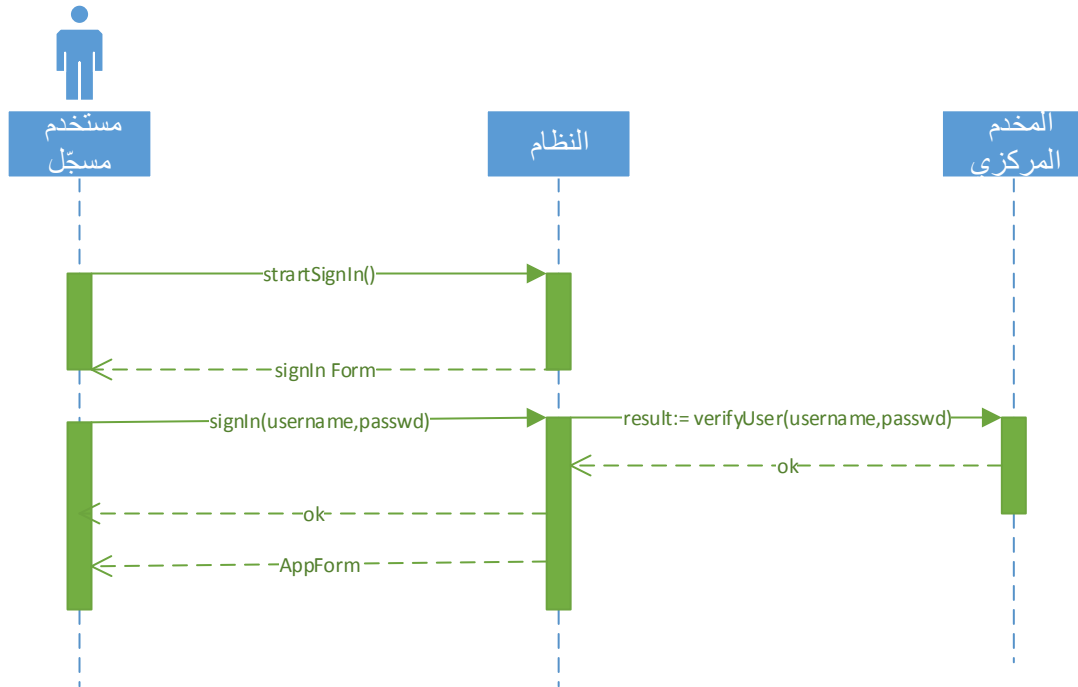
يبدأ هذا المسار عند النقطة 4 من السيناريو الأساسي الناجح.

5- يقوم المخدم المركزي بالتحقق من هوية المستثمر ويجب بعدم بصحته.

6- يظهر النظام عبارة تفيد بخطأ باسم المستخدم أو كلمة المرور، فيعود المسار إلى النقطة 3 لإعادة إدخال

معلومات المستثمر للتأكد من عدم الإدخال الخاطئ، أو تفشل حالة الاستخدام.

## مخطط التتالي للحالة:



## 2.3.4- بدء محادثة

النمط: أساسية.

ملخص: في هذه الحالة يمكن للمستخدم أن يبدأ بالمحادثة مع أحد المستخدمين النشطين على النظام.

الفاعلون: مستخدم التطبيق.

الظروف المسبقة: كلا المستخدمين الذين تجري بينهم المحادثة داخلين إلى النظام.

الظروف اللاحقة: لا يوجد.

السيناريو الأساسي الناجح:

الفاعلون	النظام
1. تبدأ هذه الحالة عندما يريد المستخدم بدء محادثة مع مستخدم آخر.	
2. يُظهر النظام النافذة الخاصة بالمحادثة مع المستخدم الآخر.	
3. يقوم المستخدم بكتابة رسالته ويضغط إرسال.	
4. يقوم النظام بإرسال اسم المرسل إليه والرسالة إلى المخدم المركزي.	
5. يرسل المخدم المركزي الرسالة إلى الشخص المطلوب.	

## السيناريوهات البديلة والأخطاء:

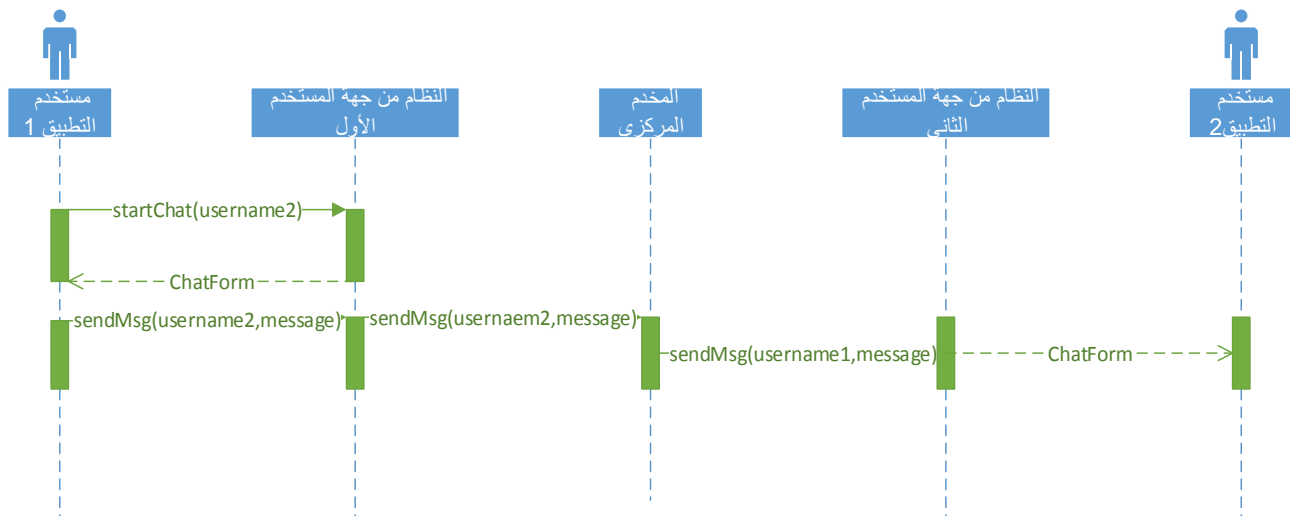
## 1. أصبح المستخدم الآخر غير نشط أثناء المحادثة

يبدأ هذا المسار عند النقطة 2 من السيناريو الأساسي الناجح

3- يقوم المخدم المركزي بإرسال رسالة للنظام تفيد بأن المستخدم أصبح غير نشط.

4- يقوم النظام بإشعار المستخدم بأن المستخدم الآخر أصبح غير نشط وينتقل الى حالة ترك رسالة من حالات الاستخدام.

## مخطط التتالي للحالة:



### 3.3.4- تحميل ملف

النمط: أساسية.

**ملخص:** في هذه الحالة يقوم المستخدم بطلب تحميل ملف من النظام كان قد اختاره من قائمة الملفات أو بحث عنه.

**الفاعلون:** مستخدم التطبيق.

**الظروف المسبقة:** المستثمر داخل إلى النظام، المستخدم اختار الملف المراد تحميله، يوجد بذرة "seed" على الأقل نشط لهذا الملف.

**الظروف اللاحقة:** يصبح الملف المطلوب موجود لدى المستخدم، يصبح المستخدم بذرة "seed" لهذا الملف، يتم حسم علامات من مجموع علامات المستخدم بحسب علامة الملف، يتم زيادة علامات على مجموع العلامات للبذرة (البذور) التي تحمل منها الملف، يُسجل الملف ضمن قائمة الملفات التي حملها المستخدم.

**السيناريو الأساسي الناجح:**

الفاعلون	النظام
1. يقوم المستخدم بطلب تحميل ملف معيّن.	
2. يقوم النظام بسؤال المخدم المركزي عن عناوين بذور هذا الملف.	
3. يعيد المخدم المركزي بذور هذا الملف للنظام.	
4. يقوم النظام باختيار بذرة (بذور) نشطة من بذور الملف.	
5. يطلب النظام من البذرة المختارة الملف المطلوب.	
6. تتحقق البذرة من وجود الملف ووجود رصيد كافٍ من العلامات لدى المحمل ويجب بالإيجاب.	
7. يعيد النظام حالة عملية التحميل للمستخدم لحين إتمام عملية التحميل.	
8. يقوم النظام بإضافة نفسه كبذرة لهذا الملف لدى المخدم المركزي.	



9. يقوم النظام بطلب توزيع العلامات من المخدم  
(زيادة علامات للمحمّل منه ونقصان من  
المحمّل) حسب علامة الملف.

السيناريوهات البديلة والأخطاء:

1. لا يوجد بذور نشطة حالياً للملف

يبدأ هذا المسار عند النقطة 3 من السيناريو الأساسي الناجح

5- لا يجد النظام أي بذرة نشطة للملف لبدء التحميل منها، فيعيد رسالة تفيد بعدم وجود بذور نشطة لهذا الملف حالياً.

2. تغيرت حالة البذرة إلى غير نشطة أثناء عملية التحميل

يبدأ هذا المسار عند النقطة 6 من السيناريو الأساسي الناجح

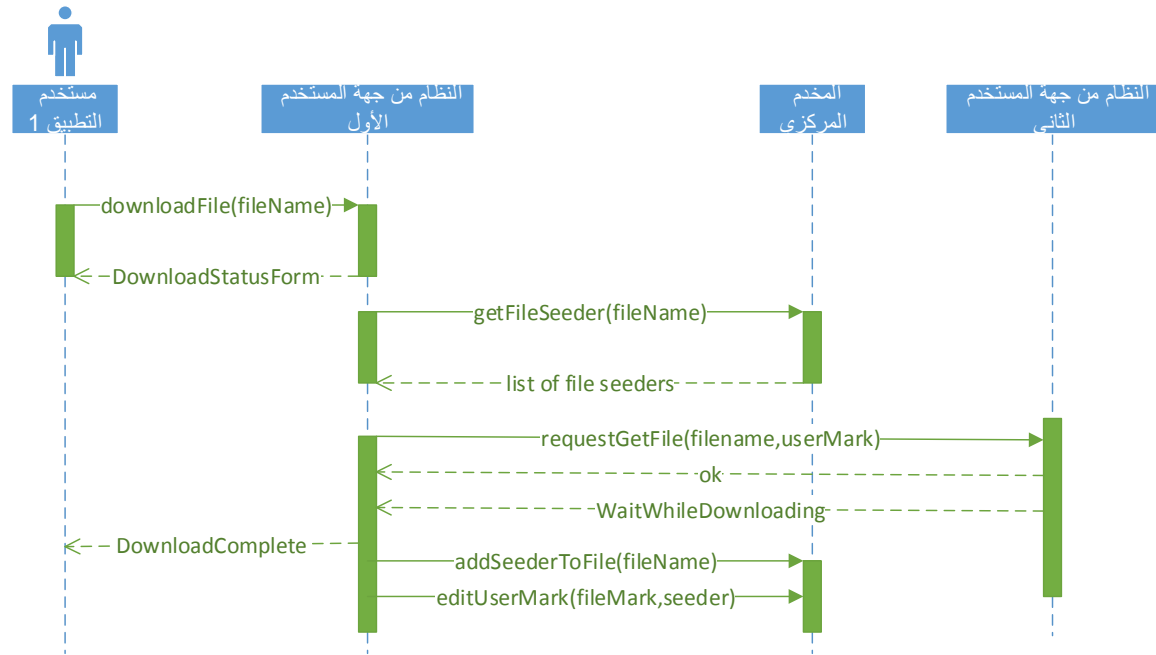
7- يقوم النظام بحفظ حالة التحميل ويطلب من بذرة نشطة أخرى إكمال عملية التحميل (إن وجد) أو الإكمال لاحقاً.

3. لا يوجد رصيد كافٍ من العلامات عند طالب الملف (المحمّل)

يبدأ هذا المسار عند النقطة 5 من السيناريو الأساسي الناجح

6- يظهر النظام عبارة تفيد بعدم وجود رصيد كافٍ من العلامات لتحميل الملف، وتفشل عملية التحميل.

مخطط التالي للحالة:



#### 4.3.4- مشاركة ملف جديد

النمط: أساسية.

ملخص: في هذه الحالة يقوم المستخدم بمشاركة ملف جديد.

الفاعلون: مستخدم التطبيق.

الظروف المسبقة: المستثمر داخل الى النظام.

الظروف اللاحقة: تصبح معلومات الملف المشارك على المخدم المركزي.

السيناريو الأساسي الناجح:

الفاعلون	النظام
1. يقوم المستخدم ببدء حالة مشاركة ملف.	
2. يظهر النظام النافذة الخاصة بمشاركة ملف جديد.	
3. يقوم المستخدم باختيار الملف المراد مشاركته ويدخل وصف عنه (إذا رغب) ويدخل علامة الملف المرغوبة.	

4. يقوم النظام بإرسال معلومات الملف المشترك إلى المخدم المركزي.

5. يقوم المخدم المركزي بتسجيل معلومات الملف المشترك به وتسجيل المستخدم كبذرة لهذا الملف.

6. يظهر النظام رسالة للمستخدم بنجاح عملية المشاركة.

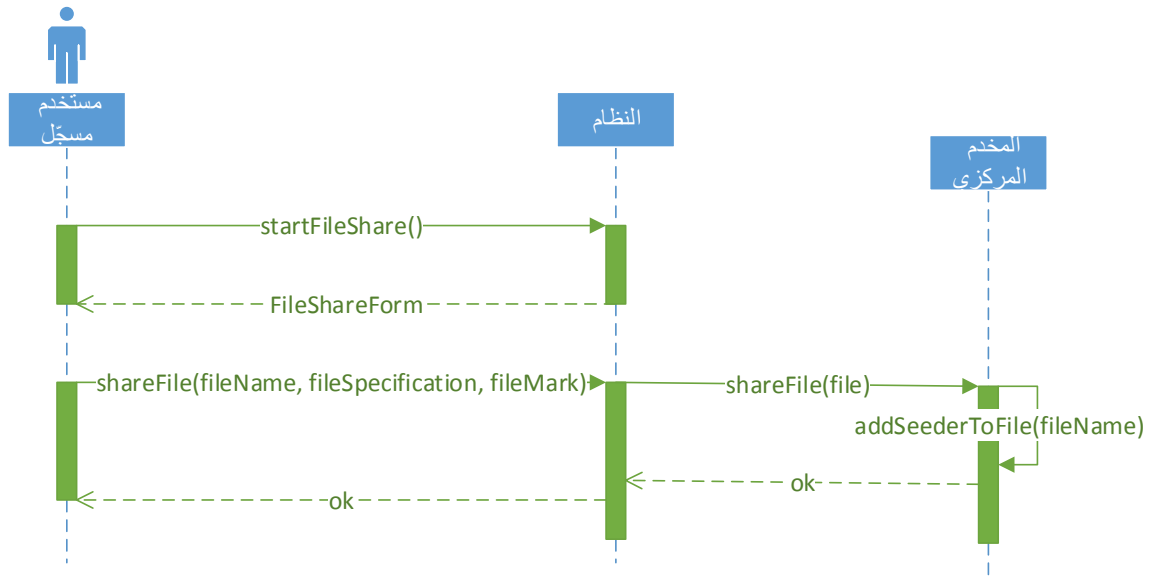
السيناريوهات البديلة والأخطاء:

4. الملف موجود مسبقاً:

يبدأ هذا المسار عند النقطة 4 من السيناريو الأساسي الناجح

5- يقوم المخدم المركزي فقط بتسجيل المستخدم المشترك كبذرة لهذا الملف.

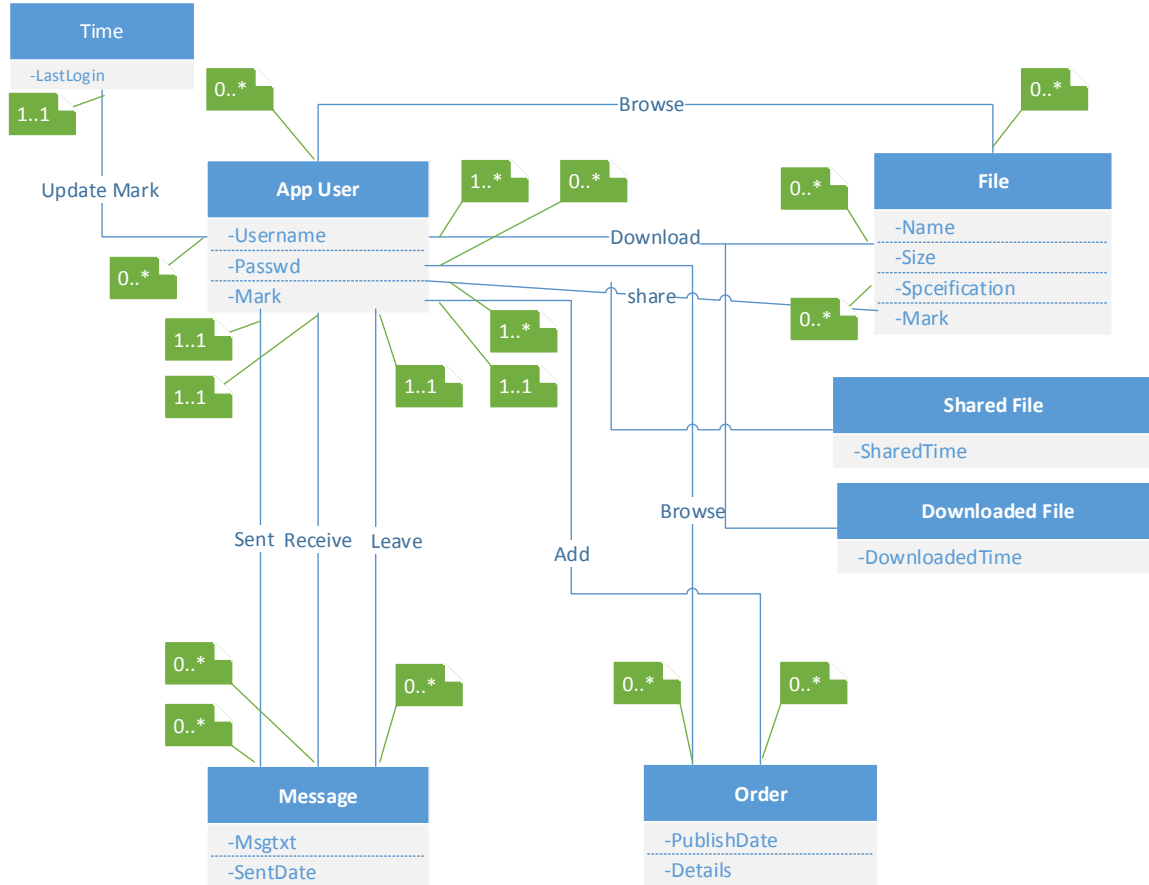
مخطط التالي للحالة:



## 4.4- مخطط المفاهيم (مخطط الصفوف الأولي)

في هذا القسم مخططاً أولياً للمفاهيم الموجودة في النظام.

### 1.4.4- مخطط المفاهيم الخاص بالنظام P2P HiastShare



الشكل 14 مخطط المفاهيم الخاص بالنظام P2P HiastShare

سنشرح فيما يلي بعض المفاهيم وبعض العلاقات المبينة في الشكل السابق.

**App User**: هو المستخدم النهائي للنظام له اسم وكلمة مرور ومجموع علامات، يستخدم الاسم وكلمة المرور للدخول إلى النظام ويستخدم علاماته لتحميل الملفات المشاركة على النظام.

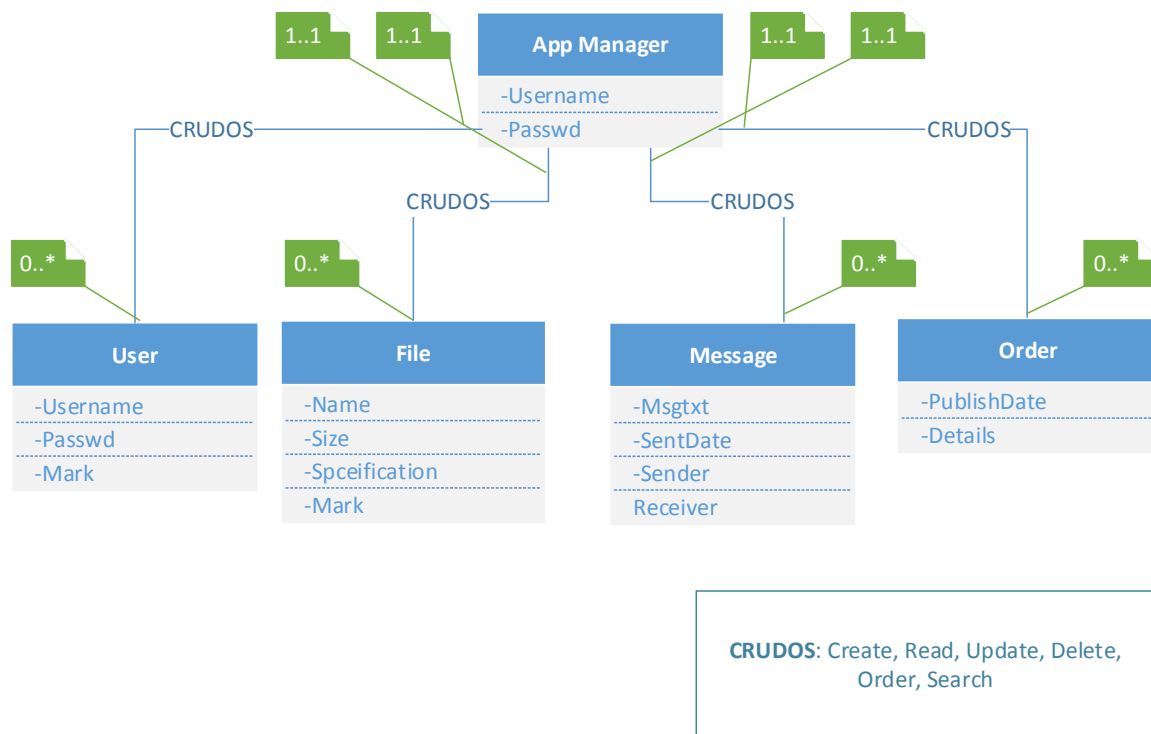
**File**: الملف المشارك أو المحتمل حيث يستطيع المستخدم تحميل ملف أو مشاركة ملف أو استعراض الملفات. له عدة واصفات: الاسم، الحجم، لحظة عن الملف، العلامة المقدرة للملف تسند هذه الواصفات للملف عندما يشاركه أحد المستخدمين.

**Message:** الرسالة المتبادلة بين المستخدمين، يمكن لأي مستخدم للنظام فتح محادثة مع أي مستخدم نشط وترك رسالة للمستخدم غير النشط.

**Order:** الطلب العام، يستطيع أي مستخدم ترك طلب عام حول ملف يحتاج أن يشارك به مستخدم آخر، فما عليه إلا أن يكتب تفاصيل طلبه ليرى بقية المستخدمين هذا الطلب ويتجاوبوا معه في حالة وجود الملف لديهم.

**Time:** يعد الزمن مفهوم من مفاهيم النظام، إذ يساهم في زيادة علامة المستخدم عند كل تسجيل دخول مضى عليه أكثر من 24 ساعة، ويعد هذا المفهوم عاملاً محفزاً لتشجيع المستخدمين على دخول النظام وكسب العلامات وبالتالي زيادة معدل المشاركة والتحميل.

#### 2.4.4- مخطط المفاهيم الخاص بنظام إدارة المخدم المركزي



الشكل 15 مخطط المفاهيم الخاص بنظام إدارة المخدم المركزي

وسنشرح فيما يلي بعض المفاهيم وبعض العلاقات المبينة في الشكل السابق.

**App Manager:** هو مدير المخدم المركزي الذي يشرف على جميع المناقالات التي تجري في النظام. فمثلاً يستطيع المدير حذف ملف مخالف مشارك على النظام (ينتهك سياسة الخصوصية مثلاً)، أيضاً من أهم صلاحياته إضافة مستخدمين إلى النظام

.

## الفصل الخامس

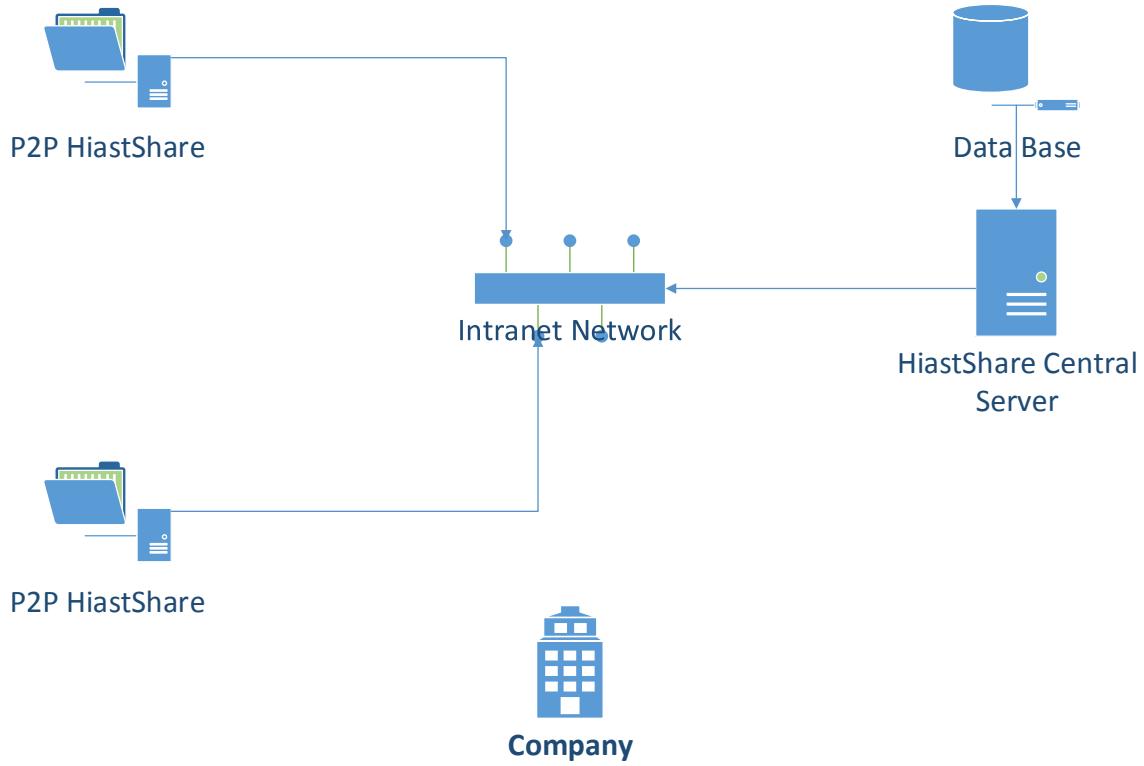
### تصميم النظام

## SD (System Design)

نقدم في هذه الفصل التصميم المعتمدة للنظام، مع تفسير وشرح تفصيلي لكل تصميم.

## 1.5- التصميم الرئيسي

كما ذكرنا في الدراسة المرجعية إن تصميم شبكة الند لند يخضع للعديد من التسويات بما يتناسب مع احتياجات مستخدمي النظام. وذكرنا أنه سوف نعتمد النموذج المركزي لمشاركة الملفات من نوع ند لند وبالتالي سيكون بنيان النظام كما في الشكل (16).



الشكل 16 معمارية نظام HiastShare

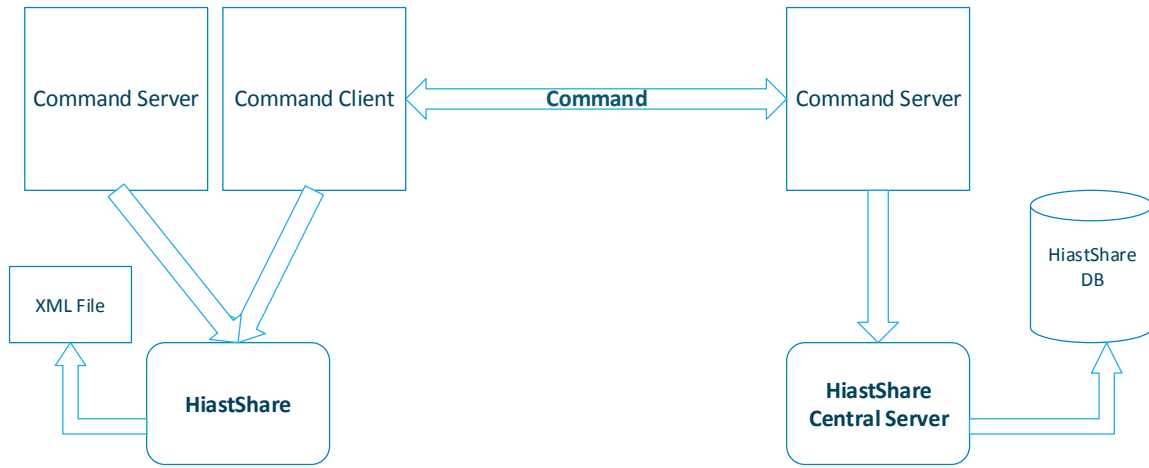
من خلال شرحنا التالي سوف يتبين سبب استخدام هذه المعمارية (المركزية) لبناء النظام. بما أنّ النظام يهدف إلى إرسال رسائل نصية بين المستخدمين وترك رسالة في حال كان المستخدم غير نشط فإن المخدم المركزي يفيد في مراقبة وتخزين كافة الرسائل المتبادلة، ولكنه سيكون عبارة عن عنق زجاجة **Bottle neck** في تصميم النظام، أي سيعمل المخدم المركزي كموزع **Dispatcher**، أي يقوم المستخدم بإرسال رسالته إلى المخدم الذي يقوم بدوره بإعادة هذه الرسالة إلى الطرف المستقبل الذي يتبادل الرسائل فقط مع المخدم. أما بالنسبة لتبادل الملفات فيتصل الطرف الأول بالمخدم ليحصل على معلومات الاتصال الخاصة بالطرف الثاني، ليقوم بعدها الطرف الأول بالاتصال بشكل مباشر بالطرف الثاني وتبادل الملف معه بشكل مباشر. لأنه سيكون الوقت اللازم لإجراء عملية التبادل عن طريق المخدم أكبر من الوقت اللازم لتبادل نفس الملف بين الطرفين بشكل مباشر في حالة الشبكة المثالية. أي سيعمل المخدم المركزي في هذه الحالة كمخدم أسماء **Naming Server**. وبالنسبة للطلبات العامة **Orders** (طلب مشاركة ملف معين من المستخدمين الآخرين) فستقتصر عملية المخدم على تخزين



الطلبات العامة والرد على طلبات المستخدمين لاستعراض الطلبات العامة، أي ستكون العلاقة زبون-مخدم بين المستخدم والمخدم المركزي.

## 2.5- المكونات الرئيسية للنظام P2P HiastShare

صُمم P2P HiastShare بشكل رئيسي على مفهوم غرضي التوجه object-oriented، حيث اعتمد على سبعة مكونات رئيسية كما هو موضح في الشكل (17).



الشكل 17 المكونات الرئيسية لنظام P2P HiastShare

- **Command**: بروتوكول التخاطب بين المخدم والزبون وسيتم شرحه لاحقاً بالتفصيل.
- **Command Server**: نواة المخدم المركزي المسؤولة عن عمليات الاتصال بما يتوافق مع بروتوكول التخاطب وهي نفسها مستخدمة كنواة للمخدم في كل عقدة<sup>4</sup>. يغلف هذا المكون وظائف طبقات Transport و Session و Presentation في نموذج OSI من جهة المخدم.
- **Command Client**: نواة العقدة (التي تمثل الزبون للمخدم المركزي أو الزبون لمخدم عقدة أخرى) المسؤولة عن الاتصال بما يتوافق مع بروتوكول التخاطب. يغلف هذا المكون وظائف طبقات Transport و Session و Presentation في نموذج OSI من جهة الزبون.
- **HiastShare DB**: قاعدة المعطيات المسؤولة عن تخزين معلومات عن المستخدمين وعن الملفات المشاركة والطلبات العامة والرسائل بين المستخدمين (الرسائل المتروكة في حالة كان أحد الأطراف غير نشط).

<sup>4</sup> هذا المكون مسؤول عن عمليات الاتصال فقط وفق بروتوكول التخاطب المصمم وليس له علاقة بنقل الملفات.

- **XML File**: هو ملف يمثل قاعدة بيانات محلية عند كل عقدة يُخزن فيه مثلاً مسارات **Paths** الملفات المشاركة في العقدة (في نظام الملفات على جهاز الحاسب الذي يمثل العقدة).
- **HiastShare Central Server**: هو المكوّن المسؤول عن معالجة اتصالات الزبائن والرد على طلباتها. يمثل هذا المكوّن التطبيق من ناحية المخدم المركزي.
- **HiastShare**: هو المكوّن المسؤول عن تزويد المستخدم بواجهة بيانية لمختلف الوظائف التي يتيحها النظام، وهو المسؤول عن عملية نقل الملفات. يمثل هذا المكوّن التطبيق من ناحية المستخدم النهائي.

### 3.5- آلية التواصل في النموذج المركزي

يقسم التواصل في النموذج المركزي إلى:

- تواصل ند- مخدم Peer- Server
- تواصل ند- لند P2P

#### 1.3.5- تواصل ند- مخدم Peer-Server

اعتمدنا في التصميم على وجود صف خاص (*Command Class*) يمثل الرسالة (برتوكول التخاطب) التي يتم تبادلها بين الزبون والمخدم الشكل (20)، حيث تحوي هذه الرسالة نوعها ومحتواها بالإضافة إلى طرقي الرسالة (مرسل، مستقبل). تختلف هذه الرسالة عن الرسالة النصية التي يرسلها المستخدم، حيث تكون الرسالة النصية غرضاً من هذا الصف له نوع "رسالة نصية". سنأتي على ذكر أنواع الرسالة المتبادلة لاحقاً.

تتكون كل رسالة من الند إلى مخدم HiastShare من "النمط Type"، وطول عنوان الهدف "Target Length"، وعنوان الهدف "Target Address"، وطول الحمل "Payload Length"، والحمل "Payload".

- يصف الحقل "Type" نوع أو نمط الرسالة.
- يصف الحقل "Target Length" طول عنوان الهدف.
- يكون الحقل "Target Address" هو عنوان الهدف مرمزاً بترميز ASCII.
- يصف الحقل "Payload Length" طول الحمل.
- يكون الحقل "Payload" الحمل في الرسالة مرمزاً بترميز Unicode<sup>5</sup>.

<sup>5</sup> الترميز Unicode هو نفسه الترميز UTF-16 وقد تم استخدام هذا الترميز لإتاحة الدردشة باللغة العربية.

<Type> (2 Byte)	<Target Length> (4 Byte)	<Target Address> (n Byte)	<Payload Length> (4 Byte)	<Payload> (n Byte)
--------------------	-----------------------------	------------------------------	------------------------------	-----------------------

الشكل 18 بنية رسالة بروتوكول HiastShare من الند إلى المخدم

تتكون كل رسالة من مخدم HiastShare إلى الند من "النمط Type"، وطول عنوان المرسل "Sender Length"، وعنوان المرسل "Sender Address"، وطول اسم المرسل "Sender Name Length"، واسم المرسل "Sender Name"، وطول عنوان الهدف "Target Length"، وعنوان الهدف "Target Address"، وطول الحمل "Payload Length"، والحمل "Payload".

- يصف الحقل "Type" نوع أو نمط الرسالة.
- يصف الحقل "Sender Length" طول عنوان المرسل.
- يكون الحقل "Sender Address" عنوان المرسل مرمّز بترميز ASCII.
- يصف الحقل "Sender Name Length" طول اسم المرسل.
- يكون الحقل "Sender Name" اسم المرسل مرمّز بترميز ASCII.
- يصف الحقل "Target Length" طول عنوان الهدف.
- يكون الحقل "Target Address" هو عنوان الهدف مرمّز بترميز ASCII.
- يصف الحقل "Payload Length" طول الحمل.
- يكون الحقل "Payload" الحمل في الرسالة مرمّز بترميز Unicode.

<Type> (2 Byte)	<Sender Length> (4 Byte)	<Sender Address> (n Byte)	<Sender Name Length> (4 Byte)	<Sender Name> (n Byte)	<Target Length> (4 Byte)	<Target Address> (n Byte)	<Payload Length> (4 Byte)	<Payload> (n Byte)
--------------------	-----------------------------	------------------------------	----------------------------------	---------------------------	-----------------------------	------------------------------	------------------------------	-----------------------

الشكل 19 بنية بروتوكول HiastShare من المخدم إلى الند

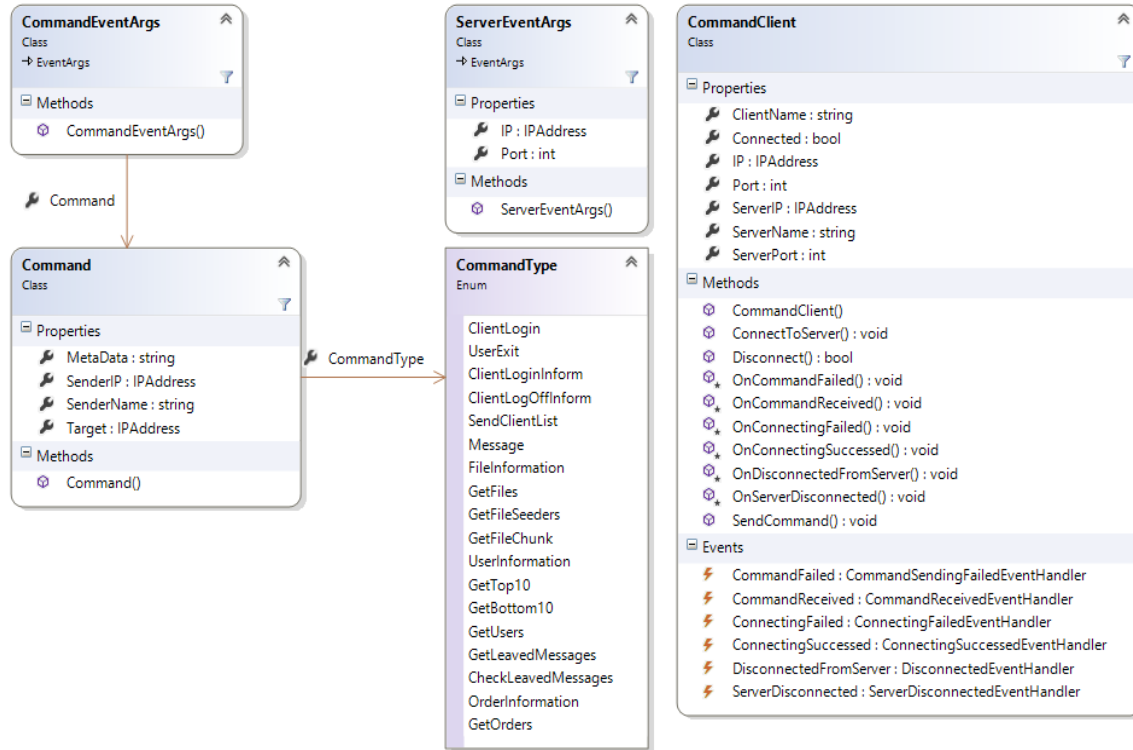
بعد اعتماد بروتوكول التخاطب السابق يوجد لدينا صفّين أساسيين الأول من جهة الزبون واسمه (*CommandClient*) والثاني من جهة المخدم واسمه (*ClientManager Class*).

<sup>6</sup>*CommandClient class*: هو الصف الأساسي ليتمكن الزبون من إنشاء الاتصال مع المخدم من أهم المعاملات التي يأخذها في بانيه Constructor عنوان ومنفذ المخدم (IPEndpoint). يمكن من خلاله الاتصال بالمخدم<sup>7</sup> أو قطع الاتصال معه أو إرسال رسالة Command وفق البروتوكول المذكور سابقاً الشكل (18) مع الأخذ بعين الاعتبار مختلف

<sup>6</sup> ألحق مع التقرير توثيق وشرح كامل لجميع صفوف وتوابع نواة اتصال الزبون ونواة اتصال المخدم.

<sup>7</sup> بعد الاتصال بالمخدم يقوم الغرض باستقبال الرسائل مباشرةً وفق البروتوكول الشكل (19).

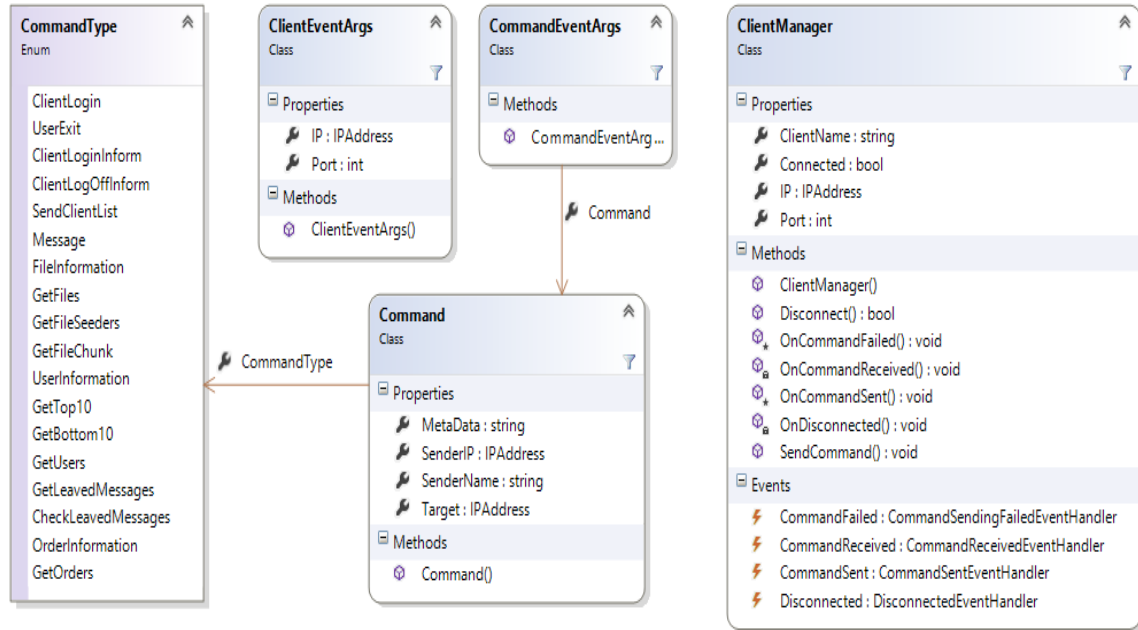
الأحداث التي يمكن الاستفادة منها لاحقاً، من هذه الأحداث<sup>8</sup> *CommandReceived Event* الذي يأخذ الغرض *CommandEventArgs* في معاملاته وهذا الغرض يفيد بتحديد نمط الرسالة المستقبلية، وبالتالي نستطيع معالجة كل رسالة مستقبلية من نمط معين معالجة خاصة بنمطها. هذا وقر الكثير من التوسعية والمرونة في تحقق متطلبات النظام حيث أصبحت بنية الاتصال منفصلة تقريباً عن التطبيق الذي سيعتمد عليها ونقل تقريباً لأن بنية البروتوكول المصمم آخذة بعين الاعتبار متطلبات التطبيق. الشكل (20) يوضح مخطط الصفوف لنواة اتصال الزبون.



الشكل 20 مخطط الصفوف لنواة اتصال الزبون

*Client Manager Class*: هو الصف الأساسي ليتمكن المخدم من التعامل مع الزبون حيث يأخذ في بانيه مقبس الزبون ليبدأ استقبال الرسائل من هذا المقبس وفق البروتوكول السابق الشكل (18) كما يمكن من خلاله إرسال رسالة إلى الزبون الشكل (19)، أيضاً أخذ بعين الاعتبار إنشاء مختلف الأحداث التي يمكن الاستفادة منها لاحقاً، من هذه الأحداث *Disconnected Event* الذي يأخذ غرض من *ClientEventArgs* في معاملاته وهذا الغرض يفيد بتحديد مقبس الزبون، وبالتالي يستطيع المخدم تتبع حالة الزبائن فيكون لديه دائماً لائحة بالزبائن النشطين خلال زمن التشغيل *Run-time*. الشكل (21) يوضح مخطط الصفوف لنواة اتصال المخدم.

<sup>8</sup> جميع الأحداث مشروحة في التوثيق المرفق مع التقرير.



الشكل 21 مخطط الصفوف لنواة اتصال المخدم

يتفق الزبون والمخدم على لائحة من أنماط الرسائل ليستخدامها التطبيق المبني على نواتي الاتصال، تكون هذه اللائحة من نمط تعداد Enum.

يستخدم المكونان HiastShare Central Server و HiastShare المكونان HiastShare السابق حيث يمثل المكون HiastShare التطبيق الذي يتعامل مع المستخدم النهائي و المكون HiastShare Central Server تطبيق المخدم المركزي. سنشرح فيما يلي جدول (5) الرسائل التي يتبادلها التطبيقان ووظيفة كل منها باختصار (تكون معاملة الرسالة عند المخدم مختلفة عنها عند الزبون مثلاً عندما يرسل الزبون رسالة من نمط ClientLogin يرسل معها اسم المستخدم وكلمة المرور في الحمل انظر الشكل (18)، المخدم يفهم أنه عندما يستقبل هذا النمط من الرسائل عليه أن يستخلص اسم المستخدم وكلمة المرور من الحمل ويتحقق من صحتها ثم يعيد الرد إلى الزبون)

وظيفة الرسالة	مخطط الرسالة
طلب تسجيل دخول. تحوي اسم المستخدم وكلمة المرور كحمل	<b>ClientLogin</b>
تُرسل هذه الرسالة إلى جميع المستخدمين <sup>9</sup> عندما يسجل أحد المستخدمين دخوله إلى المخدم لإعلامهم أن هذا المستخدم أصبح نشطاً. تحوي عنوان الزبون ClientIP واسمه كحمل	<b>ClientLoginInform</b>
تُرسل هذه الرسالة لجميع المستخدمين عندما يسجل أحد المستخدمين خروجه من المخدم لإعلامهم أن هذا المستخدم أصبح غير نشط	<b>ClientLogOffInform</b>
تُرسل هذه الرسالة إلى المستخدمين المتصلين حديثاً لتخبرهم عن جميع المستخدمين المتصلين حالياً في النظام وتكون هذه الرسالة رداً على طلب من المستخدم نفسه يرسله بعد عملية تسجيل الدخول مباشرة	<b>SendClientList</b>
تحتوي هذه الرسالة جسم الرسالة كحمل وعنوان المرسل إليه كعنوان للهدف يستقبلها المخدم المركزي ثم يعيد إرسالها إلى الهدف	<b>Message</b>
تحتوي هذه الرسالة معلومات الملف المشارك كحمل وتُرسل إلى المخدم المركزي ليسجل هذه المعلومات في قاعدة المعطيات ويربط هذه المعلومات مع المستخدم الذي شارك هذه الملف	<b>FileInformation</b>
تُرسل هذه الرسالة إلى المخدم لطلب لائحة بالملفات المشاركة على النظام	<b>GetFiles</b>
تُرسل هذه الرسالة إلى المخدم لطلب بذور ملف، حيث تحوي معرف الملف كحمل	<b>GetFileSeeders</b>
تُرسل هذه الرسالة إلى المخدم لمعرفة مجموع علامات مستخدم، تحوي اسم المستخدم كحمل	<b>UserInformation</b>
للسؤال عن أعلى 10 مستخدمين في النظام من ناحية مجموع العلامات	<b>GetTop10</b>

<sup>9</sup> جميع المستخدمين هنا تعني جميع المستخدمين النشطين حالياً فالمخدم يملك دائماً لائحة بهم تتحدث خلال زمن التشغيل كما ذكرنا سابقاً.

للسؤال عن أدنى 10 مستخدمين في النظام من ناحية مجموع العلامات	<b>GetBottom10</b>
للسؤال عن أسماء جميع المستخدمين المسجلين بالنظام	<b>GetUsers</b>
لسؤال المخدم عن الرسائل التي تركت له حينما كانت حالته غير نشط	<b>GetLeavedMessages</b>
لسؤال المخدم عن وجود رسائل متروكة جديدة، حيث يُرسل معرّف آخر رسالة استلمها المخدم كحمل	<b>CheckLeavedMessages</b>
تحتوي هذه الرسالة تفاصيل الطلب العام كحمل وتُرسل إلى المخدم المركزي ليسجل هذه المعلومات في قاعدة المعطيات ويربط هذه المعلومات مع المستخدم الذي نشر الطلب	<b>OrderInformation</b>
تُرسل هذه الرسالة للمخدم لطلب لائحة بالطلبات العامة المنشورة على النظام	<b>GetOrders</b>

جدول 5 أنماط الرسائل المتبادلة بين الند والمخدم في نظام P2P HiastShare

### 2.3.5- تواصل ند- لند P2P

اعتمدنا في تواصل الند للند على نفس نواقي الاتصال المستخدمتان في تواصل ند- مخدم لكون هاتان النواتان مجردتان عن التطبيق الذي سيعتمد عليهما طالما أنّ بروتوكول التخاطب يلبي حاجات التطبيق. احتجنا لرسالة واحدة فقط جدول (6) يرسلها الند للند الآخر

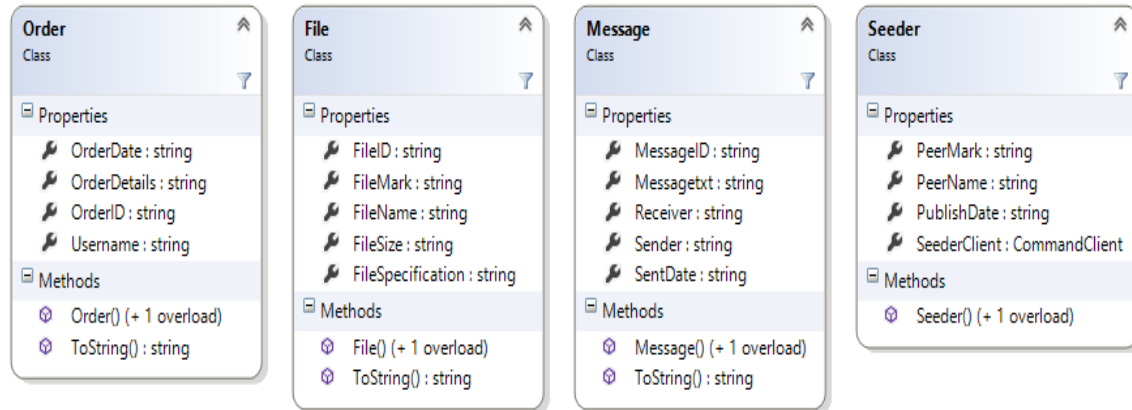
وظيفة الرسالة	نمط الرسالة
يطلب الند في هذه الرسالة قطعة من الملف <sup>10</sup> ، حيث تحوي اسم الملف المراد تحميله والإزاحة (رقم البايت المراد بدء التحميل من عنده) وطول القطعة (عدد البايتات المراد تحميلها)	<b>GetFileChunk</b>

جدول 6 أنماط الرسائل المتبادلة بين الند والند الآخر في نظام P2P HiastShare

<sup>10</sup> قد تكون هذه القطعة كامل الملف اذا كانت الإزاحة 0 وطول القطعة هو طول كامل الملف.

### 3.3.5- التواصل بين تطبيق الزبون HiastShare وتطبيق المخدم المركزي

يملك كل من التطبيقان (تطبيق الزبون HiastShare وتطبيق المخدم المركزي HiastShare Central Server) كيانات مشتركة Shared Entities الشكل (22) لتسهيل التعامل بين بعضهم البعض فمثلاً عندما يطلب المستخدم لائحة بالملفات المشاركة على النظام يقوم المخدم بالتعامل مع لائحة من الملفات ويرسلها إلى المستخدم الذي يستقبلها كلائحة من الملفات أيضاً.



الشكل 22 الكيانات المشتركة بين تطبيق الزبون وتطبيق المخدم المركزي

نلاحظ وجود زيادة تحميل على تابع ToString() في أغلب الكيانات، وذلك بهدف تحضير الغرض ليكون حمل في الرسالة command المتبادلة بين التطبيقين.

كما نلاحظ وجود الوصفة SeederCleint من واصفات الصف Seeder وهذه الوصفة من نوع CommandClient وهو الصف الأساسي المسؤول عن عملية الاتصال بالمخدم (هنا المخدم الموجود في البذرة) كما ذكرنا سابقاً، أي أصبح الاتصال بالبذرة وتبادل الرسائل معها عن طريق هذه الوصفة.



## 4.5- مخطط قاعدة معطيات المخدّم المركزي

يحتوي المخدّم المركزي قاعدة معطيات لتخزين معلومات عن مختلف الكيانات التي تتفاعل في النظام الشكل (23).

الكيان **File**: هو الكيان الذي يمثّل الملف المشارك، فيه كافة الواصفات المحددة للملف وهي:

- **FileID**: معرّف الملف.
- **FileName**: اسم الملف.
- **FileSize**: حجم الملف.
- **FileSpecification**: لمحة مختصرة عن الملف.
- **FileMark**: العلامات المخصصة للملف.

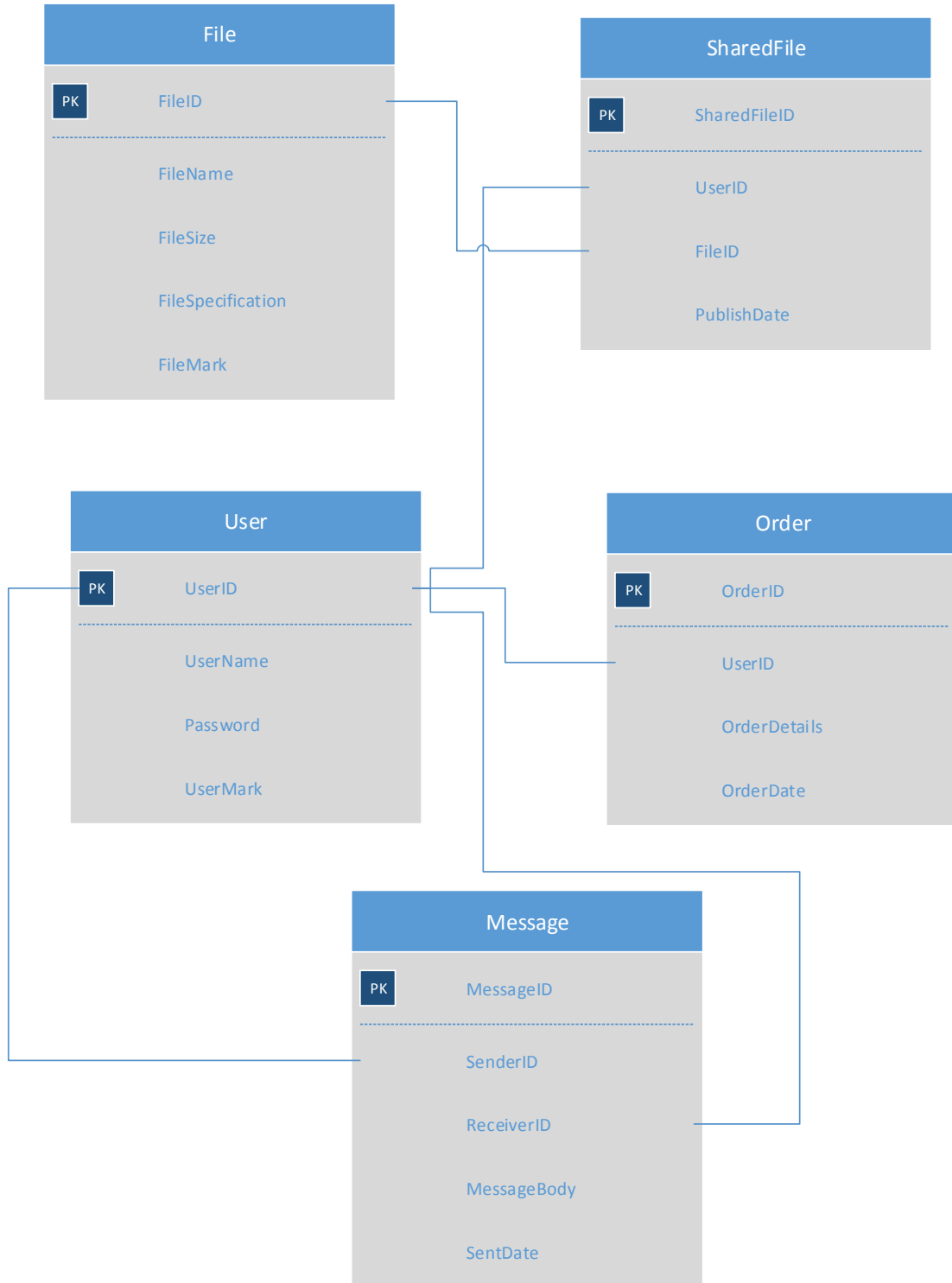
الكيان **User**: هو الكيان الذي يمثّل المستخدم المسجّل في النظام يضيفه مدير النظام من تطبيق آخر سنتكلم عنه لاحقاً فيه كافة الواصفات المحددة للمستخدم وهي:

- **UserID**: معرّف المستخدم.
- **UserName**: اسم المستخدم.
- **Password**: كلمة المرور الخاصة بالمستخدم.
- **UserMark**: مجموع علامات المستخدم.

يرتبط الملف **File** بعلاقة **many\_to\_many** مع المستخدم **User** حيث يمكن للمستخدم مشاركة أكثر من ملف ويمكن للملف أن يُشارك من قبل أكثر من مستخدم، ولتمثيل هذه العلاقة في قاعدة المعطيات نحتاج إلى كيان الملفات المشاركة **SharedFile** الذي يربط كل ملف بالمستخدم المشارك له، يحوي المفتاح الأساسي لكلا الكيانين كمفاتيح خارجية بالإضافة إلى واصفة مشتركة تخص كلا الكيانين **SharedDate**.

الكيان **Message**: هو الكيان الذي يمثّل الرسالة التي يتركها مستخدم لمستخدم آخر في حال كان هذا المستخدم غير نشط يحوي على معرّف المرسل والمستقبل كمفاتيح خارجية، (مرتبط مع الكيان مستخدم بعلاقة **One\_to\_Many**).

الكيان **Order**: يمثّل هذا الكيان الطلب العام الذي يتركه المستخدم من واصفاته تفاصيل الطلب **OrderDetails** وتاريخ نشر الطلب **OrderDate** وأيضاً يحوي على معرّف المستخدم الذي نشر الطلب، (مرتبط مع الكيان مستخدم بعلاقة **One\_to\_Many**).



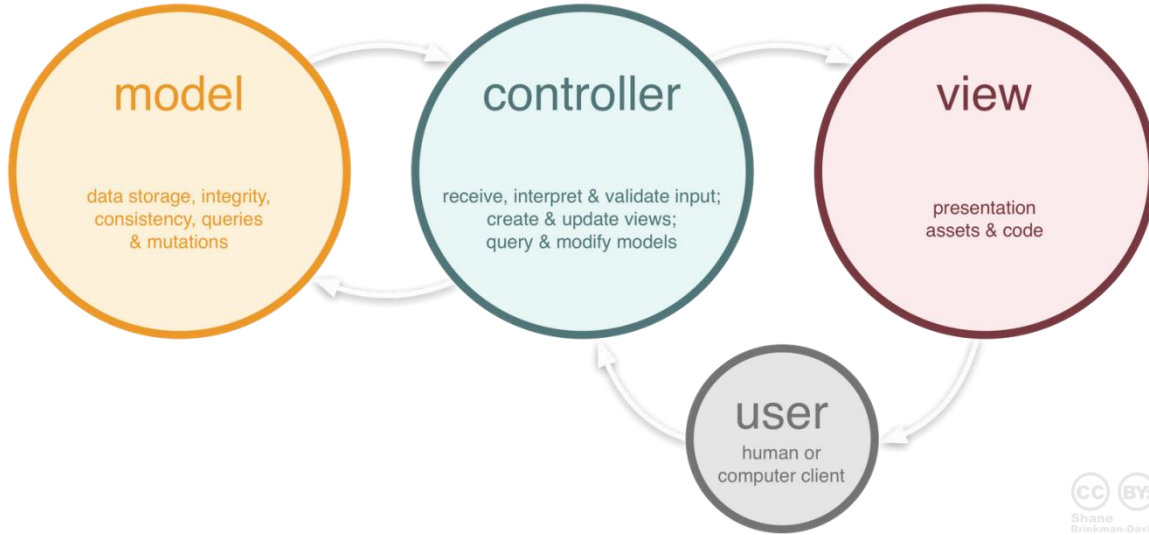
الشكل 23 مخطط قاعدة معطيات المخدم المركزي

## 5.5- تصميم نظام إدارة المخدم المركزي

تم إنشاء هذا التطبيق كتطبيق وب يتيح لمدير النظام بإدارة ومراقبة جميع التناقلات التي تجري خلال النظام .

إن بيئة العمل المستخدمة ASP MVC مصممة وفق مقاربة غرضية التوجه **Object Oriented Design**، وهي بالأخص تعتمد نمط تصميم **MVC (Model, View, Controller)** وتعتمد إلى هذا النمط بالأساس لفصل التطبيق إلى ثلاث طبقات وبالتالي تفصل التمثيل الداخلي للمعلومات ومعالجتها عن طريقة عرضها، حيث توفر في بنية مجلداتها مجلد خاص لكل طبقة من الطبقات السابقة لوضع الجزء البرمجي الخاص به داخل هذا المجلد. أما عن الطبقات فهي:

- **طبقة النموذج Model:** تتعامل هذه الطبقة بشكل رئيسي مع المعطيات المخزنة مثل قاعدة المعطيات أو ملفات تخزينية أخرى وجلب المعطيات منها.
- **طبقة المتحكم Controller:** يستلم المتحكم الطلبات الداخلة إلى التطبيق من المستخدم النهائي، يعالج المتحكم الطلبات حيث يستلم معطيات الدخل ويمررها الى طبقة Model التي تعيد بدورها معطيات الخرج ثم يقوم المتحكم أيضاً بمعالجتها مرة أخرى قبل أن يمررها الى طبقة View لإظهارها.
- **طبقة الإظهار View:** تتحكم هذه الطبقة بالإظهار وتديره ليظهر بشكله النهائي للمستخدم النهائي.



الشكل 24 بنية MVC وآلية عملها

وجب التنويه إلى أنّ طبقة النموذج أو Model تشترك في هذا التطبيق مع تطبيق المخدم المركزي إذ أنّ كلا التطبيقين يتعامل مع قاعدة معطيات واحدة الشكل (23) ولكن تختلف طريقة المعالجة والإظهار.

## 6.5- تأثير التصميم المتبع على توسيع النظام

سندرس في هذه الفقرة تأثير القرارات التصميمية (تقسيم الطبقات) على سهولة توسيع النظام scalability.

أولاً بالنسبة إلى نظام HiastShare فقد بينا سابقاً أنه مبني على الصف Command الذي جرد وظيفة الرسالة فلاضافة متطلب جديد أو حالة استخدام جديدة يكفي تعريف نمط الرسالة عند كل من الزبون والمخدّم ومعالجة هذه الرسالة حسب الحاجة في كلا الطرفين مع الأخذ بعين الاعتبار بنية البروتوكول المصمم.

ثانياً بالنسبة إلى نظام إدارة المخدّم المركزي إن المقاربة التي تم استخدامها في تصميم النظام MVC جعلت من السهل تطوير أو صيانة أي جزء من النظام، وكذلك الأمر بالنسبة لتوسعة النظام، ذلك بسبب ما توفره من إمكانية إعادة استخدام أي مجتزأ آخر.

ومن جهة أخرى أعطى استخدام تصميم الطبقات السابقة المنفصلة عن بعضها بنيانياً، النظام مرونة في تطوير أي طبقة بمعزل عن الطبقات الأخرى، كما أصبح إضافة مجتزأ جديد إلى النظام أمراً جلياً، فما على المطور سوى إضافة المجترآت الفرعية إلى كل طبقة من الطبقات السابقة ليتحقق التكامل بين المجتزأ الجديد والنظام.

## الفصل السادس

### التنفيذ والاختبارات

## Implementation & Testing

نقدم في هذا الفصل شرحاً عن بيئة العمل المستخدمة في تنجيز النظام نبين فيه الأدوات المستعملة. كما نبين بعض الاختبارات التي تم اعتمادها في التحقق من صحة التطبيق. نورد أخيراً الخلاصة وبعض الآفاق المستقبلية.

## 1.6- بيئة العمل وأدوات التنفيذ لنظام HiastShare

### مقدمة:

تعدّ برمجة المقابس socket programing نواة البرمجة الشبكية في نظامي Windows و Linux، وفي وقتنا الراهن تعدّ منصة .Net من أقوى المنصات التي نفذت هذه النواة.

اعتمدنا في تنفيذنا لنواة الاتصال في نظام HiastShare بشكل أساسي على برمجة المقابس في منصة .Net. بلغة C#<sup>11</sup>. للدقة أكثر نُفذت نواة الاتصال عند الزبون CommandClient ونواة الاتصال عند المخدم CommandServer لإتاحة الاتصال بين المخدم والزبائن (الذين يمكن أن يصل عددهم إلى أكثر من 200 زبون) عن طريق إرسال رسائل من نمط معين (معرفة في تعداد CommandType Enum كما ذكرنا سابقاً).

بُني تطبيق HiastShare من ناحية الزبون من نوع Windows Forms Application وبُني التطبيق HiastShare Central Server من ناحية المخدم من نوع Console Application.

كلا التطبيقين يستخدمان نواتا الاتصال لتحقيق وظائف نقل الملفات والمحادثة ووظائف أخرى مذكورة في المتطلبات. كما اخترنا SQL server 2012 كنظام إدارة قواعد المعطيات المربوطة مع المخدم المركزي كما يوضح الشكل (16) في فقرة التصميم الرئيسي من الفصل السابق.

### 1.1.6- تنفيذ نواة اتصال المخدم CommandServer

يوجد صف يسمى ClientManager كما ذكرنا سابقاً في التصميم، فعندما ينشأ الاتصال بين المخدم والزبون يمرر مقبس الاتصال إلى بائي هذا الصف وينشأ غرض جديد لدى المخدم من نوع ClientManager يضاف إلى لائحة الزبائن المتصلين حالياً. يكون كل غرض من هذه اللائحة مسؤولاً عن التواصل مع الزبون الخاص به. ينشر Publish الصف ClientManager العديد من الأحداث Events، يسجل Subscribe بما تطبيق المخدم أهم هذه الأحداث<sup>12</sup>:

```
public event CommandReceivedEventHandler CommandReceived;
```

يظهر هذا الحدث عندما تصل رسالة command من الزبون.

```
public event DisconnectedEventHandler Disconnected;
```

<sup>11</sup> تم الاعتماد بشكل رئيسي على مكتبي "System.Net.Sockets" التي تحتوي على الصفوف المغلفة لواجهات Winsock منخفضة المستوى، و مكتبة

"System.Threading" للتعامل مع التباين Threads والتومعة Semaphore.

<sup>12</sup> جميع الأحداث مشروحة ومبيّنة بالتوثيق المرفق مع التقرير.

يظهر هذا الحدث عندما يقطع أحد الزبائن اتصاله بهذا المخدم.

## 2.1.6- تنفيذ نواة اتصال الزبون CommandClient

تشبه هذه النواة نواة اتصال المخدم من حيث البنية؛ الصف الأساسي والمسؤول عن كل شيء في هذه النواة هو الصف CommandClient. أيضاً ينشر Publish الصف CommandClient العديد من الأحداث Events، يسجل Subscribe بها تطبيق الزبون. من هذه الأحداث:

```
public event CommandReceivedEventHandler CommandReceived;
```

يظهر هذا الحدث عندما تصل رسالة command من المخدم.

```
public event ServerDisconnectedEventHandler ServerDisconnected;
```

يظهر هذا الحدث عندما يقع المخدم اتصاله بهذا الزبون.

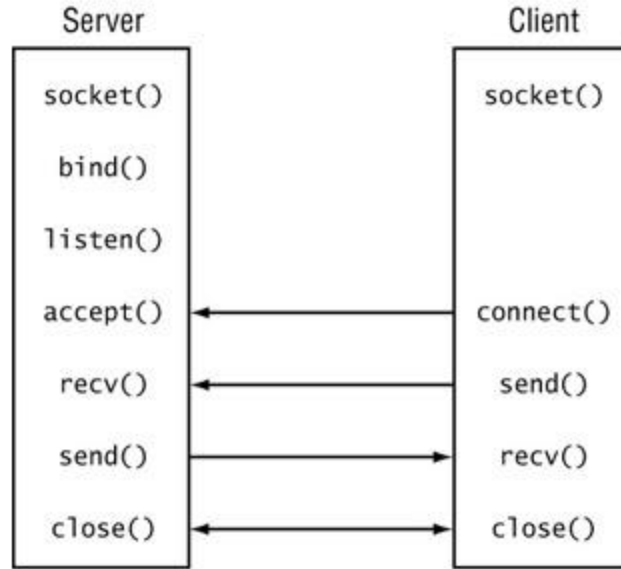
## 3.1.6- بروتوكول النقل المستخدم

يوجد نوعان من الاتصالات: الاتصال ارتباطي التوجه والاتصال عديم الارتباط. في المقبس ارتباطي التوجه يُستخدم بروتوكول TCP لإنشاء جلسة الاتصال connection بين نقطتين طرفيتين two IP Address endpoint. يوجد حمل مضاف overhead في عملية إنشاء الاتصال لكن عندما ينشأ تُنقل المعطيات Data بشكل موثوق بين الأجهزة المتصلة.

تُستخدم المقابس عديمة الارتباط البروتوكول UDP. لا يحوي هذا النوع من البروتوكولات على حمل زائد لإنشاء الاتصال فلا يوجد أي معلومات إضافية لبدء عملية التراسل بين الأجهزة وهذا ما يصعب عملية تحديد الجهاز الذي يعمل كزبون والجهاز الذي يعمل كمخدم. اعتمدنا في عملية الاتصال في نظامنا على النوع الأول Connection-oriented. وبالتالي تم إنشاء مقابس ارتباطية التوجه التي سنتكلم عنها في الفقرة التالية.

## 4.1.6- استخدام المقابس ارتباطية التوجه Connection-oriented Sockets

استُخدمت برمجة المقابس socket programming لإنشاء الاتصالات بين الأنداد peers وأيضاً بين الند والمخدم المركزي. حيث يجري أولاً تهيئة إجرائية جديدة للتصنت على منفذ معين، ثم يجري استدعاء طريقة من الصف المسؤول عن التصنت listen class لمراقبة طلبات الاتصال القادمة. وفي حال الموافقة على فتح الاتصال يتم إنشاء الاتصال وتعيد الطريقة غرض من الصف مقبس socket class لهذا الاتصال. بعدها يجري التواصل على هذا المقبس المنشأ كما هو موضح في الشكل (25).



الشكل 25 آلية إنشاء مقبس بين مخدم وزبون

### 5.1.6- مشكلة التوقف Blocking في التطبيقات الشبكية

تكون المقابس افتراضياً في وضع التوقف Blocking mode [10] في هذا الوضع سيوقف التنفيذ طالما لم يتم إنهاء العمل. وبالتالي ستتوقف وظائف البرنامج الأخرى ضمن البرنامج منتظرة انتهاء عمل المقبس. مثلاً ستجمد الواجهة في حالة التطبيقات التي تعتمد على الواجهات الرسومية.

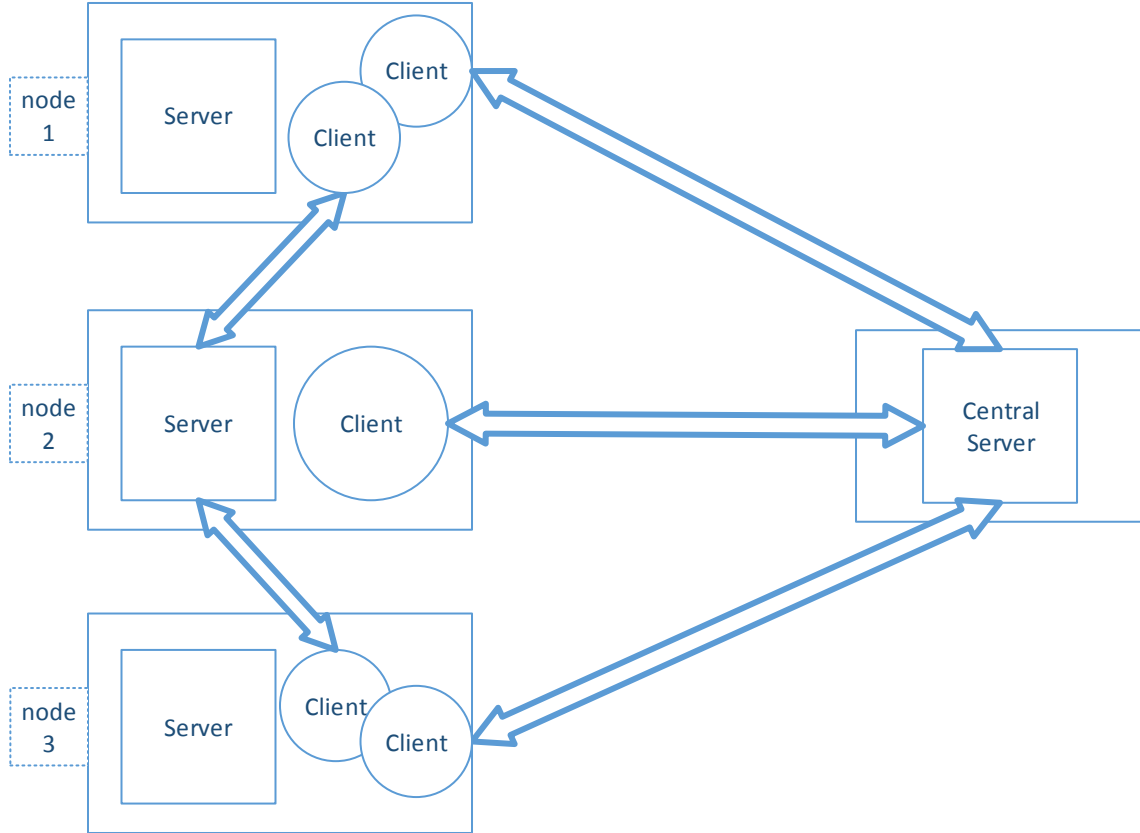
يوجد أكثر من طريقة لحل هذه المشكلة أشهرها استخدام تعددية النياسب Multi-threading أو استخدام البرمجة الشبكية غير المتزامنة asynchronous Network Programing أو طريقة "Non-Blocking Socket". من ناحية السهولة اخترنا طريقة تعددية النياسب للتغلب هذه المشكلة وتطرقنا أيضاً إلى استخدام البرمجة غير المتزامنة لكن بشكل أقل، واستخدمنا BackgroundWorker لتنفيذ تعددية النياسب في الوظائف التي تستغرق زمناً طويلاً في الزبون والمخدم.

### 6.1.6- إدارة النياسب

صُمم النظام بحيث يقبل المخدم (سواء المخدم المركزي أو المخدم في كل عقدة) التعامل مع أكثر من اتصال كما هو موضح في الشكل (26) أي التعامل مع أكثر من زبون، بهذه الطريقة نضمن توسع تطبيقات النظام وزيادة كفاءته. حيث يُنشأ نيسب thread عند كل طلب اتصال connection request، ويكون هذا النيسب مسؤولاً عن تخدم الطلبات. يُستقبل الطلب المرسل من عقدة المرسل ثم يُتابع عن طريق نيسب خاص. إذاً سيكون المخدم قادر على تخدم عدد جيد من الأنداد peers في نفس الوقت، حيث تعتبر إدارة الأنداد من المميزات الهامة لهذا النظام وهو عامل مهم لتحديد فعالية النظام ككل. يوجد لكل غرض من نوع ClientManager (وهو الصف الأساسي المسؤول عن التعامل مع الزبون في



المخدم)، ولكل غرض من نوع **CommandClient** (وهو الصف الأساسي المسؤول عن التعامل مع المخدم في الزبون) نيسب خاص بالإرسال ونيسب خاص بالاستقبال مع مراعاة وجود المقاطع الحرجة **Critical Section** واستخدام التومئة **Semaphore** لتنظيم دخول النياسب إلى هذه المقاطع.



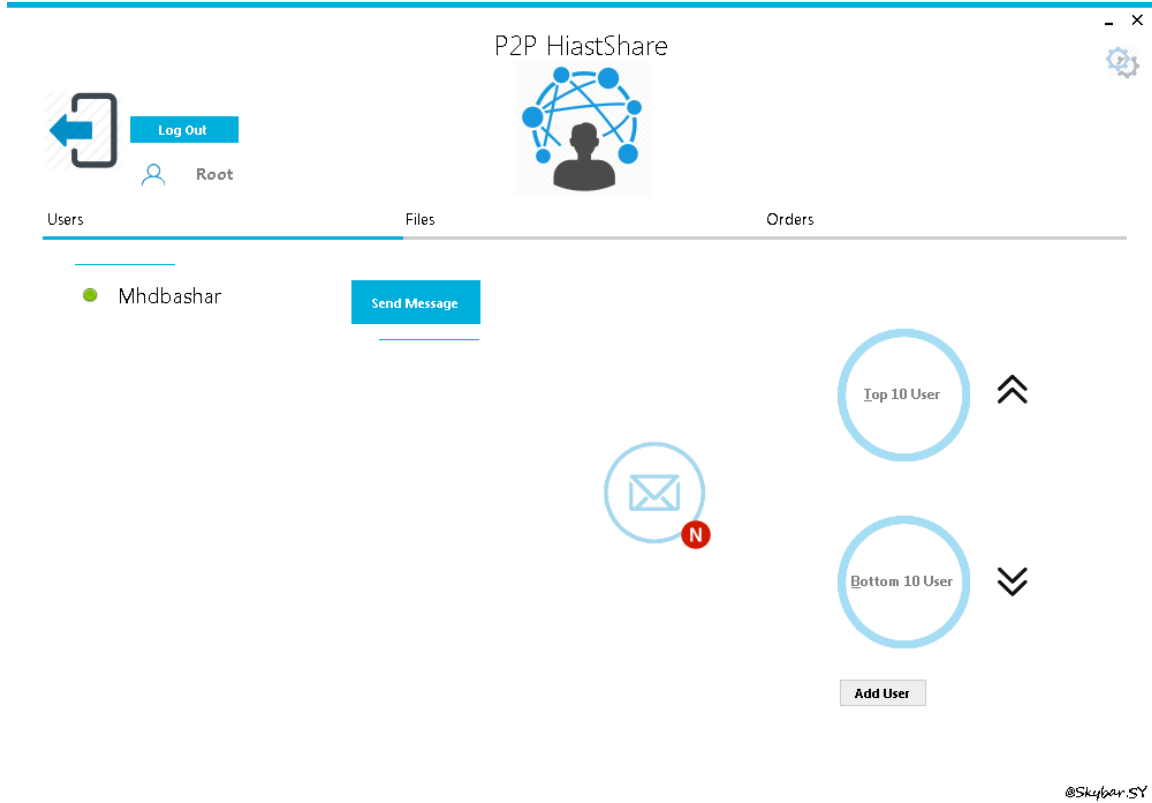
الشكل 26 بنية النظام من وجهة نظر زبون-مخدم

### 7.1.6- إرسال واستقبال البيانات

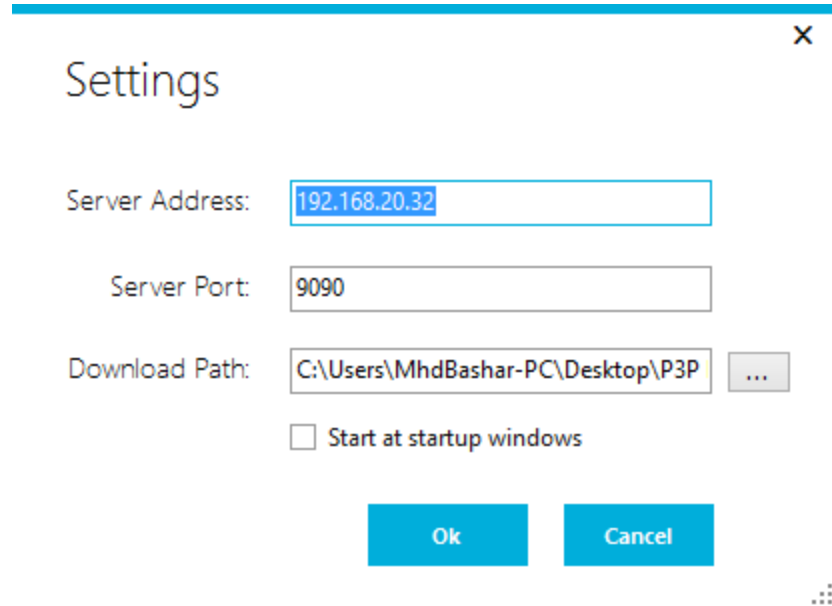
يعتمد النظام على الرسائل **command** المتبادلة بين المخدم والزبون، لذا سنحتاج إلى غرض من هذه الرسالة لإرسال واستقبال البيانات. لهذا الغرض تم بناء الصف **Commend Class** كما ذكرنا في الدراسة التصميمية وحسب الشكليات (18, 19) وهو يمثل بروتوكول التخاطب، يوجد هذا الصف عند كل من المخدم والزبون. فعندما يريد المخدم إرسال رسالة **command** إلى الزبون يبني غرض من هذا الصف ويرسله إلى الزبون والعكس بالعكس. يُحول هذا الغرض إلى مصفوفة من البايتات ثم تُرسل أو تُستقبل عبر الشبكة.

## 8.1.6- تنفيذ التطبيق HiastShare

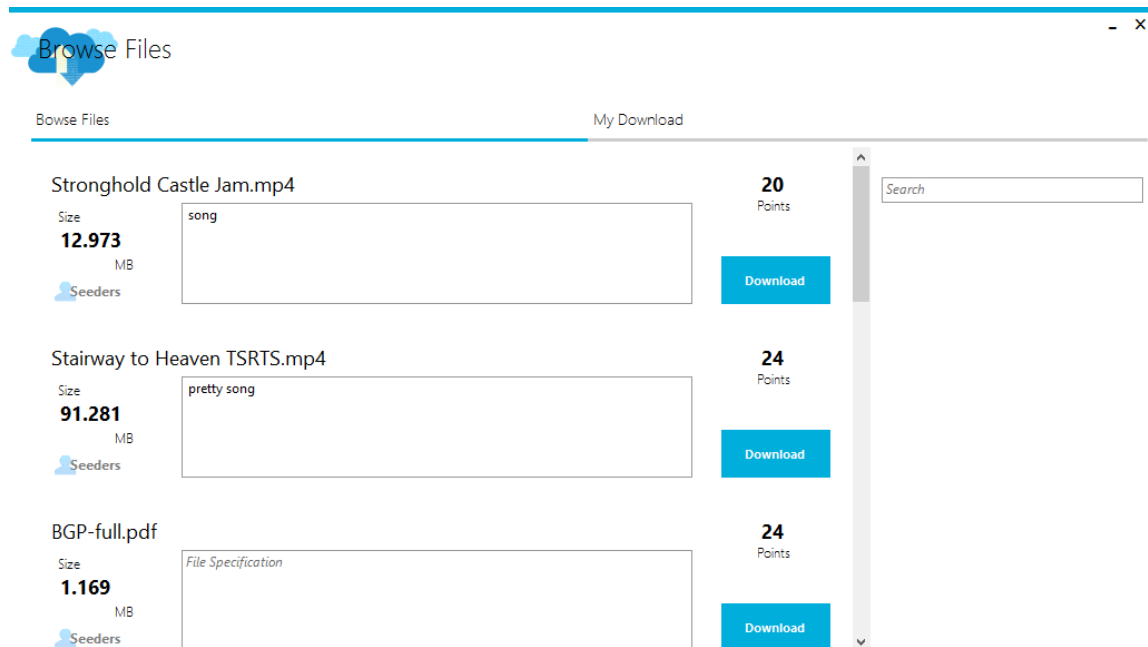
الواجهة الأساسية للتطبيق هي واجهة رسومية تؤمن العديد من الخيارات للمستخدم، الغرض الأساسي من الواجهة الرسومية هو جعل المستخدم يتفاعل مع التطبيق قدر الإمكان. حيثُ أخذ بعين الاعتبار كل الوسائل التي تريح المستخدم في تعامله مع التطبيق وتعريفه بكل الإمكانيات والوظائف التي يقدمها التطبيق. تمَّ استخدام الإطار Metro Framework لبناء واجهات التطبيق. بعض واجهات التطبيق:



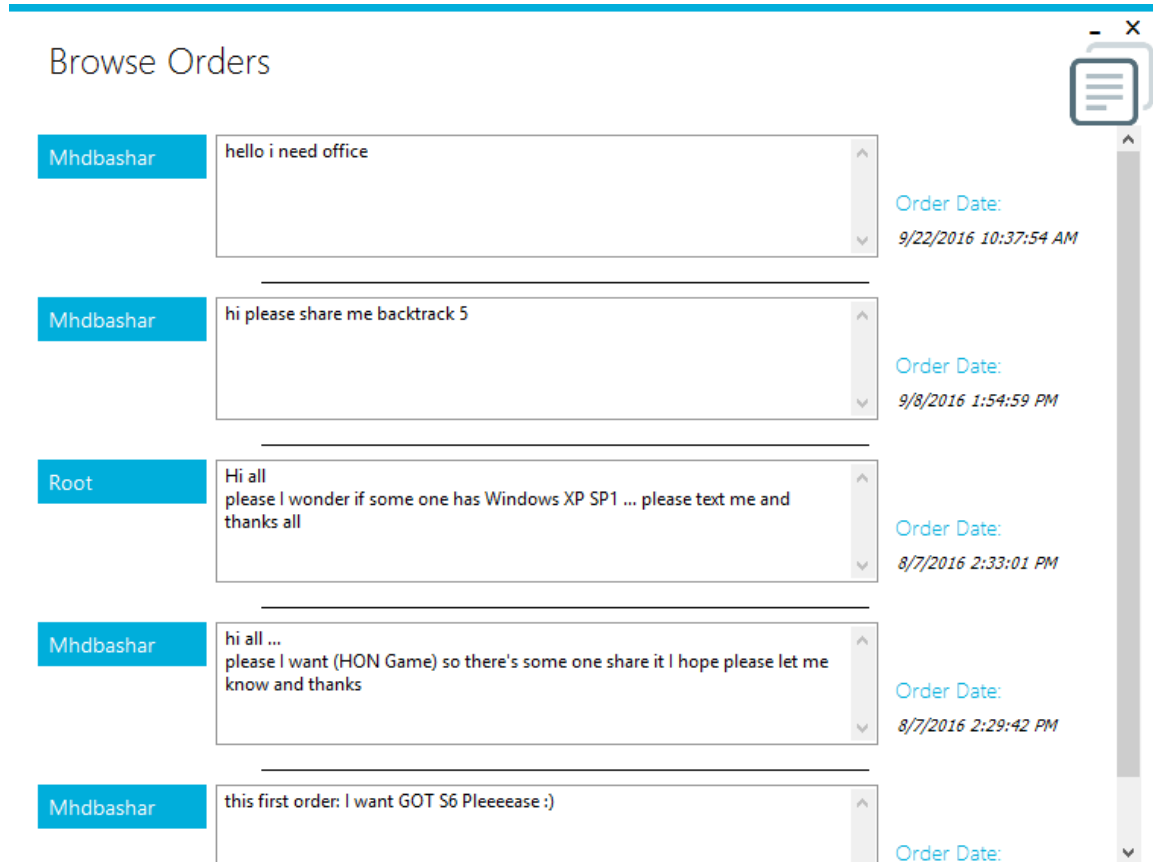
الشكل 27 واجهة التطبيق الرئيسية



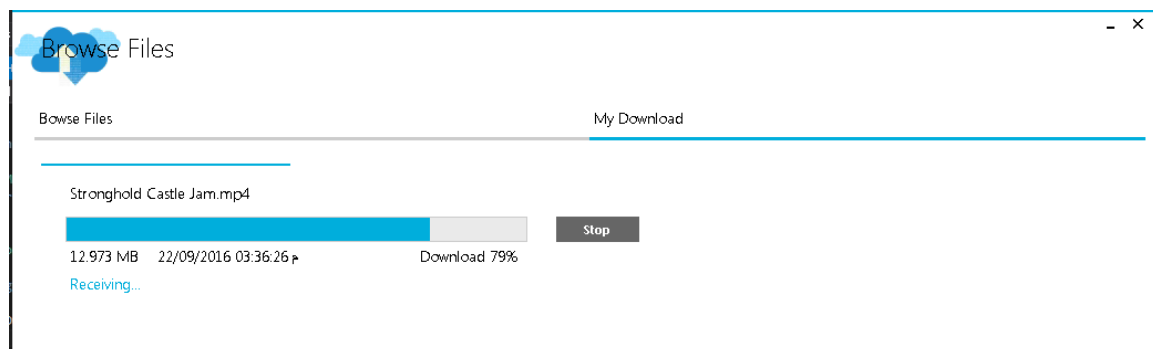
الشكل 28 واجهة إعدادات التطبيق



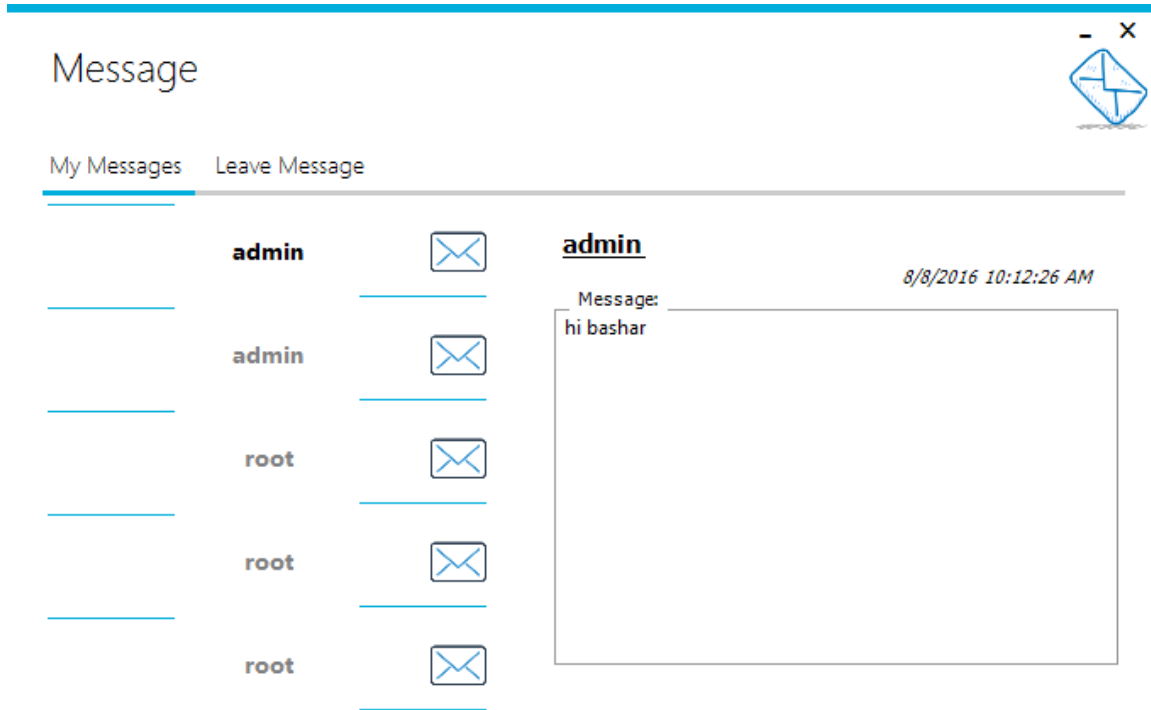
الشكل 29 واجهة استعراض الملفات المشاركة



الشكل 30 واجهة الطلبات العامة



الشكل 31 واجهة تحميل ملف



الشكل 32 واجهة عرض الرسائل المتروكة عندما كان المستخدم غير نشط

#### 1.8.1.6- إدارة بدء تشغيل التطبيق startup

سنسمح بإضافة التطبيق إلى قائمة البرامج التي تعمل عند بدء التشغيل، وذلك لنسهل عليهم عملية تسجيل الدخول اليومي والاستفادة من سياسة اكتساب النقاط عند كل تسجيل دخول يمضي عليه أكثر من 24 ساعة. فقد وضع خيار بدء تشغيل التطبيق مع بدء التشغيل في لوحة إعدادات التطبيق يستطيع المستخدم تفعيلها أو إلغاء تفعيلها. واستخدمنا لذلك الغرض التسجيلات registry التي يقدمها نظام التشغيل لإضافة إعدادات خاصة بالتطبيقات، فعلينا استخدام المفتاح `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run` الخاص بالبرامج التي تعمل في البدء، وإنشاء مدخل جديد مفتاحه هو اسم التطبيق أو اسم الشركة وقيمته هو المسار الخاص بالتطبيق، وحذف هذا الخيار سهل إذ يكفي حذف قيمة المدخل.

#### 2.8.1.6- إنشاء نسخة واحدة فقط للتطبيق single-instance

لقد أخذ بعين الاعتبار تشغيل نسخة واحدة من التطبيق، فإذا حاول المستخدم فتح نسخة أخرى من البرنامج تظهر رسالة بأن التطبيق يعمل بالفعل وبالتالي لا تعمل إلا نسخة واحدة من التطبيق. والهدف من ذلك تفادي الأخطاء من أجل عمل الخادم الخاص بالتطبيق فهو يستمع إلى منفذ محدد ولا يمكن تشغيل خادم آخر يستمع إلى المنفذ نفسه. لتنفيذ هذا الغرض تم إنشاء إقفال متبادل (mutex (Mutual Exclusion باستخدام الصف mutex، هذا القفل يسمح بتنظيم مشاركة

الموارد بين النياسب. فعندما يتم تشغيل التطبيق للمرة الأولى يتم إنشاء إقفال متبادل mutex، وعند محاولة فتح نسخة أخرى من التطبيق يتم التأكد من وجود إقفال متبادل. يتم إزالة الإقفال المتبادل عندما يتم إغلاق النسخة الأولى من التطبيق.

## 9.1.6- تنفيذ تطبيق المخدم المركزي

كما ذكرنا هو تطبيق ساكن Console Application له تطبيق وب للإدارة كما سنرى في الفقرة القادمة، الواجهة الرئيسية تعرض تقارير عن الأحداث التي تحصل خلال زمن التشغيل، كما تُكتب هذه الأحداث كسجلات Logs للمراقبة والإدارة.

```
file:///C:/Users/MhdBashar-PC/Dropbox/P3P 1/P3PServer/bin/Debug/P3PServer....
*** Server starts listening on port 0.0.0.0:9090. Press Enter to Shutdown.
Client 192.168.20.32:12177 has been Connected < 9/22/2016 13:38:06 PM >
Client 192.168.21.221:12172 has been Connected < 9/22/2016 13:38:11 PM >
Client 192.168.20.32:12177 has been Disconnected. < 9/22/2016 13:39:14 PM >
```

الشكل 33 واجهة المخدم المركزي

## 2.6- بيئة العمل وأدوات التنفيذ لنظام إدارة المخدم المركزي

### 1.2.6- بيئة العمل

تم استخدام ASP.Net MVC 4 كإطار عمل Framework، باستخدام لغة C#، يمتلك هذا النظام قاعدة معطيات مشتركة مع نظام HiastShare وبالتالي، كما ذكرنا سابقاً، تم اختيار SQL Server 2012 كنظام لإدارة قواعد المعطيات.

## Asp.Net MVC 4 Framework

هي إطار عمل مفتوحة المصدر تقريباً لبناء تطبيقات الويب التي تتمتع بقابلية التوسع Scalability بطريقة قياسية، باستخدام أنماط Patterns إضافة إلى ميزات ASP.NET، كما تم تنجيز نمط MVC من خلالها لكي تساعد المطور على التنجيز مباشرة دون الاهتمام بتفاصيل البنية التحتية Infrastructure لنمط MVC.

تتميز Asp.Net MVC 4 بكونها بنية خفيفة Lightweight، وقابلة للاختبار Testable بطريقة فعالة، ومنتجة بشكل سريع Productive.

يمكن السبب الرئيسي لاختيار هذه البيئة في كونها متكاملة Integrated مع SQL Server المستخدم في نظام HiastShare بالأصل والسبب الآخر لأغراض تعليمية كوني لم أتطرق للعمل في هذه البيئة خلال المرحلة الدراسية إضافة لما تتمتع به البنية من سرعة في الإنتاج Productivity.

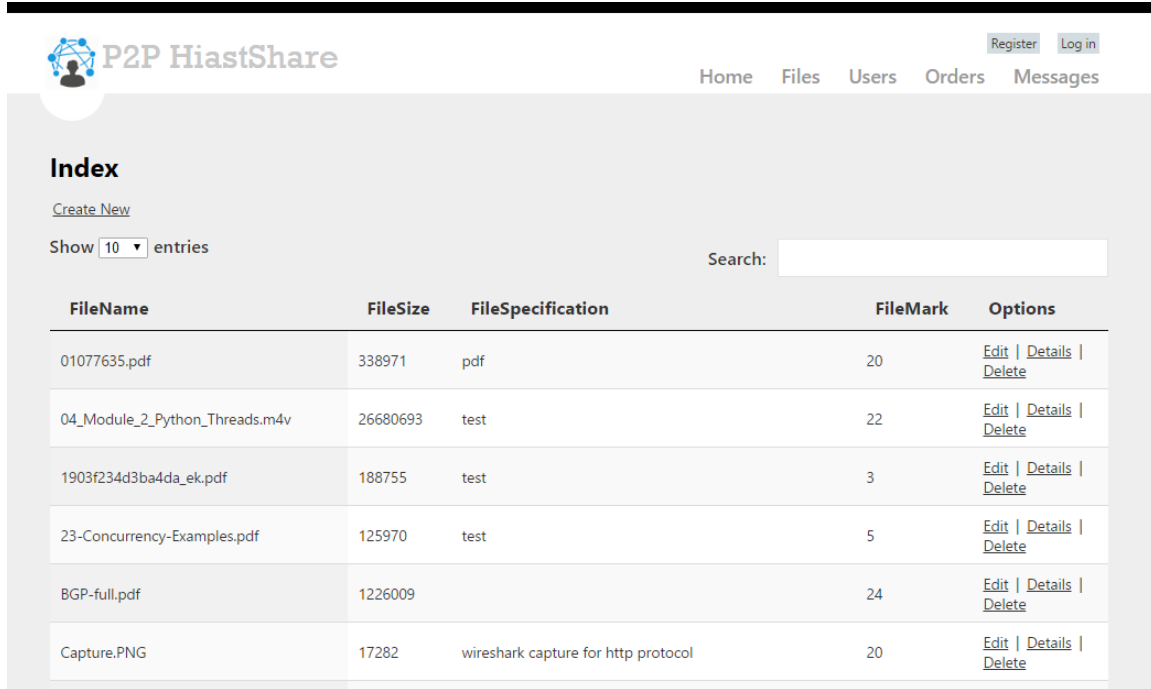
## 2.2.6- أدوات التطوير

### Entity Framework

هي أداة للربط بين الكائنات أو الأغراض وبين العلاقات تدعى بالمطابق الغرضي العلاقي ORM Object-Relation Mapper، أي تقوم هذه الأداة بأخذ بنية قاعدة المعطيات وتحولها إلى أغراض يمكن لبيئة .NET فهمها. يقوم المطورين باستخدام هذه الأغراض للاتصال بقاعدة المعطيات بدلاً من الاتصال المباشر بقاعدة المعطيات. يمكن القيام بكل العمليات التي تتم عادة على قاعدة المعطيات مباشرة باستخدام Entity Framework. هنالك ثلاث طرق تعمل بها Entity Framework:

- الرمز أولاً Code-First
- القالب أولاً Model-First
- قاعدة المعطيات أولاً Database-First

تم اعتماد طريقة قاعدة المعطيات أولاً كونها مبنية سابقاً في نظام HiastShare.



The screenshot shows the P2P HiastShare web interface. At the top, there is a navigation bar with links for Home, Files, Users, Orders, and Messages. Below the navigation bar, there is a section titled "Index" with a "Create New" link and a "Show 10 entries" dropdown. A search bar is also present. The main content is a table listing files with columns: FileName, FileSize, FileSpecification, FileMark, and Options. The table contains six entries, each with a file name, size, specification, mark, and options (Edit, Details, Delete).

FileName	FileSize	FileSpecification	FileMark	Options
01077635.pdf	338971	pdf	20	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
04_Module_2_Python_Threads.m4v	26680693	test	22	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
1903f234d3ba4da_ek.pdf	188755	test	3	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
23-Concurrency-Examples.pdf	125970	test	5	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
BGP-full.pdf	1226009		24	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Capture.PNG	17282	wireshark capture for http protocol	20	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

الشكل 34 واجهة إدارة تطبيق المخدم المركزي

### 3.6- بعض الحلول التقنية المفيد ذكرها

تحتاج بيئة العمل إلى أكثر من حاسب للاختبار والتصحيح Debugging فالمشكلة الأولى التي تواجه مثل هذه البيئة هي أن التعديل على الرمز Code في حاسب يجب أن يتزامن في كافة الحواسيب ولا يتم التعديل يدوياً على كل حاسب. يتمثل الحل في استخدام الخدمة السحابية المجانية Drop Box فما علينا إلا اختيار مجلد المشروع الذي نود مشاركته ومع من نريد مشاركته ليظهر أي تعديل يعدله أحد المشاركين عند الجميع. أما المشكلة الثانية هي اختبار البرنامج في المدينة السكنية، لكن البنية الشبكية في المدينة السكنية تدعم العناوين المنطقية من النسخة السادسة IPV6 ولا تدعم العناوين المنطقية من النسخة الرابعة بالتالي لا يمكننا الاتصال من التطبيق إلى حواسيب المخبر التي تعمل على النسخة الرابعة IPV4، الحل إعداد مخدم اتصال شبكية افتراضية VPN على أحد حواسيب المخبر والاتصال به من المدينة السكنية لأخذ عنوان من ضمن مجال Range العناوين التي في المخبر وبالتالي إمكانية تحقيق الاتصال في النظام HiastShare.



## 4.6- الخاتمة وآفاق مستقبلية

تمّ في هذا العمل بناء تطبيق ند لند من أجل مشاركة الملفات يعتمد على البنية المركزية من أجل تسهيل آلية بناء الشبكة وتصميمها، وكذلك تسهيل آلية البحث وتسريعها.

يمكن أن نقوم بتطوير التطبيق من ناحية تحميل الملفات بحيث تحقق أعلى معدل نقل ممكن throughput، أيضاً من تحسين التطبيق من الناحية الأمنية مثل تشفير المعلومات المهمة فلا تتعرض لخطر الهجوم عند انتقالها على الشبكة.



## المصطلحات

الاختصار	التفاصيل	شرح مبسط
FTP	File Transfer Protocol	بروتوكول نقل الملفات
HTTP	Hyper Text Transfer Protocol	بروتوكول النقل الفائق
IP	Internet Protocol	بروتوكول الترابط الشبكي
Mutex	Mutual Exclusion	الإقفال المتبادل
ORM	Object-Relation Mapper	المطابق الغرضي العلاقي
P2P	Peer-to-peer	النند للنند
TCP	Transmission Control Protocol	بروتوكول تحكم النقل
UDP	User Datagram Protocol	بروتوكول برقية معطيات المستخدم
VPN	Virtual private network	شبكة خصوصية افتراضية



## المراجع

- [1] A. Vasudeva, Sandeepan and N. Kumar, "PASE: P2P Network Based Academic Search and File Sharing Application," in *1st International Conference On Computational Intelligence*, 2009.
- [2] Steven J. Vaughan-Nichols, "There Is No Conspiracy Against BITTorrent," 23 June 2005.
- [3] H. H. Tanaka, "Post-Napster: Peer-To-Peer File Sharing Systems Current And Future Issues On Secondary Liability," in *Loyola Of Los Angeles Futertainment Law Review*, 2001.
- [4] T. R. Somro, M. S. Laghari and H. Wahba, "A2A Share: Towards Multilingual Academic P2P," in *International Conference On Sociality and Economic Development, (IPEDR)*, 2011.
- [5] Siu Man Lui and Sai Ho Kwok, "Interoperability of Peer-To-Peer File Sharing Protocols," *ACMSIGecomExchanges*, vol. 3, no. 3, pp. 1-9, 3 August 2002.
- [6] "Architecture of peer-to-peer-systems - an overview," [Online]. Available: [http://archiv.iwi.uni-hannover.de/lv/seminar\\_ss03/Linck/hp2p.htm](http://archiv.iwi.uni-hannover.de/lv/seminar_ss03/Linck/hp2p.htm). [Accessed 20 September 2016].
- [7] Choon Hong Ding, Darana Nutanong and Rajkumar Buyya, "Peer-To-Peer Networks for Content Sharing," The University of Melbourne, 2002.
- [8] "Bit Torrent," 10 September 2016. [Online]. Available: <https://en.wikipedia.org/wiki/BitTorrent>. [Accessed 20 September 2016].
- [9] M. Eric Johnson, Dan McGuire and Nicholas D. Willey, "The Evolution of the Peer-To-Peer File Sharing Industry and the Security Risks for Users," in *Proceedings of the 41st Hawaii International Confernce on Systems Sciences*, 2008.
- [10] David B. Makofske, Michael J. Donahoo and Kenneth L. Calvert, *TCP\_IP Sockets In C# - Practical Guide For Programmers*, San Francisco: Elsevier Inc, 2004.

