



دانشکده فناوری اطلاعات و کامپیوتر

کارشناسی مهندسی کامپیوتر-نرم افزار

عنوان :

پیام رسان اندروید تحت پروتکل xmpp

استاد راهنما :

دکتر سیامک سرمدی

نام و نام خانوادگی دانشجوی :

فرید عمرزاده

شماره دانشجویی :

۹۲۱۱۲۲۲۰۵

تابستان ۹۵

فهرست مطالب

فهرست مطالب

۴	چکیده
۵	فصل اول
۶	پیشگفتار
۷	۱-۱ برنامه نویسی اندروید
۷	۱-۱-۲ معماری اندروید
۹	۱-۲ پروتکل xmpp
۱۰	۱-۲-۱ معماری xmpp
۱۲	فصل دوم
۱۲	۲-۱ محیط برنامه نویسی اندروید استادیو (Android Studio)
۱۵	۲-۲ سرور openfire
۱۶	۲-۳ نرم افزار pidgin
۱۸	فصل سوم
۱۸	۳-۱ کتابخانه ها و API ها
۱۹	۳-۱-۲ RecyclerView
۲۰	۳-۱-۳ Broadcast ها
۲۳	۳-۱-۴ handler ها
۲۴	۳-۱-۵ Service ها
۲۶	۳-۲ جریان کلی اجرای پیام رسان
۳۲	فصل چهارم
۳۲	نتیجه گیری
۳۲	پیشنهادهات

چکیده

در ابتدا نگاهی اجمالی به برنامه نویسی اندروید و محیط برنامه نویسی اندروید استادیو (Android Studio) خواهیم داشت و سپس در مورد پروتکل xmpp صحبت خواهیم کرد . توضیحاتی در مورد سرور openfire خواهیم داشت و سپس بر روی تکنیک های مورد استفاده در پروژه صحبت خواهیم کرد و به شرح کلی پروژه و توضیحاتش خواهیم پرداخت . در انتها ،خروجی پروژه یک برنامه کاربردی برای سیستم عامل اندروید خواهد بود.

فصل اول

مقدمه

پیشگفتار

اکثر برنامه‌های محبوب سیستم‌عامل اندروید و iOS در دسته برنامه‌هایی با قابلیت ارتباطی قرار می‌گیرند. نرم‌افزارهایی مجهز به قابلیت‌هایی مانند ارسال پیامک، برقراری تماس یا برنامه‌هایی برای دسترسی به شبکه‌های اجتماعی، از این دسته هستند.

شاید یکی از دلایل این امر آن است که گفت‌وگو و برقراری ارتباط با دیگران یک نیاز اصلی در میان انسان‌ها است. همان‌طور که در مکالمات دنیای واقعی هم مشاهده می‌کنیم، کاربران این برنامه‌ها از روش‌ها و الگوهای متفاوتی برای برقراری ارتباط با دیگران استفاده می‌کنند.

برخی جملات کوتاه و ساده، برخی دیگر استفاده از استیکرها و عکس‌های مختلف و برخی دیگر از کاربران نیز شرکت در گروه‌ها را ترجیح می‌دهند. اما از نگاه سازندگان، با توجه به وجود روش‌های مختلف برای تولید این سبک برنامه‌ها، ایده‌های مختلفی هم پیاده‌سازی شده است. سازندگان این نوع برنامه‌ها هر کدام از رویکرد‌ها و پروتکل‌های مختص خود برای طراحی برنامه‌هایشان استفاده می‌کنند. یکی از این پروتکل‌ها که قابلیت ایجاد برنامه‌های پیام‌رسان را دارد پروتکل XMPP می‌باشد که در ادامه به آن می‌پردازیم.

۱-۱ برنامه نویسی اندروید

۱-۱-۱ نگاه اجمالی به اندروید

شرکت توسعه دهنده سیستم عامل اندروید که توسط اندی رابین، ریچ ماینرز، نیک سیرز، کریس وایت پایه گذاری شده بود، در سال ۲۰۰۵ توسط شرکت گوگل خریداری شد. پس از آن open handset alliance بود که توسط چندین شرکت از جمله سامسونگ، ال جی، اینتل و... به توسعه اندروید ادامه دادند. اولین گوشی اندرویدی در سال ۲۰۰۸ توسط htc عرضه شد.

سیستم عامل اندروید بر پایه هسته لینوکس ساخته شده و کدهای برنامه نویسی آن به زبان جاوا نوشته می شوند. تاکنون بیش از میلیون ها نرم افزار برای اندروید نوشته شده و هر روز بر تعداد آنها اضافه می شود. اندروید یک چهارچوب است که ساخت برنامه های نوآورانه و بازی برای دستگاه های تلفن همراه در یک محیط زبان جاوا را فراهم می کند.

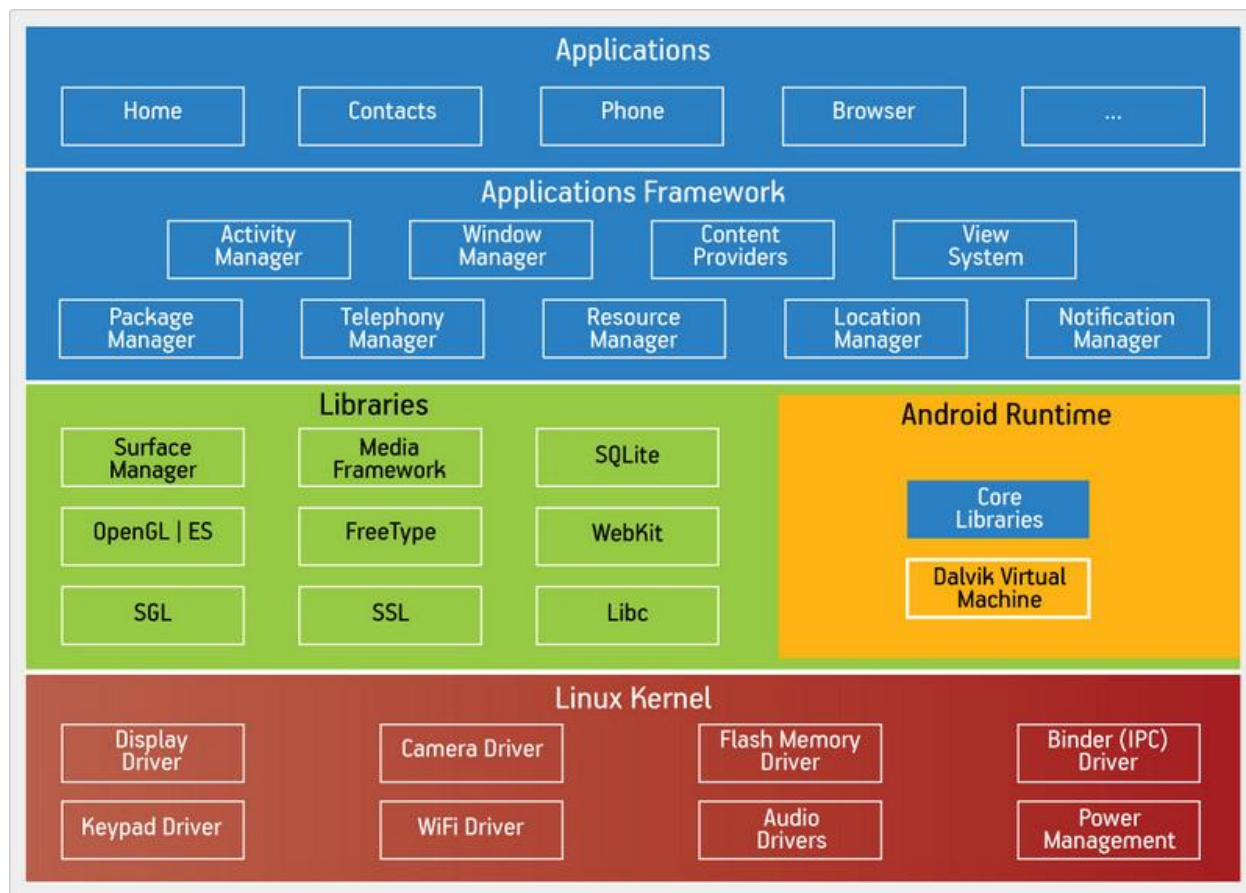
معروفترین زبان برنامه نویسی برای اندروید جاوا است. جاوا توسط شرکت اوراکل توسعه داده شده است و توسط گستره عظیمی از دستگاه ها پشتیبانی می شود. برای برنامه نویسی اندروید لازم است که حداقل syntax زبان جاوا را بلد باشید. اندروید برای قسمت گرافیکی برنامه ها هم از XML استفاده میکند.

این زبان طبق آخرین آمار منتشر شده ۱۰۴ میلیارد دستگاه اکتیو شده دارد. در ایران نیز طبق آخرین آمار منتشر شده از مارکت معروف کافه بازار، بیش از ۲۰ میلیون دستگاه اندرویدی (که کافه بازار را نصب دارند) وجود دارد.

برای برنامه نویسی اندروید شما به یک بسته نرم افزاری توسعه دهنده (SDK) که توسط گوگل منتشر شده است نیاز دارید. همچنین یک نرم افزار برای برنامه نویسی تحت اندروید (IDE) نیاز است. که به سفارش گوگل بهترین نرم افزار برای این کار اندروید استودیو می باشد.

۱-۲ معماری اندروید

سیستم عامل اندروید یک پشته برای اجزای نرم افزار ها می سازد که در دید کلی به پنج بخش و چهار لایه اصلی تقسیم می شوند، در تصویر زیر نمودار معماری نشان داده شده است :



شکل_۱ معماری اندروید

هسته لینوکس

پایه و اساس سیستم عامل اندروید هسته لینوکس است. برای مثال مباحثی مانند مدیریت نخ ها و همچنین مدیریت حافظه در (ART) Android Runtime برپایه هسته لینوکس بنا شده است. همچنین این موضوع باعث شده که اندروید از مزایا و امکانات امنیتی هسته لینوکس نیز بهره ببرد.

کتابخانه ها

در بالای هسته لینوکس مجموعه ای از کتابخانه ها وجود دارند از جمله مرورگر WebKits که از مرورگر های open source می باشد. کتابخانه های دیگری نیز قابل تشخیص می باشند مانند پایگاه داده SQLite که یک بانک اطلاعاتی

قوی برای ذخیره سازی داده ها و به اشتراک گذاری آنهاست ، از دیگر کتابخانه ها میتوان به پخش صوت و تصویر ، کتابخانه های SSL برای امنیت نیز اشاره کرد

Android Runtime

بخش سوم از معماری Android Runtime می باشد و از پایین لایه دوم است این لایه یک جزء کلیدی به نام Dalvik Virtual Machine تولید می نماید که یک نوع از ماشین مجازی جاوا ویژه و بهینه سازی شده برای اندروید است

DALVIK VM از ویژگی های هسته لینوکس مانند مدیریت حافظه استفاده می کند

چند نخ، در زبان جاوا ذاتی است

DALVIK VM هر نرم افزار اندروید را به اجرا در پروسه خود قادر می سازد ، Android Runtime همچنین مجموعه ای از کتابخانه های هسته را فراهم می کند که قادر است به برنامه نویسان اندروید اجازه دهد از زبان استاندارد برنامه نویسی جاوا برای نوشتن برنامه های کاربردی اندروید استفاده نمایند .

Application Framework

Application Framework ها بسیاری از خدمات سطح بالاتر لایه ها را فراهم می کند تا برنامه های کاربردی را در قالب کلاس های جاوا به خوبی اجرا شوند برنامه نویسان می توانند از این خدمات را در برنامه های کاربردی خود استفاده نمایند.

Applications

تمام نرم افزار های اندروید را لایه Applications پیدا کنید پس از نصب نرم افزار شما دسترسی را فقط در این لایه خواهید داشت از جمله برنامه های کاربردی مانند مخاطب ها، مرورگر، بازی و. غیره.

۱-۲ پروتکل xmpp

Xmpp پروتکل حضور و گسترش پیام یک تکنولوژی بر پایه xml برای ارتباط real-time است که توانایی وسیعی را در اختیار دامنه وسیعی از برنامه ها شامل پیام رسانی فوری (IM)، حضور و همکاری میدهد. این پروتکل در ابتدا jabbar نام داشت زیرا توسط کمیته jabbar در سال ۱۹۹۹ توسعه داده شد.

برای فهم بهتر از پروتکل به موارد زیر دقت کنید

P-Protocol

xmpp یک پروتکل است یعنی مجموعه ای از استانداردها که به سیستم ها اجازه میدهد با هم صحبت کنند. xmpp بطور گسترده در وب استفاده میشود. پروتکل بر پایه XSF است.

P-Presence

شناسه حضور (presence) به سرور اطلاع میدهد که شما online/offline/busy هستید. در اصطلاح فنی ، حضور حالت موجودی xmpp را مشخص میکند. در اصطلاح غیر فنی یعنی آیا شما برای دریافت پیام آماده هستید یا خیر.

M-Messaging

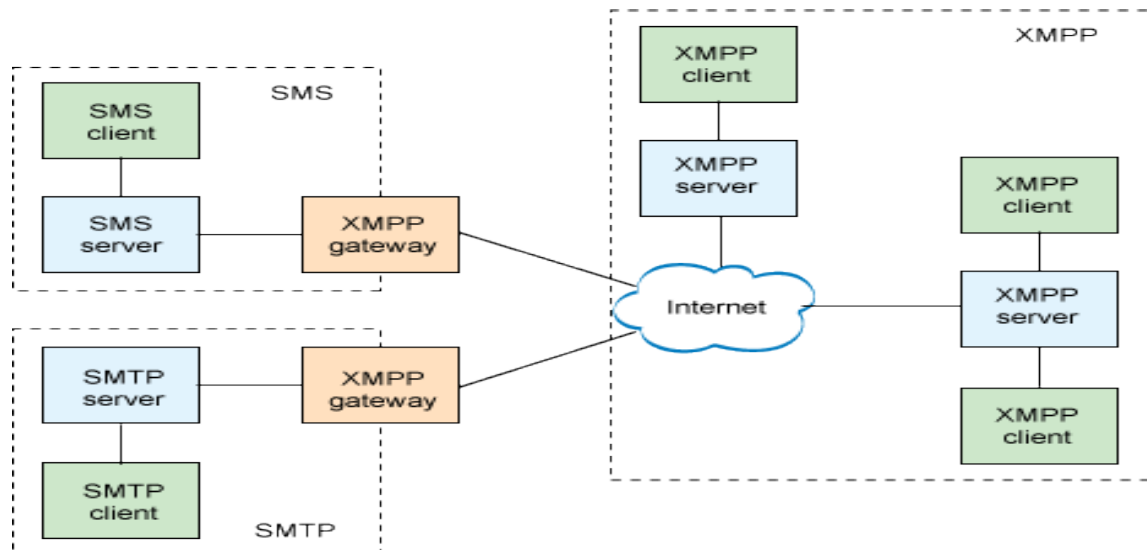
بخش "پیام" پروتکل قسمتی است که شما آن را می بینید. پیام فوری (IM) که بین کلاینت ها ارسال میشود. xmpp برای ارسال تمامی پیام ها در زمان real-time و با استفاده از یک مکانیزم قوی push ، طراحی شده است. در هر حال وجود مکانیسم وب و بارگذاری شبکه باعث میشود که زمان آن واقعا real-time نباشد.

X-eXtensible

در یک استاندارد باز و با استفاده از سیستم های باز طراحی شده است. xmpp برای گسترش طراحی شده است و یک پروتکل open source است.

۱-۲-۱ معماری xmpp

xmpp شباهت هایی به سایر پروتکل های لایه کاربردی مانند SMTP دارد. در این معماری ها یک کاربر با یک نام منحصر به فرد میتواند با کاربر دیگر ، از طریق سرور هماهنگ کننده ، ارتباط داشته باشد. هر کاربر بخش سمت کاربر پروتکل را پیاده سازی میکند و سرور ها قابلیت مسیر یابی را فراهم می کنند. سرور ها هم می توانند به منظور مسیریابی در دامنه های متفاوت با هم ارتباط برقرار کنند به این منظور Gateway هایی میتوانند وجود داشته باشند.



شکل ۲ معماری xmpp

• آدرس ها در xmpp

در xmpp آدرس ها به شکل زیر هستند

Node @ domain/resource

که node همان نام کاربری است و domain دامنه میباشد و resource هم مشخص کننده این است که کلاینت از چه برنامه کاربردی برای برقراری ارتباط استفاده میکند.

فصل دوم

نگاهی به نرم افزار های استفاده شده در پروژه

در این فصل به معرفی و بررسی نرم افزار ها و امکانات بکار گرفته در پروژه خواهیم پرداخت. همانند:

- محیط برنامه نویسی android studio
- سرور openfire
- نرم افزار پیام رسان pidgin

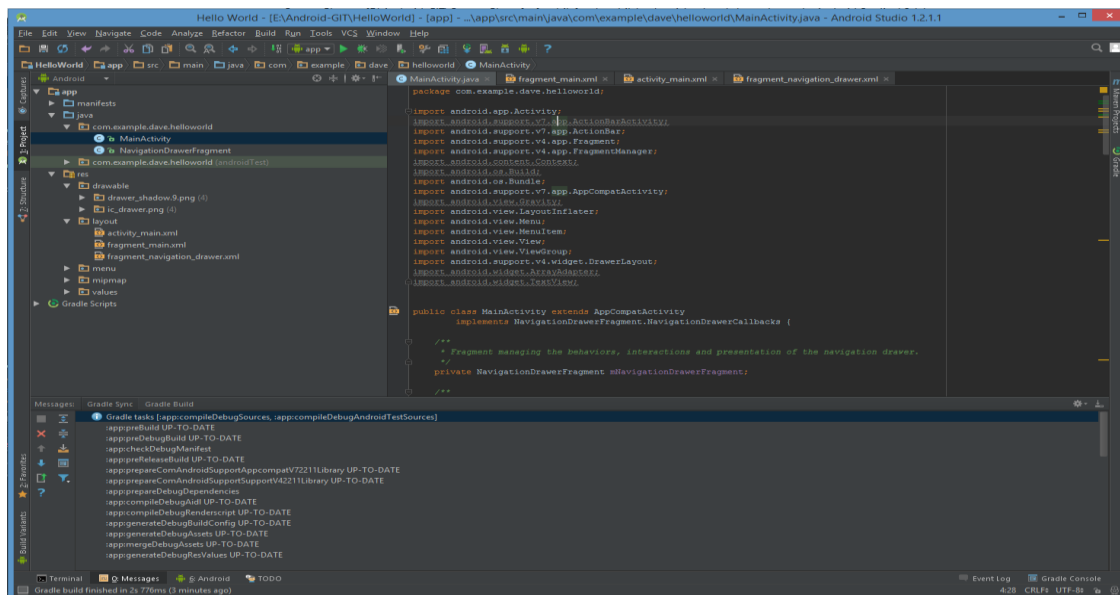
۲-۱ محیط برنامه نویسی اندروید استادیو (Android Studio)

اندروید استادیو یک محیط برنامه نویسی برای پلتفرم اندروید است. این برنامه در ۱۶ می ۲۰۱۳ توسط مدیران گوگل در کنفرانس گوگل آی/او معرفی شد. از ماه جوئن ۳ ۲۰۱۳ نسخه پیش نمایش این برنامه (به صورت رایگان) برای امتحان در دسترس توسعه دهنده ها قرار گرفت. این محیط بر اساس نرم افزار محبوب جت برینز طراحی شده است، این استودیو بطور اختصاصی برای اندروید طراحی شده. هم اکنون این استودیو برای دانلود بر روی ویندوز، مک و لینوکس در دسترس می باشد.

چندین ویژگی جدید در این نرم افزار برای کاربران می باشد:

- چیدمان زنده: ویرایشگر WYSIWYG کدنویسی زنده – رندر هم زمان برنامه
- کنسول توسعه دهنده: راهنمای بهینه سازی – کمک برای ترجمه – ردیابی ارجاع – طریقه استفاده
- ارائه نسخه بتا و اجرای صحنه
- پشتیبانی از ساخت مبتنی بر Gradle
- رفع و عیب یابی مخصوص اندروید
- ابزار لینت برای گرفتن عملکرد، قابلیت استفاده، نسخه سازگار با مشکلات دیگر
- قابلیت Proguard و امضای برنامه
- مبتنی بر الگوی wizard برای ایجاد طرح اندروید و اجزای مشترک

- ویرایشگر طرح بندی غنی که به شما اجازه کشیدن و رها کردن UI کامنت‌ها را می‌دهند. گزینه‌ای برای تنظیمات پیش نمایش طرح بندی در صفحه نمایش چندگانه



شکل ۳_ اندروید استادیو

اجرای فوری با Instant run

می‌توانید کدها و منابع برنامه تان را تغییر بدهید و همزمان تغییرات را روی دستگاه یا شبیه ساز مشاهده کنید.



شکل ۴_ اجرای فوری

ویرایشگر متن هوشمند

این ویرایشگر متن هوشمند که در هر مرحله شما را همیاری می کند می توانید بهتر کدنویسی کنید، سریع تر کار کنید و بهره وری خود را افزایش دهید.

سیستم build قدرتمند و انعطاف پذیر

به راحتی می توانید کتابخانه های کد را به پروژه تان اضافه کنید و از یک پروژه، چندین Build مختلف تولید کنید. اندروید استودیو به همراه Gradle، یک اتوماسیون Build با عملکرد بالا، مدیریت وابستگی قدرتمند و پیکربندی های Build قابل شخصی سازی ارائه می دهد.



شکل_۵ سیستم build

قالب های کد و ادغام با github

می توانید پروژه ها را با Template ها شروع کنید تا الگوهایی مانند منوی ناوبری و ViewPager داشته باشید یا اینکه نمونه های کد گوگل را از GitHub وارد کنید.

با پنجره های Project Wizard اندروید استودیو، اضافه کردن کد به پروژه های جدید از همیشه راحت تر است.



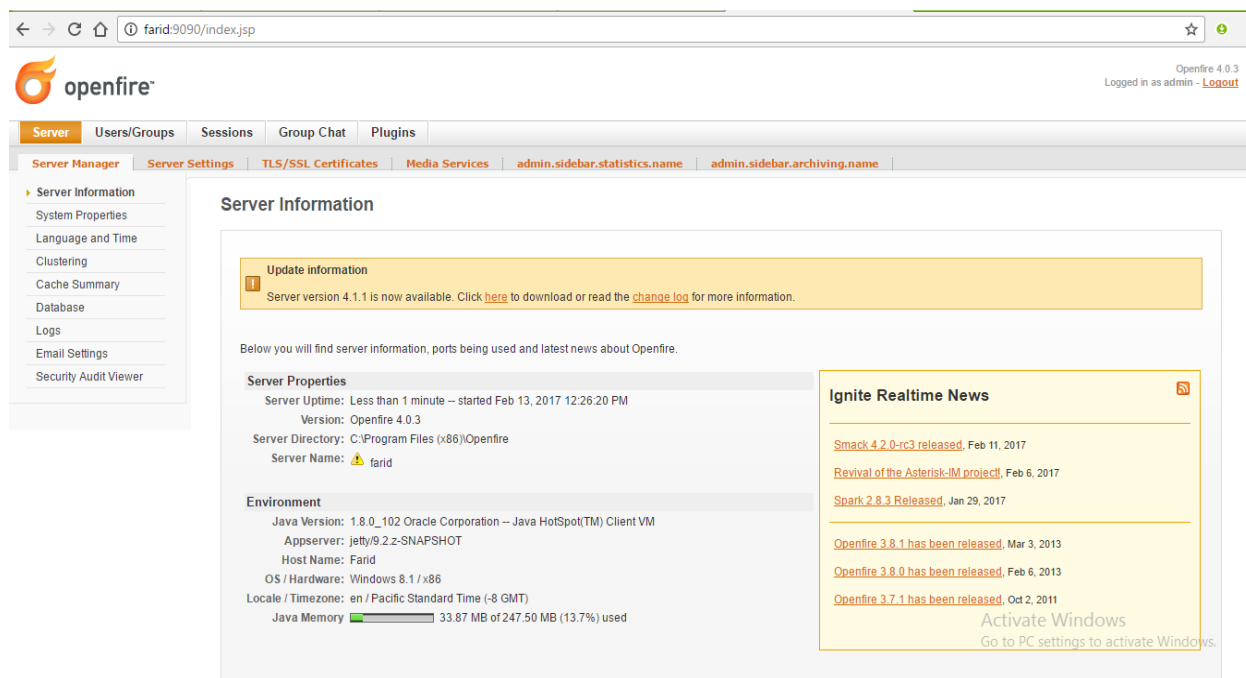
شکل_۶ استفاده از github

۲-۲ سرور openfire

Openfire یک سرور همکاری real-time تحت لیسانس موسسه اوپن سورس اپاچی است و فقط از پروتکل xmpp استفاده میکند. این سرور ابتدا در سال ۲۰۰۲ و توسط jive software ایجاد شد. بعد ها در سال ۲۰۰۵ به wildfire تغییر نام پیدا کرد. بخاطر معضلات تجاری در سال ۲۰۰۷ به openfire تغییر نام پیدا کرد و موسسه jive تا به امروز از آن پشتیبانی میکند.

ویژگی ها

- پنل مدیریتی تحت وب
- قابلیت شخصی سازی
- حمایت از ssl/tls
- قابلیت اتصال به پایگاه های داده
- قابلیت اتصال به LDAP
- مستقل از سیستم عامل، نوشته شده با زبان برنامه نویسی جاوا
- قابلیت حمایت همزمان از ۵۰۰۰۰ کاربر



شکل ۷ سرور openfire

۲-۳ نرم افزار pidgin

Pidgin یک نرم افزار پیام رسانی فوری open source است که قابلیت اجرا بر روی پلاتفرم های مختلف را دارد ، بر پایه کتابخانه libpurple ایجاد شده که طیف وسیعی از پروتکل های پیام رسانی را پشتیبانی می کند. به کاربر این اجازه را می دهد که بصورت همزمان با استفاده از یک برنامه به چندین سرویس دسترسی داشته باشد.

در سال ۲۰۰۷ تعداد کاربران pidgin سه میلیون نفر بودند.

تاریخچه

این برنامه توسط یکی از دانشجویان سال آخر دانشگاه Auburn University نوشته شده بود و هدف آن شبیه سازی برنامه پیام رسان AOL بود . اولین نسخه آن در دسامبر ۱۹۹۸ ارائه شد. برنامه شبیه سازی بر پایه مهندسی معکوس نبود اما بر پایه پروتکلی بود که AOL در وبسایتش منتشر نموده بود در آپریل ۲۰۰۷ نام pidgin برای برنامه انتخاب شد.

در جولای ۲۰۰۶ Pidgin امتیاز ۷ از ۷ را توسط بنیاد پیشره الکترونیک به خاطر پیام رسانی امن دریافت نمود.

مزایا

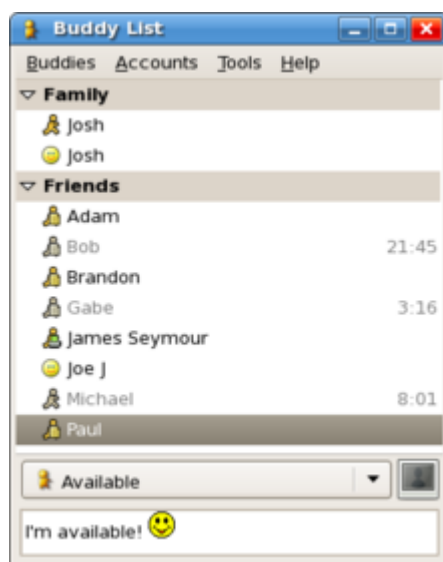
Pidgin از یک واسط گرافیکی برای libpurple با GTK+ استفاده می کند. Libpurple از پروتکل های بسیار زیادی از پیام رسان ها پشتیبانی می کند. pidgin از سیستم عامل های متفاوتی مثل windows و سیستم های مشابه unix مانند linux و BSDs و AmigaOS پشتیبانی می کند.

- Pluggable بودن

Pidgin چگونه ای طراحی شده است که از plugin های متفاوت برای توسعه خود استفاده می کند. Plugin ها برای اضافه کردن پشتیبانی بیشتر به پروتکل ها مورد استفاده قرار می گیرند. برای مثال سیستم فشرده سازی TLS از قابلیت pluggable برخوردار است که به انواع کتابخانه های TLS اجازه میدهد که به راحتی جایگزین یکدیگر شوند.

- مخاطب ها

به جای مدیریت کاربران از پروتکل های متفاوت میتوان همه آن ها را در یک مخاطب جمع کرد و به آن نام مستعار داده و در گروه های متفاوت قرار داد.



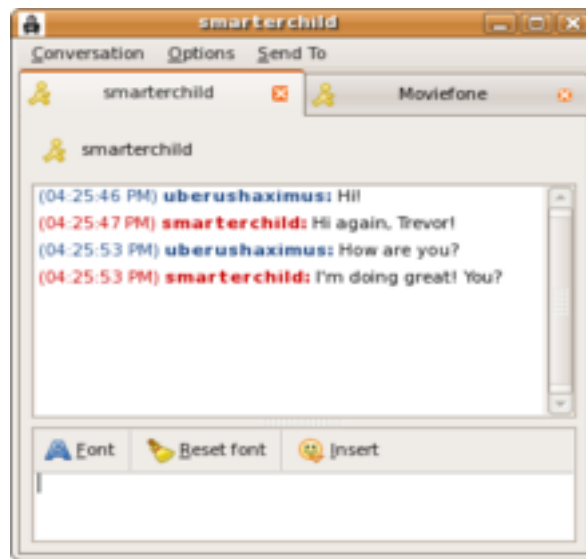
شکل ۸ نمایش مخاطب ها در pidgin

- انتقال فایل

Pidgin از انتقال فایل برای بسیاری از پروتکل ها پشتیبانی می کند. انتقال مستقیم peer-to-peer فایل در پروتکل های xmpp و msn را پشتیبانی می کند.

- چت صدا و تصویری

از نسخه ۲٫۶ به بعد pidgin از تماس های صدا/تصویر را پشتیبانی می کند. از جولای ۲۰۱۵ تماس ها فقط از طریق پروتکل xmpp امکان پذیر شدند.



شکل ۹ چت متنی در pidgin

فصل سوم

مشروح کار های انجام شده در پروژه

در این فصل به روند کلی انجام پروژه و تکنیک های بکارگرفته در پروژه خواهیم پرداخت . تمامی کدهای منبع پیوست شده اند.

۳-۱ کتابخانه ها و API ها و تکنیک ها

Smack ۳-۱-۱

Smack یک کتابخانه متن باز برای بخش سمت کاربر پروتکل xmpp است . که از آن برای پیام رسانی تحت پروتکل فوق الذکر استفاده می شود. این کتابخانه را میتوان به راحتی به هر پروژه ای افزود. و از تمامی امکانات پروتکل xmpp پشتیبانی می کند از push notification گرفته تا presence-enabling .

- مزایای اصلی smack

۱. ساده برای استفاده و قدرتمند بودن

۲. شما را مجبور به کد زنی در سطح پکت نمی کند. smack از ساختار های هوشمند سطح بالایی همچون کلاس های Roster و chat برخوردار است.
۳. لازم نیست که با ساختار xmpp xml آشنایی داشته باشید.
۴. فراهم کردن ارتباط راحت ماشین به ماشین. به شما اجازه میدهد هر نوع property را به پیام هایتان اضافه کنید.
۵. منبع باز بودن تحت لیسانس آپاچی.

Recycler View ۳-۱-۲

بسیاری از برنامه های اندروید محتاج به نمایش گرافیکی اجزای مبتنی بر مجموعه های داده و یا داده هایی که بصورت مکرر عوض میشوند، هستند. برای مثال برنامه های پخش صوت ممکن است به اطلاعات مربوط به هزاران آلبوم را نمایش دهد ولی فقط ۱۲ تا از موزیک ها در هر لحظه بر روی صفحه باشند. اگر برنامه برای هر کدام از این آلبوم ها یک widget بسازد حجم بسیار زیادی از حافظه را مصرف خواهد کرد. همچنین با حرکت از هر widget به widget دیگر باید هر بار widget قبلی را از بین برده و بعدی را بسازد و این کار باعث کندی برنامه خواهد شد.

برای حل این مشکل اندروید مجموعه اشیاای Recycler را فراهم کرده است. Recycler و کلاس ها و واسط های گرافیکی مرتبط با آن به شما کمک می کند که یک رابط گرافیکی پویا که بصورت کارا اجرا میشود را پیاده سازی کنید. شما میتوانید از کلاس هایش همانگونه که هستند استفاده کنید و یا آن را تغییر دهید.

ساختار نمایش پویا

در پشت مدل RecyclerView کامپوننت های مختلف و متفاوتی برای نمایش داده ها با هم کار می کنند. بعضی از این کامپوننت ها ممکن است در شکل تغییر نشده شان مورد استفاده قرار گیرند. برای مثال برنامه شما ممکن است مستقیماً از RecyclerView استفاده کند • در دیگر موارد ، برنامه شما لازم است که این کامپوننت ها را گسترش و تغییر دهد برای مثال هر برنامه ای که از RecyclerView استفاده میکند لازم است که view holder مخصوص خودش را تعریف کند. که میتوان آن را از کلاس انتزاعی RecyclerView.ViewHolder بسط داد.

کانتینر کلی برای رابط گرافیکی پویای شما RecyclerView است که شما آن را به layout (fragment یا activity) خود اضافه می کنید. RecyclerView در عوض خودش را با view های کوچکی که ایتم هایش را تشکیل می دهد پر می کند

ایتم ها به تنهایی توسط اشیاای view holder ارائه می شوند. این اشیا نمونه هایی از کلاس RecyclerView.ViewHolder هستند که آن را بسط داده اید. هر کدام از holder ها دارای view مخصوص به خودش است تنها یک ایتم را نمایش می دهد.

اشیای `view holder` توسط `adapter` مدیریت می شوند. که شما از بسط دادن کلاس `RecyclerView.Adapter` آن را ایجاد می کنید. `Adapter` ها تعداد `holder` هایی که لازم هستند را می سازد. `Adapter` همچنین `holder` ها را به داده هایی که باید نمایش دهد متصل می کند. این کار را با دادن یک موقعیت به `holder` و فراخوانی متد `onBindViewHolder()` کلاس `adapter` انجام می دهد.

`RecyclerView` کار های بهینه زیادی را انجام می دهد

- زمانی که `view` برای اولین بار پر می شود ، `holder` هایی را به هر طرف لیست اضافه می کند. برای مثال اگر که شما قرار است آلبوم های ۰ تا ۹ را نمایش دهید `RecyclerView` موقعیت ایتm شماره ۱۰ را هم ایجاد می کند تا اگر `scroll` بخورد آماده باشد.
- همگام با `scroll` خوردن `RecyclerView` ، `holder` های جدیدی را می سازد
- زمانی که داده های نمایش داده شده تغییر کنند ، برنامه با فراخوانی متد `notify()` از کلاس `adapter` به آن خبر می دهد. سپس `adapter` ایتm ها را تغییر می دهد

۳-۱-۳ Broadcast ها

برنامه های اندرویدی میتوانند پیام های `broadcast` را از سیستم اندروید و یا دیگر برنامه ها ارسال و دریافت کنند، این `Broadcast` ها زمانی که یک رخداد مورد نظر رخ دهد ارسال می شوند. برای مثال سیستم اندروید `broadcast` های مختلفی را ارسال می کند همانند راه اندازی سیستم و یا شروع به شارژ شدن. برنامه ها هم ممکن است `broadcast` های مختلفی را ارسال کنند برای مثال آگاه کردن سایر برنامه ها از یک رخدادی که شاید برایشان جالب توجه باشد (برای مثال داده جدیدی دانلود شد)

برنامه ها می توانند مشخص کنند که `broadcast` های مشخصی را دریافت کنند. زمانی که `Broadcast` فرستاده شد سیستم بصورت اتوماتیک آن را به برنامه هایی که مشخص کرده اند به این نوع `broadcast` علاقه مند هستند هدایت می کند

همانند گفتگو های عمومی، `broadcast` ها برای ارسال پیام به خارج از برنامه می توانند مورد استفاده قرار بگیرند.

• Broadcast های سیستمی

سیستم بصورت اتوماتیک `broadcast` هایی را هنگام رخ دادن رخداد های مختلف می فرستد مانند زمانی که سیستم به حالت پرواز در می آید یا از حالت پرواز خارج می شود. `Broadcast` های سیستم برای همه برنامه هایی که مشخص کرده اند به دریافت آنها علاقه مند هستند فرستاده می شود.

پیام broadcast سیستم در داخل intent ای قرار گرفته شده است که با یک رشته نوع رخداد صورت گرفته را مشخص می کند (برای مثال android.intent.action.AIRPLANE_MODE intent) هم چنین ممکن است حاوی اطلاعات اضافی دیگری هم باشد که در داخل فیلد extra آن قرار می گیرد. برای مثال مد حالت پرواز شامل یک مقدار Boolean نیز می باشد که مشخص می کند آیا حالت پرواز روشن است یا خیر.

- دریافت Broadcast ها

برنامه ها از دو طریق broadcast دریافت می کنند : اعلام در manifest و context-receiver

از طریق اعلام در manifest

۱. مشخص کردن receiver در manifest

```
<receiver android:name=".MyBroadcastReceiver" android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.intent.action.INPUT_METHOD_CHANGED" />
    </intent-filter>
</receiver>
```

۲. پیاده کردن BroadcastReceiver و پیاده سازی متد onReceive(Context,intent) مانند مثال زیر

```
1) public class MyBroadcastReceiver extends BroadcastReceiver {
    private static final String TAG = "MyBroadcastReceiver";
    @Override
    public void onReceive(Context context, Intent intent) {
        StringBuilder sb = new StringBuilder();
        sb.append("Action: " + intent.getAction() + "\n");
        sb.append("URI: " + intent.toUri(Intent.URI_INTENT_SCHEME).toString() +
            "\n");
        String log = sb.toString();
        Log.d(TAG, log);
        Toast.makeText(context, log, Toast.LENGTH_LONG).show();
    }
}
```

دریافت کننده های context-receiver

(۱) ایجاد یک نمونه از BroadcastReceiver

```
BroadcastReceiver br = new MyBroadcastReceiver();
```

(۲) ایجاد یک IntentFilter و ثبت نام برای دریافت با فراخوانی
registerReceiver(BroadcastReceiver, IntentFilter)

```
IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);  
intentFilter.addAction(Intent.ACTION_AIRPLANE_MODE_CHANGED);  
this.registerReceiver(br, filter);
```

(۳) توقف دریافت broadcast

برای توقف دریافت broadcast ها متد unregisterReceiver(BroadcastReceiver) را فراخوانی کنید

دریافت کننده های context-receiver ها فقط تا زمانی که context معتبر باشد broadcast دریافت می کنند، برای مثال، اگر شما در یک activity ثبت نام کرده باشید فقط تا زمانی که activity در حال اجرا می باشد دریافت می کند.

• ارسال Broadcast ها

اندروید برای ارسال broadcast ها سه روش فراهم کرده است

۱. متد sendOrderedBroadcast(Intent, String) در هر زمان broadcast ها را به یک receiver

می فرستد. به محض اجرا شدن receiver می تواند آن را به receiver دیگر بفرستد و یا جلوی پخش آن را بگیرد

۲. متد sendBroadcast(Intent) به همه receiver ها بصورت نامنظم broadcast می فرستد. به این

روش ارسال نرمال می گویند. این روش بسیار موثر است ولی receiver ها قابل به استفاده از نتایج سایر receiver ها نیستند.

```
Intent intent = new Intent();
intent.setAction("com.example.broadcast.MY_NOTIFICATION");
intent.putExtra("data", "Notice me senpai!");
sendBroadcast(intent);
```

۳. متد `LocalBroadcastManager.sendBroadcast` برای ارسال broadcast ها به receiver هایی که در همان برنامه قرار دارند مورد استفاده قرار می گیرد.

۳-۱-۴ handler ها

یک handler به شما اجازه می دهد که پیام و اشیایی که قابلیت اجرا دارند و در ارتباط با `MessageQueue` نخ هستند را فرستاده و پردازش کنید. هر handler با یک نخ و پشته نخ همکاری دارد. زمانی که شما یک handler جدید می سازید به نخ/پشته نخ `bound` می شود. از این نقطه handler پیغام ها و یا اشیای قابل اجرا را به نخ می فرستد و آنهایی که از پشته نخ خارج می شوند را اجرا می کند.

دو استفاده اصلی handler شامل (۱) زمان بندی پیام ها و اشیای قابل اجرا برای اجرا شدن در زمان هایی در آینده و (۲) وارد کردن یک عملیات برای اجرا شدن در نخی به غیر از نخ فعلی شما

زمانبندی پیام ها با استفاده از متد های `post(Runnable)`, `postAtTime(Runnable, long)`, `postDelayed(Runnable, long)`, `sendEmptyMessage(int)`, `sendMessage(Message)`, `sendMessageAtTime(Message, long)`, `sendMessageDelayed(Message, long)` می تواند اجرا شود. نسخه `post` به شما اجازه می دهد که با اشیای قابل اجرا کار کنید و نسخه `message` به شما اجازه می دهد با پیغام ها کار کنید.

هنگام `post` و یا `send` کردن به یک handler می توانید تعیین کنید که پردازش به محض آماده بودن پشته نخ انجام گیرد و یا یک تاخیر مشخص قبل از اجرای آن در نظر بگیرید و یا اینکه یک مدت زمان خاص برای پردازشش مشخص کنید.

زمانی که فرآیندی برای برنامه شما ایجاد می شود، نخ اصلی آن به مراقبت و اجرای اشیای سطح بالای برنامه (`Activity`, `broadcastreceiver`, ...) اختصاص می یابد. و شما می توانید نخ های مخصوص خود را بسازید و از طریق handler ها با نخ اصلی گفتگو داشته باشید.

۳-۱-۵ Service ها

سرویس یک کامپوننت است که میتواند عملیات های پشت زمینه طولانی را به انجام برساند و رابط گرافیکی ارائه نمی دهد. حتی اگر کاربر به یک برنامه دیگر وارد شده و از برنامه فعلی خارج شود سرویس به کار پشت زمینه ای خود ادامه می دهد. به علاوه یک کامپوننت می تواند به یک سرویس bind شود تا با آن تعامل داشته باشد و یا حتی کار های بین پروسه ای IPC انجام دهد. برای مثال یک سرویس می تواند تراکنش های شبکه را handle کند، موزیک پخش کند، کار I/O انجام دهد . و تمامی این کار ها در پشت زمینه انجام می گیرند.

• پایه

برای ایجاد یک سرویس باید یک زیر کلاس از Service ایجاد کنید و یا از یکی از زیر کلاس هایش استفاده کنید. در پیاده سازی خود باید بعضی از متد هایی که کار های اصلی سرویس را انجام می دهند و مکانیسمی برای bind شدن به سرویس ارائه می دهند را override کنید. متد های زیر مهمترین متد های سرویس هستند

onStartCommand()

سیستم این متد را با صدا زدن startService() زمانی که یک کامپوننت (مثلا activity) می خواهد سرویس اجرا شود فراخوانی می کند. زمانیکه این متد اجرا شد سرویس بصورت نامحدود در پشت زمینه اجرا می شود. اگر شما این متد را پیاده سازی کردید وظیفه شماست که آن را متوقف کنید با فراخوانی stopSelf() یا stopService() اگر شما فقط binding را می خواهید لزومی ندارد که این متد را فراخوانی کنید.

onBind()

سیستم این سرویس را زمانی که یک کامپوننت دیگر بخواهد به سرویس bind شود با صدا زدن bindService() فراخوانی می کند. در پیاده سازی تان باید یک رابط گرافیکی تهیه کنید که کاربر بتواند با سرویس تعامل داشته باشد با برگرداندن یک IBinder . شما همیشه باید این متد را پیاده سازی کنید در هر حال اگر احتیاجی به binding ندارید میتوانید مقدار null برگردانید .

onCreate()

سیستم این متد را برای مقدار دهی اولیه فراخوانی می کند.

onDestroy()

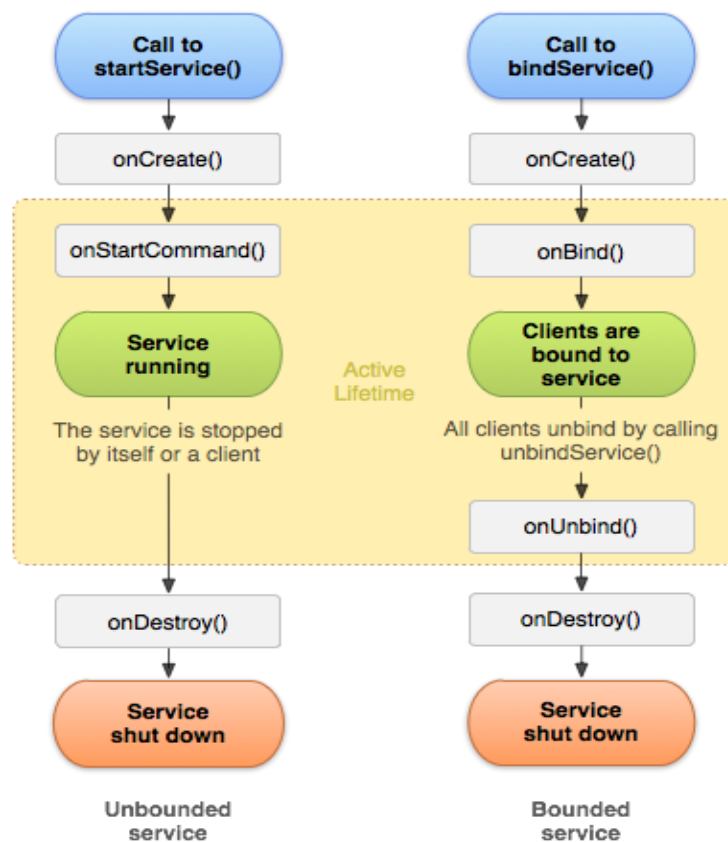
سیستم این متد را زمانی که سرویس دیگر مورد استفاده قرار نمیگیرد و یا دارد از بین میرود فراخوانی می کند. سرویس شما باید این متد را پیاده سازی کند برای پاک کردن منابع مانند نخ ها، registered listener ، receiver ها . این آخرین فراخوانی است که سرویس انجام می دهد .

- اعلان سرویس در manifest

شما باید سرویس را در manifest اعلان کنید

```
<manifest ... >
...
<application ... >
    <service android:name=".ExampleService" />
    ...
</application>
</manifest>
```

- چرخه طول عمر سرویس



شکل ۱۰_ چرخه حیات سرویس

۳-۲ جریان کلی اجرای پیام رسان

جریان کلی اجرای این برنامه به این صورت است که زمانی که کاربر برنامه را اجرا می کند اول صفحه login را میبیند سپس با وارد کردن ID و password وارد می شود و لیست مخاطب ها را می بیند . سپس با کلیک کردن بر روی هر کدام از مخاطب ها وارد صفحه چت با آن مخاطب می شود و شروع به ارسال و دریافت پیام می کند. همچنین می تواند با زدن toolbar و با وارد کردن JID و Nickname مخاطب دیگری را به لیست مخاطب هایش اضافه کند.

در اوایل کار وضعیت user ها در سرور openfire به صورت زیر است . ما میخواهیم به عنوان کاربر farzad وارد سیستم شویم. همانطور که می بینید این کاربر در حالت offline قرار دارد.

Openfire 4.0.3
Logged in as admin - [Logout](#)

Server **Users/Groups** Sessions Group Chat Plugins

Users Groups

User Summary
Create New User
User Search
Advanced User Search

User Summary

Total Users: 5 -- Sorted by Username -- Users per page: 100

Online	Username	Name	Groups	Created	Last Logout	Edit	Delete
1	admin	Administrator	None	Dec 31, 1969			
2	farid		None	Oct 11, 2016	9 days, 9 hours, 31 minutes		
3	farzad		None	Oct 11, 2016	14 days, 23 hours, 53 minutes		
4	masih		None	Oct 11, 2016	1 minute		
5	user		None	Feb 21, 2017			

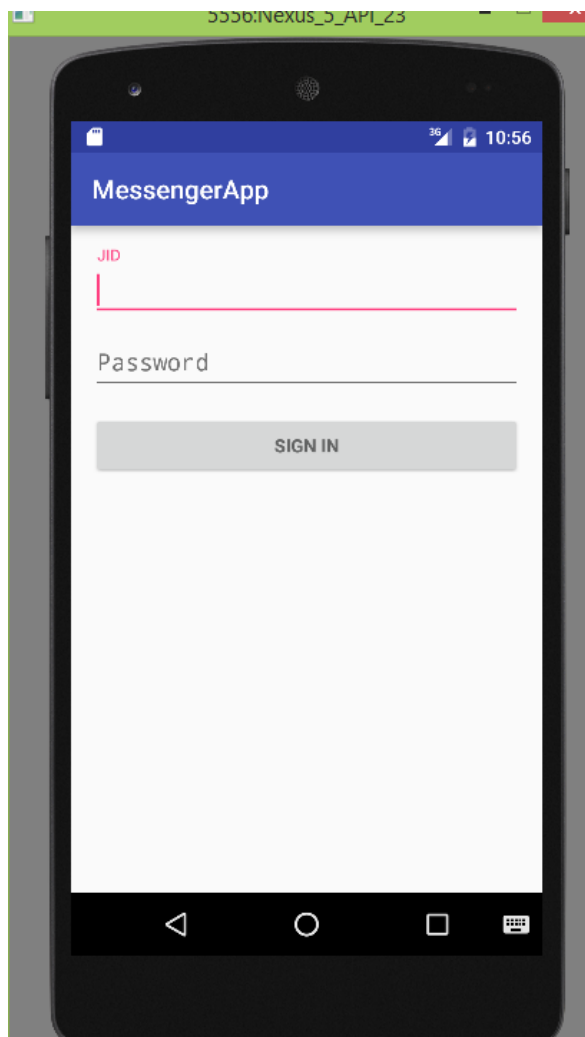
Server | [Users/Groups](#) | [Sessions](#) | [Group Chat](#) | [Plugins](#)

Built by [Jive Software](#) and the [IgniteRealtime.org](#) community

Activate Windows

شکل_۱۱ وضعیت اولیه کاربر ها

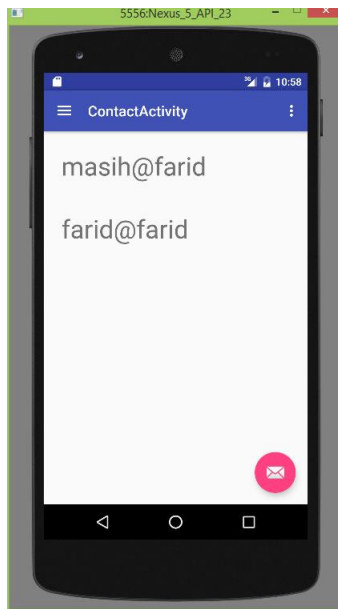
برنامه را اجرا می کنیم و صفحه ورود به پیام رسان بصورت زیر است



شکل_۱۲ صفحه ورود

بعد از وارد کردن ID و password اولین بار **loginActivity** اجرا می شود که در متد `onCreate()` خود یکی دیگر از متدهایش به نام `Login()` را فراخوانی می کند در `login()` هم این متد سعی می کند که **ConnectionService** را که یک سرویس است را فراخوانی می کند در این سرویس در متد `onStartCommand()` متد `Start()` را فراخوانی می کنیم و این متد دارای یک `handler` است و در داخل یک نخ دیگر کلاس `connection` را ایجاد می کند . یعنی ما یک سرویس داریم که **connection** ما در داخل آن قرار دارد در داخل `connection` که از `connectionListener` ارث بری شده است در متد `authenticated` آن متد `showContactlist()` را فراخوانی می کنیم که یک `intent` می سازد و `action` آن را `ConnectionService.UI_AUTENTICATED` قرار می دهیم و سپس با استفاده از **Broadcast** این `intent` را میفرستیم در **loginActivity** یک **broadcastReceiver** داریم `broadcast` فرستاده شده از سرویس را می گیرد و بنا به `action` تعریف شده آن لیست مخاطب ها را که در کلاس های مختلف متدهایش تعریف شده بصورت پویا از

سرور گرفته و وارد **RecyclerView** کلاس **contactActivity** می کند پس ما به صفحه لیست مخاطب ها هدایت می شویم.



شکل_۱۳ صفحه نمایش مخاطب ها

در شکل زیر وضعیت مخاطب ها بعد از login کردن در سرور **openfire** نمایش داده شده است. همانطور که می بینید کاربر **farzad** حالتش از **offline** به **online** تغییر کرده است

← → ↻ ⌂ ⓘ farid:9090/user-summary.jsp ☆ ⓘ ⋮

openfire Openfire 4.0.3
Logged in as admin - [Logout](#)

Server **Users/Groups** Sessions Group Chat Plugins

Users Groups

- User Summary
- Create New User
- User Search
- Advanced User Search

User Summary

Total Users: 5 -- Sorted by Username -- Users per page: 100 ▼

	Online	Username	Name	Groups	Created	Last Logout	Edit	Delete
1		admin	Administrator	None	Dec 31, 1969			
2		farid		None	Oct 11, 2016	9 days, 9 hours, 46 minutes		
3		farzad		None	Oct 11, 2016			
4		masih		None	Oct 11, 2016			
5		user		None	Feb 21, 2017			

Server | [Users/Groups](#) | [Sessions](#) | [Group Chat](#) | [Plugins](#)

Built by [Jive Software](#) and the [IgniteRealtime.org](#) community

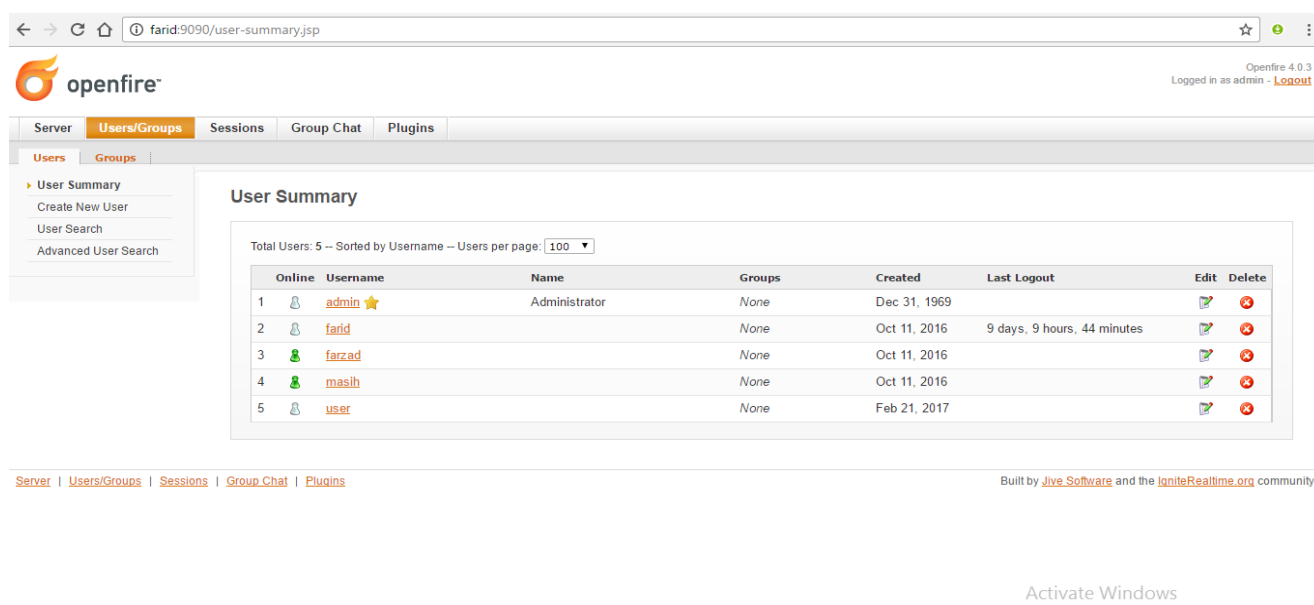
Activate Windows

شکل_۱۴ وضعیت کاربر ها بعد از login در سرور

در این صفحه ID ها بصورت username@server می باشند که سرور ما local بوده و اسم آن farid است.

با کلیک کردن بر هر کدام از مخاطب ها وارد chatActivity می شویم که در chatActivity با نوشتن پیام و زدن دکمه ، پیام ما به پردازش شده و سپس از طریق broadcast فرستاده میشود به کلاس connection و در آنجا ما یک broadcastReceiver داریم که بنابر action داده شده اقدام به ارسال پیام می کند. هنگام دریافت پیام هم همین روند به صورت معکوس انجام می شود.

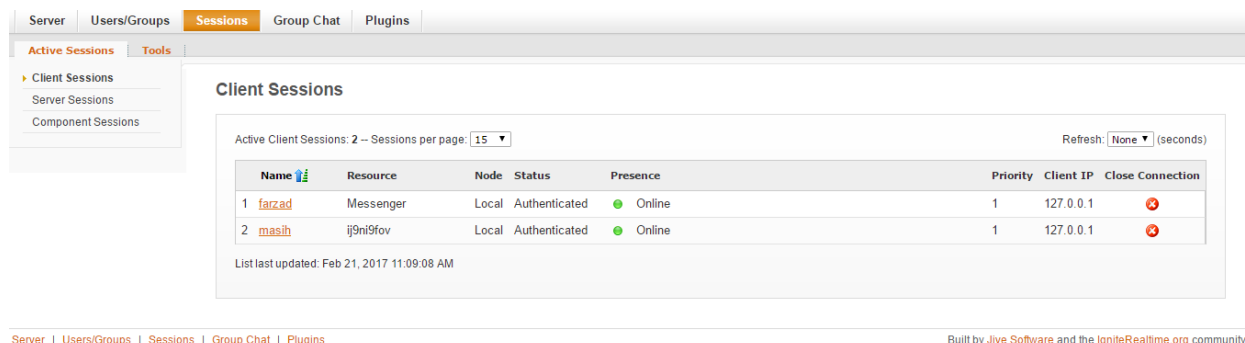
حال ما با استفاده از یک messenger دیگر به نام pidgin هم با کاربر masih وارد میشویم تا امکان چت بین دو کاربر online ایجاد بشه. وضعیت کاربر ها بعد از login کردن با pidgin در سرور به صورت زیر است.



The screenshot shows the Openfire 4.0.3 web interface. The top navigation bar includes 'Server', 'Users/Groups' (selected), 'Sessions', 'Group Chat', and 'Plugins'. The left sidebar has 'Users' and 'Groups' tabs, with 'Users' selected. Under 'Users', there are links for 'User Summary', 'Create New User', 'User Search', and 'Advanced User Search'. The main content area is titled 'User Summary' and displays a table of users. The table has columns for 'Online', 'Username', 'Name', 'Groups', 'Created', 'Last Logout', 'Edit', and 'Delete'. There are 5 users listed: 'admin' (Administrator), 'farid', 'farzad', 'masih', and 'user'. The bottom of the page shows the footer with 'Server | Users/Groups | Sessions | Group Chat | Plugins' and 'Built by Jive Software and the IgniteRealtime.org community'. An 'Activate Windows' watermark is visible at the bottom right.

Online	Username	Name	Groups	Created	Last Logout	Edit	Delete
1	admin	Administrator	None	Dec 31, 1969			
2	farid		None	Oct 11, 2016	9 days, 9 hours, 44 minutes		
3	farzad		None	Oct 11, 2016			
4	masih		None	Oct 11, 2016			
5	user		None	Feb 21, 2017			

شکل ۱۵_ وضعیت کاربر ها بعد از لاگین کردن با pidgin



The screenshot shows the Openfire 4.0.3 web interface with the 'Sessions' tab selected. The left sidebar has 'Active Sessions' and 'Tools' tabs, with 'Active Sessions' selected. Under 'Active Sessions', there are links for 'Client Sessions', 'Server Sessions', and 'Component Sessions'. The main content area is titled 'Client Sessions' and displays a table of active client sessions. The table has columns for 'Name', 'Resource', 'Node', 'Status', 'Presence', 'Priority', 'Client IP', and 'Close Connection'. There are 2 active client sessions: 'farzad' (Messenger, Local, Authenticated, Online) and 'masih' (ij9ni9fov, Local, Authenticated, Online). The bottom of the page shows the footer with 'Server | Users/Groups | Sessions | Group Chat | Plugins' and 'Built by Jive Software and the IgniteRealtime.org community'.

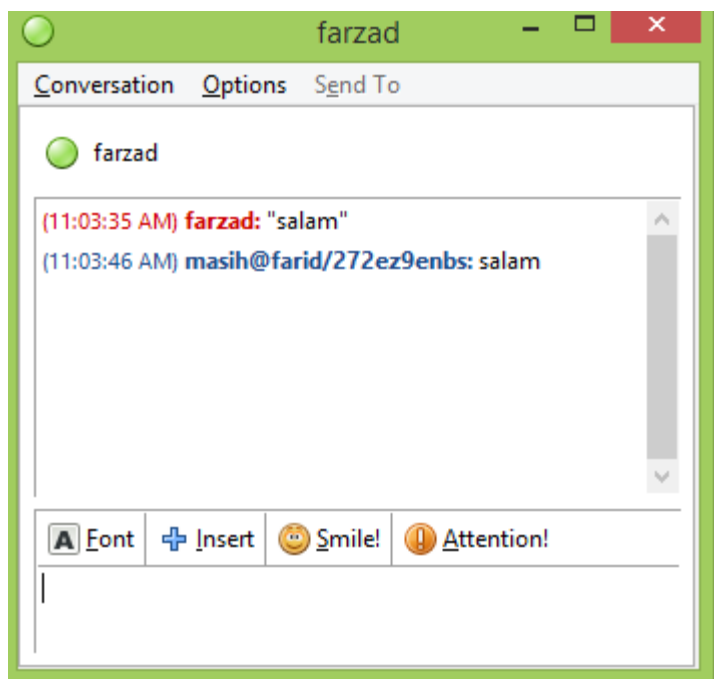
Name	Resource	Node	Status	Presence	Priority	Client IP	Close Connection
1 farzad	Messenger	Local	Authenticated	Online	1	127.0.0.1	
2 masih	ij9ni9fov	Local	Authenticated	Online	1	127.0.0.1	

شکل ۱۶_ وضعیت session کاربر ها که از دو resource متفاوت pidgin و پیام رسان طراحی شده login کرده اند

بعد از کلیک کردن بر روی مخاطب **masih** شروع به چت کردن با این کاربر می کنیم

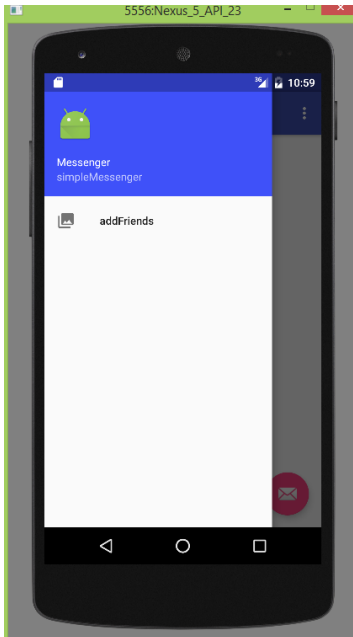


شکل_۱۷ چت سمت برنامه طراحی شده



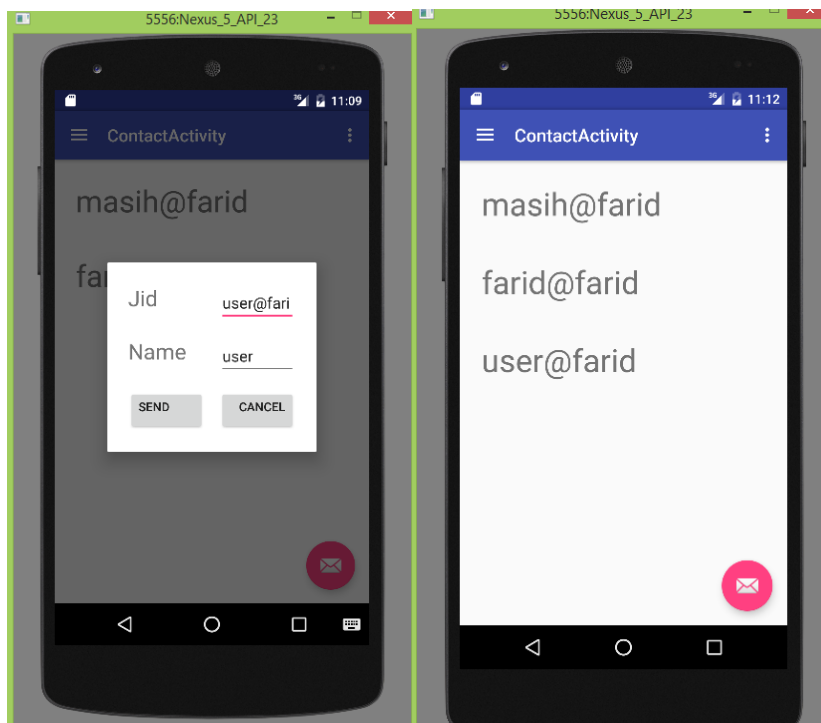
شکل_۱۸ چت سمت pidgin

حال می خواهیم که درخواست دوستی برای یک کاربر دیگر بفرستیم با کلیک بر toolbar داریم



شکل_۱۹

با کلیک بر addFriend یک dialog نمایش داده می شود و از آنجا مخاطب جدیدی را اضافه می کنیم



شکل_۲۰ dialog اضافه کردن کاربر و لیست مخاطبین بعد از آن

نتیجه گیری و پیشنهادات

نتیجه گیری

با پیشبرد برنامه پیام رسان، در فصل اول، ابتدا به شرح برنامه نویسی اندروید پرداختیم و ساختار سیستم عامل اندروید را مورد بررسی قرار داده و در مورد کتابخانه ها و لایه های مختلف آن پرداختیم سپس پروتکل xmpp را مورد بررسی قرار دادیم و جنبه های مختلف آن را شرح دادیم و همچنین در مورد ساختار این پروتکل صحبت کردیم .

در فصل دوم به بررسی انواع برنامه ها و ابزار های بکار گرفته شده در پروژه پرداختیم از جمله با محیط برنامه نویسی اندروید استادیو آشنا شدیم و بعضی ویژگی های آن را ذکر کرده و شرح دادیم و همچنین در مورد سرور openfire و ویژگی هایش صحبت کردیم نگاهی به نرم افزار پیام رسان pidgin داشتیم و با آن آشنا شدیم.

در فصل سوم با انواع تکنیک های برنامه نویسی اندروید آشنا شدیم و بصورت مفصل آنها را بررسی کردیم در ابتدا با کتابخانه smack که برای پیاده سازی پروتکل سمت کاربر xmpp مورد استفاده قرار می گیرد آشنا شدیم. به بررسی انواع broadcast ها پرداختیم و همچنین در مورد handler ها و service ها بصورت مفصل بحث کردیم. در انتها به نحوه بکار گیری این تکنیک ها و نرم افزار ها در پروژه مان پرداختیم و برنامه طراحی شده را به نمایش گذاشتیم.

پیشنهادهات

در ادامه مجموعه ای از ویژگی هایی که می توان به این پروژه اضافه کرد را لیست کرده ایم

۱. امکان ایجاد چت گروهی
۲. امکان block کردن کاربران
۳. امکان فراهم کردن جستجو با roster
۴. امکان ایجاد چت صوتی
۵. استفاده از plugin برای سرور به منظور نگه داشتن پیام های سابق
۶. امکان تغییر حالت از online به busy و away و

- David Koelle, Lloyd Watkin: Practical Protocols: XMPP
- <http://stackoverflow.com>
- <https://developer.android.com>
- <http://wikipedia.org>
- <https://www.igniterealtime.org>
- <https://github.com>