بسم الله الرحمن الرحیم

طراحی پیام رسان (Zprava)

کیان بهزاد ـ پارسا اسدی ـ امیر بیات

۲۶ تیر ۱۳۹۷

در این گزارش هدف این است که گزارشی در مورد بخش های مختلف این پروژه ارائه شود. ابتدا بخش Client توضیح داده می شود سپس بخش Server و در انتها امکاناتی که در برنامه قرار دادیم را به زبان ساده و به دور از جزییات برنامه نویسی بیان میکنیم. بنابراین اگر قصد دارید تنها از امکانات برنامه آگاه شوید بخش سوم را مطالعه فرمایید

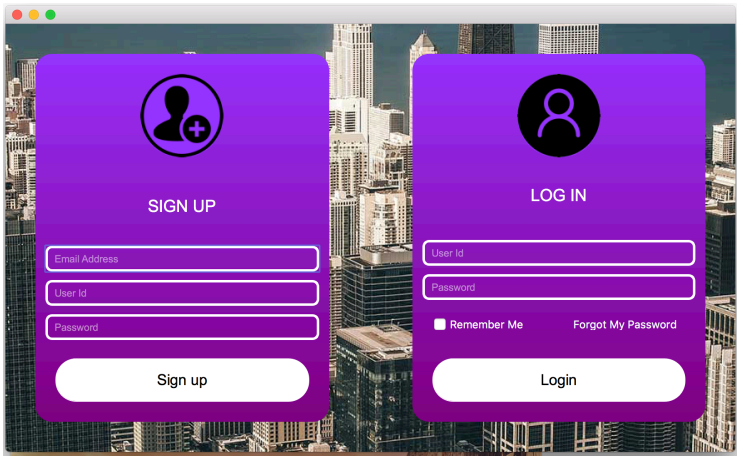
۱ بخش کاربری (Client)

۱.۱ مقدمه

برنامه طرف کاربری توسط Qt و به زبان C++ توسعه داده شده است. بخش های اصلی این قسمت عبارت اند از کلاس هایی که در ادامه توضیح داده می شوند.

۲.۱ کلاس Zprava

وظیفه این کلاس به طور کلی اجرای برنامه است یعنی تمام کلاس های دیگر که نوشته شده اند را نمایش می دهد. همچنین متغیر WHOAMI که از نوع ZpUser است(در ادامه توضیح داده می شود)و نقش محوری در برنامه دارد نیز ساخته می شود. در این کلاس تمام Thread هایی که در برنامه قرار است استفاده شوند نیز ساخته می شوند. طول و عرض پنجره برنامه که بعد از اجرای کد نمایش داده می شود نیز در این کلاس تنظیم شده است و این کلاس دارای فایل StyleSheet جداگانه در پوشه Resources است که ظاهر کلاس را زیباتر می کند.



شکل ۱: کلاس ZpForm

۳.۱ کلاس ZpForm

به محض اجرای برنامه این کلاس اجرا می شود که در حقیقت فرم Signup و Login است.(اگر قبلا ثبت نام کرده باشید باید Login می کنید.) کانستراکتور این کلاس شامل چهار تابع اصلی است:

- apply_stylesheet()

این تابع فایل stylesheet.qss را از پوشه Resources فراخوانی می کند و به برنامه می شناساند. این فایل به نوعی معادل فایل CSS در Html است و کد های مربوط به Styling کلاس در آن قرار می گیرد.

- create_form_widget()

این تابع وظیفه ساخت گرافیک کلاس ZpForm را بر عهده دارد. به این شکل که هر widget را در یک layout قرار می دهد. سپس این layout ها را به QWidget تبدیل می کند. در نهایت تمام این widget های مرکب را در یک layout کلی می گذارد و در انتها ZpForm را با layout نهایی setlayout می کند.

- initiate_networking()

این تابع ارتباط ZpForm با سرور را برقرار می کند. به این شکل که ابتدا URL را که قرار است با آن Request بدهد را تعریف می کند سپس با توجه به مطالبی که کاربر در Field ها نوشته است و دکمه ای که زده است. سیگنالی ساطع می شود و به یک Slot وصل می شود و Request به درستی

به سرور فرستاده می شود. جواب سرور نیز در تابعی دیگر دریافت می شود و به Slot دیگر به نام handle_replay می رود تا در آن جا با توجه به جواب سرور تصمیم درست در client اتخاذ شود.

- is_kept_logged_in()

این تابع مسئول آن است که اگر کاربر در حین Login گزینه Remember me را انتخاب کرد از دفعه بعد مستقیم وارد صفحه چت شود و دیگر نیاز به اطلاعات کاربر نباشد(مانند پیامرسان Telegram) نحوه انجام این کار به این صورت است که اگر کاربر مایل به این کار باشد این تابع یک فایل text در پوشه پروژه می سازد و دفعه بعد که کاربر وارد برنامه شد این تابع چک می کند که آیا این تابع وجود دارد یا خیر و اگر وجود داشت به سرور reuqest می دهد که کاربر را Login در نظر بگیرد و صفحه چت نمایان می شود.

می توانید نمای این کلاس را در شکل ۱ مشاهده کنید.

۴.۱ کلاس ZpChatType

شکل ۲: کلاس ZpChatType

این کلاس در حقیقت همان محیطی است که که کاربر پیام خود را در آن می نویسد و می فرستد. قابلیت ویژه ای که این widget دارد این است که responsive است . یعنی با توجه به تعداد خطوط پیام نوشته شده اندازه اش تنظیم می شود(مانند تلگرام و چه بسا بهتر) این کار به این شکل انجام شده است که بعد از تایپ هر character سیگنالی ساطع می شود و به یک Slot وصل می شود. و در آن جا تعداد خطوط در هر لحظه چک می شود و تصمیم مناسب برای تغییر اندازه آن اتخاذ می شود.

ویژگی دیگر این کلاس این است که از Qpushbutton برای ارسال پیام در آن استفاده نشده است زیرا نمیتوان در آن عکس گذاشت. اما در QLabel می توان عکس گذاشت. پس کلاسی نوشته شد که از QLabel ارث ببرد و امکان کلیک کردن روی آن قرار داده شد و به هدفمان که همان طراحی Label با ویژگی های button بود رسیدیم.

قابلیت دیگر آن این است که اگر اینترنت موجود نباشد. تشخیص داده می شود و با زدن دکمه ارسال علامت قرمز رنگی ظاهر می شود.

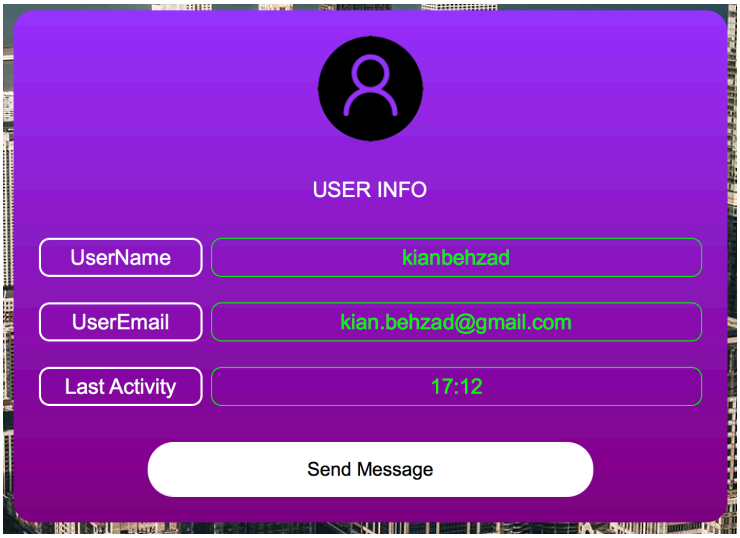
شکل ۳: کلاس ZpNavigationBar

۵.۱ کلاس ZpNavigationBar

این کلاس همان نوار بالای صفحه چت است که در اکثر پیامرسان ها وجود دارد و از طریق آن میتوان کاربران دیگر پیامرسان را جست و جو کرد و با آن ها چتی را شروع کرد. نحوه کار به این صورت است که با زدن هر character در Field جست و جو سیگنالی ساطع می شود. این سیگنال به Slot مناسب وصل می شود و کلمه تایپ شده به سرور فرستاده می شود و در آن جا با Regular Expression تمام کاربرانی که username آن ها با کلمه فرستاده شده match است به کلاینت باز گردانده می شوند. و به صورت suggestion در هنگام تایپ نمایش داده می شوند. همچنین در این widget دو دکمه setting و Start chat وجود دارد که کلاس های دیگر مانند ZpSetting و ZpUserInfo را نمایش می دهند که در ادامه این کلاس ها توضیح داده می شوند.

شکل ۴: Autocomplete

۶.۱ کلاس ZpUserInfo



شکل ۵: کلاس ZpUserInfo

این کلاس از طریق کلاس ZpNavigationBar و از طریق زدن دکمه Start Chat اجرا می شود. در آن مشخصات کاربر جست و جو شده نمایش داده می شود و می توان با او شروع به گفت و گو کرد. نحوه به دست آوردن این اطلاعات به این صورت است که وقتی کاربر username را جست و جو می کند. این کلمه به سرور می رود و اطلاعات کاربری که username آن دریافت شده است را(در صورت وجود) به Client باز میگرداند تا در ZpUserInfo نمایش داده شود.

۷.۱ کلاس ZpSetting

این widget برای تغییر رمز عبور و username برای کاربر طراحی شده است. و از ZpNavigationBar قابل دسترسی است. نحوه کارش نیز به اینگونه است که گذرواژه و نام کاربری جدید به سرور فرستاده می شود و اگر قبلا توسط کاربر دیگر گرفته نشده باشد به کاربر تخصیص می یابد.

۸.۱ کلاس ZpUser

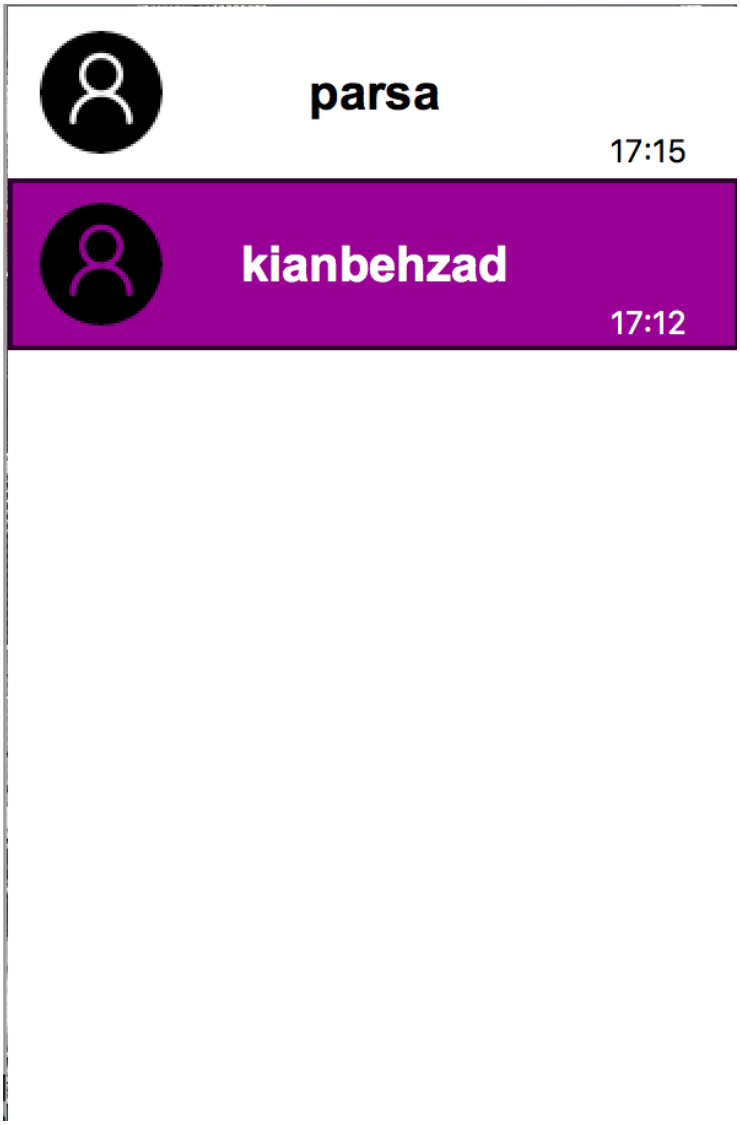
اشیا این کلاس نمایانگر کاربران هستند. در کانستراکتور این کلاس یک رشته ورودی وجود دارد که همان نام کاربری است. هر گاه شئ با Username خاص از این کلاس ساخته شود. نام کاربری کاربر ساخته شده به سرور می رود و در آن جا چک می شود که آیا واقعا چنین کاربری در سرور وجود دارد یا خیر. گفتنی است رمز عبور کاربران به دلایل امنیتی بهتر است در کلاس ZpUser وجود نداشته باشد و فقط در سرور باشد و ما هم این کار را کردیم.

۹.۱ کلاس ZpContact

می دانیم تمام پیام رسان های امروزی دارای Contactlist هستند. پیام رسان Zprava نیز از این قائده مستثنی نیست. هر Zpuser به صورت یک شئ از کلاس ZpContact در می آید و آماده نمایش در ZpContactList (که در ادامه توضیح داده می شود) می شود. کانستراکتور این کلاس یک رشته می گیرد که Username همان کاربری است که قرار است در ContactList نمایش داده شود. این کلاس شامل چهار متغیر کلیدی زیر است:

- picture: همان عکس کاربر است که نمایش داده می شود.
- title: همان نام کاربری است که نمایش داده می شود
- notification: ممکن است کاربر mute شده باشد. از این متغیر برای پیگیری مسائل مربوط به این قضیه استفاده می شود.

- datetime: از این متغیر برای تشخیص ترتیب ZpContact ها در ZpContactList استفاده می شود. زیرا باید همواره به ترتیب خاصی در Contactlist حاضر باشند.



شکل ۶: کلاس ZpContactList

۱۰.۱ کلاس ZpContactList

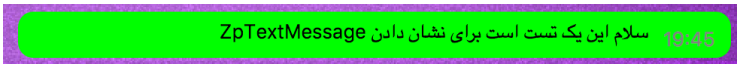
این کلاس در حقیقت همان Contactlist در پیام رسان است. در این کلاس لیستی از ZpContact ها که در مورد آن ها توضیح داده شد وجود دارد. این کلاس از کلاس ZpThread (که در مورد آن توضیح داده خواهد شد) ارث می برد. چون باید در فواصل زمانی نسبتا کوتاه به سرور request دهد و خود را update کند.(Notification پیام های جدید، ترتیب ZpContact ها و ...) بنابراین بهتر است در Thread جداگانه باشد.

۱۱.۱ کلاس ZpMessage

این کلاس در حقیقت همان پیام هایی است که در صفحه چت نمایش داده می شود و دارای متغیر های مهم زیر است:

- opponent:این متغیر از نوع ZpUser است و آن کاربری است که این ZpMessage به آن فرستاده شده است.
- origin_publisher: همان کاربری است که این پیام را فرستاده است. دلیل این نامگذاری این است که باید فرستنده حقیقی هر پیام در پیامرسان مشخص باشد تا می تواستیم ویژگی Message Forwarding را به آن اضافه کنیم.
- datetime:این متغیر زمان ارسال پیام است. و باید مشخص باشد تا پیام های یک چت به صورت درستی مرتب شوند(بدیهی است که پیام های جدید تر پایین تر می آید)
- pk:هر پیامی چه در Client و چه در سرور دارای یک pk است. این متغیر یک عدد است و هر پیامی pk منحصر به فرد خود را دارد. مانند اثرانگشت آن پیام است. این کلاس در حقیقت برای این طراحی شد که parent کلاس های دیگر باشد که هر کدام نماینده یک نوع خاص از پیام است(پیام تصویری- پیام صوتی و ...)

۱۲.۱ کلاس ZpTextMessage

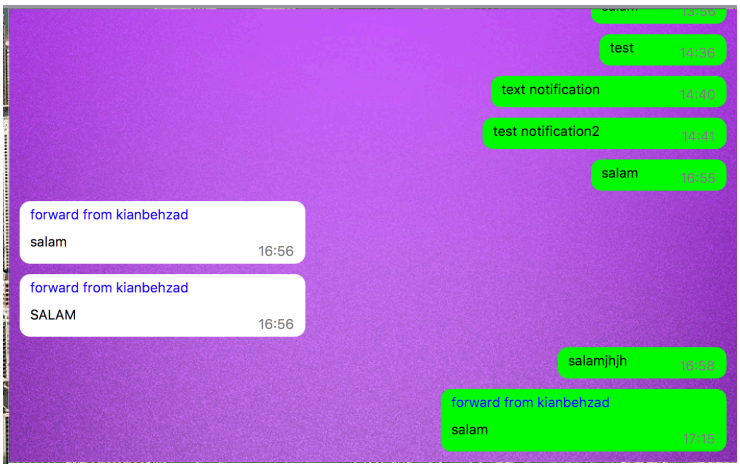


شکل ۷: کلاس ZpTextMessage

این کلاس از کلاس ZpMessage ارث می برد. متغیر های کلیدی اش همان هایی است که به ارث می برد و چند تابع اضافه دارد که برای کمک به نمایش دادن پیام در صفحه چت است.(مثلا این که طول widget

با توجه به طول بلند ترین سطر پیام مشخص شود و ...) یک تابع که در کلاس وجود دارد و نقش مهمی دارد تابعی است که پیام را تکه تکه می کند. متاسفانه تابعی که نوشته درون یک QLineEdit را می گیرد آن را به صورت یک رشته تک سطری بر می گرداند و اگر قرار بود برای نمایش پیام از خروجی این تابع مستقیما استفاده می کردیم با مشکل جدی مواجه می شدیم چون کل عرض صفحه چت پر می شد. بنابراین مجبور شدیم برای یک سطر طول ماکزیمم مشخص کنیم و پیام کاربر را با تابعی که نوشتیم خودمان قطعه قطعه کنیم.

۱۳.۱ کلاس ZpChatView



شکل ۸: کلاس ZpChatView

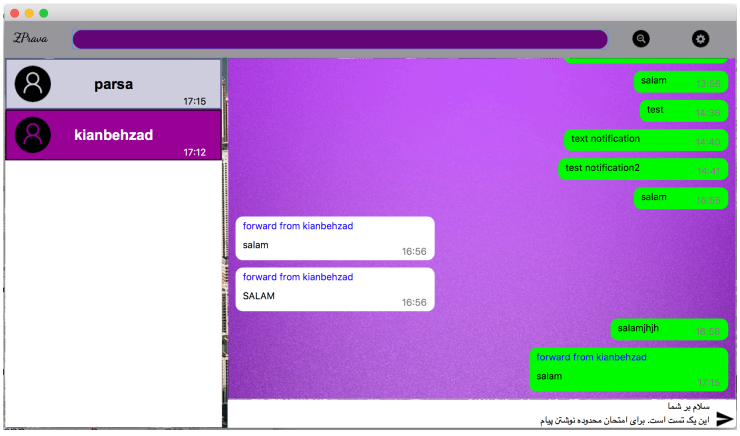
این کلاس همان محیطی است که پیام های ارسالی در آن به ترتیب تاریخ نمایش داده می شوند(شکل ۸).کانستراکتور آن متغیری از نوع ZpUser می گیرد که همان طرف دوم چت است. از جمله توابع مهم این کلاس تابع sort است که بعد از هر تغییری در ZpChatView اجرا می شود و تمام ZpTextMessage ها را بر اساس تاریخ آن ها مرتب می کند.این تابع با استفاده از توابع STL نوشته شده است.

۱۴.۱ کلاس ZpChatWindow

این کلاسی است که تمامی widget های که شرح داده شد را در بر میگیرد و نشان می دهد.

۱۵.۱ کلاس ZpThread

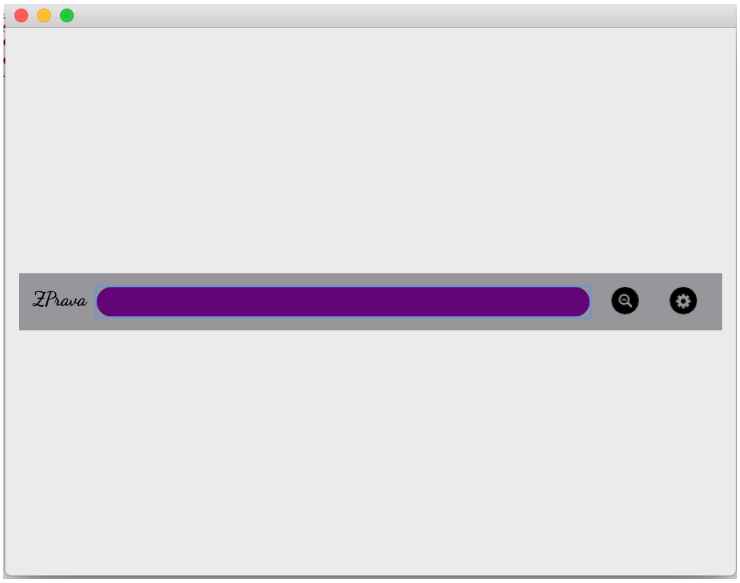
موازی با برنامه اصلی (بخش گرافیک) هر نیم ثانیه یک بار به سرور درخواست گرفتن تمام چت های کاربر را میفرستد. به این صورت که فایل json که حاوی اطلاعات چت ها می باشد را به روز رسانی میکند.



شکل ۹: کلاس ZpChatWindow

۱۶.۱ کلاس ZpExperimental

در طول توسعه برنامه همواره این نیاز حس می شد که widget هایی که میسازیم را کجا (خارج از برنامه اصلی) آزمایش کنیم که درست کار کند برای همین این کلاس ایجاد شد.



شکل ۱۰: کلاس ZpExperimental

۲ بخش سرور(Server)

۱.۲ مقدمه

بخش سرور با Django توسعه یافته است. تا یک هفته مانده به تحویل پروژه تمام تست های بخش کاربری روی local host انجام میگرفت. پس از کامل شدن کد های سرور و Client و تمام شدن پروژه سرور را خریدیم و برنامه را روی آن بردیم. سرور همانند یک کامپیوتر مستقل است و ما در ابتدا مانند تمام کامپیوتر های دیگر باید موارد مورد نیاز مانند Django و Python را در آن نصب میکردیم. این کار را انجام دادیم پس از آماده شدن سرور کدهای Django را در آن قرار دادیم و کار کرد. بخش کدهای سرور به پیچیدگی و سختی بخش Client نبود و ساده تر بود. ساختار کلی سرور به این شکل است که دارای قسمت های مختلفی است که به آن ها App می گویند. این App ها شامل توابع مختلفی هستند که Client در حقیقت به آن ها Request می دهد و مقدار بازگشتی این توابع همان Replay است که به Client باز میگردد(این توابع در فایل views.py تعریف می شوند). همچنین هر Request در قالب یک URL به App ها می آید و در ابتدا باید این URL ها را تعریف کنیم (set url).این کار در فایل urls.py انجام می شود.

موضوع مهم دیگری که وجود دارد این است که Database شامل تعدادی Object از کلاس های مختلف مانند: Users یا TextMessage و ... هستند به این کلاس ها در Django مدل گفته می شود. و برای هر App در فایل models.py مربوط به آن نوشته می شود.در حقیقت model ها واحد های DataBase هستند. علاوه بر کد می توان به طور مستقیم نیز در سرور از کلاس هایی که داریم object بسازیم(شکل ۱۱) در ادامه به بررسی App ها می پردازیم:

۲.۲ Chat

این App شامل دو model به نام های Chat و TextMessage است. کلاس Chat شامل دو متغیر first_size و second_side است که نماینده طرف اول و طرف دوم چت است. کلاس TextMessage شامل متغیر های publisher که نشان دهنده فرستنده پیام است و subscriber که نشان دهنده ی دریافت کننده ی پیام است و datetime که نشان دهنده ی زمان صدور پیام است و is_seen که نشان دهنده ی این است که این پیام توسط گیرنده دیده شده است یا خیر و text که نشان دهنده ی متن پیام است. نمونه زیر کد یک مدل است:

```
class TextMessage(models.Model):
    publisher = models.ForeignKey(Users,
     related_name="published_text_messages",
      null=True, on_delete=models.CASCADE)
    subscriber = models.ForeignKey(Users,
     related_name="subscribed_text_messages",
      null=True, on_delete=models.CASCADE)
    chat = models.ForeignKey(Chat,
```

```
 related_name="chat_text_messages",
  null=True, on_delete=models.CASCADE)
is_seen = models.BooleanField(default=False)
datetime = models.CharField(max_length=50, null=True)
text = models.CharField(max_length=1000)
type = models.CharField(max_length=50, null=True)#TEXT-PHOTO-AUDIO-VIDEO
def __str__(self):
    return self.publisher.__str__()
    + ' - ' + self.subscriber.__str__()
    + ' : ' + self.text[0:20] + '...'
```

برای مثال در کد بالا Charfield به معنی این است که در دیتابیس این متغیر به نوعی string است و ماکزیمم اندازه آن نیز مشخص شده است. serializer نیز تعریف شده است که کار آن تبدیل json است. و برای پاسخ به client بعضی توابع که باید به صورت json باشد کاربرد دارد.

حال به بررسی توابع این App می پردازیم.

- seen() : برنامه زمانی که پیام توسط گیرنده دیده شد به این تابع request می دهد تا متغیر بولی is_seen از مدل TextMessage که بالا تر توضیح داده شد را True کند. request شامل نام کاربری فرستنده و نام کاربری گیرنده است. این تابع به طور خودکار با همین دو ورودی سرور را جست و جو می کند تا چت های مربوط به آن دو را بیابد و متغیر is_seen آن ها را عوض کند.
- getmessage() : به این تابع request شامل فرستنده و گیرنده یک پیام و همچنین Pk آن پیام داده می شود. تابع سرور را با توجه به این ورودی ها می گردد و پیام را پیدا می کند. و آن را خروجی می دهد.(در مورد این که Pk چیست در بخش Client در بالا توضیح داده شد.)
- chatheaders() : این تابع یک request شامل یک username را می گیرد و Pk تمام پیام هایی که این کاربر تا به حال داده است را بر می گرداند. در ابتدا این تابع طوری بود که کل chat ها را بر می گرداند و این بهینه نبود برای همین آن را عوض کردیم.

۳.۲ Navigation

این App مدلی ندارد و فقط توابعی دارد:

- userdata() : این تابع یک request شامل یک username می گیرد و سرور را می گردد تا این کاربر را پیدا کند. وقتی پیدا شد object یافت شده که از نوع user است را با serializer به json تبدیل می کند و خروجی می دهد. این تابع وقتی یک ZpUser در طرف کاربری ایجاد می شود اجرا می شود.
- search() : این تابع یک request شامل یک string می گیرد. این رشته همان رشته ای است که کاربر در ZpNavigationBar آن را در بخش جست و جو وارد کرده است. این تابع با استفاده از ماژول re

یک pattern از این رشته ورودی می سازد و تمام username هایی که با رشته ورودی match می شوند را به صورت لیست خروجی می دهد. این لیست همان لیست suggestion هایی است که کاربر در حین search در بخش کاربری می بیند.

Login ۴.۲

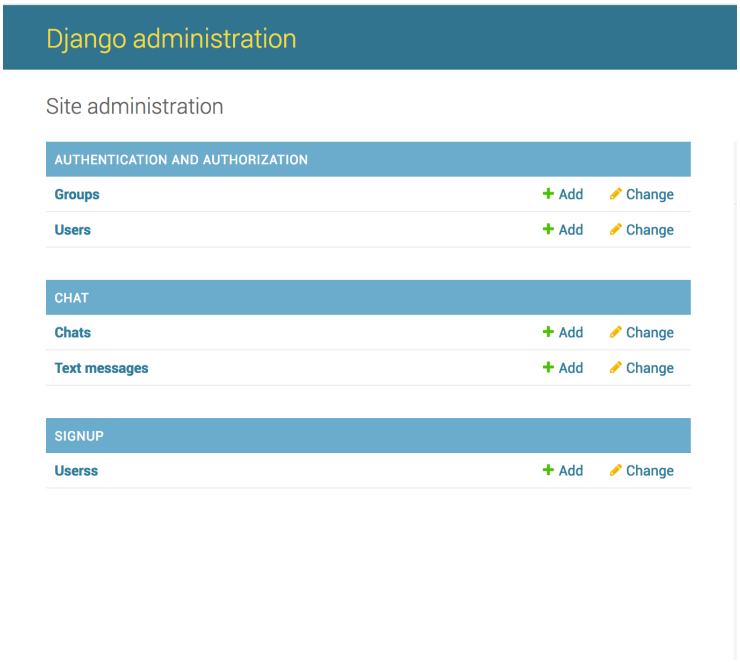
این App نیز مدلی ندارد و شامل توابع زیر است:

- login() : به این تابع یک request از طرف ZpForm شامل نام کاربری و گذر واژه ای است که کاربر میخواهد با آن login کند. این تابع چک می کند که نام کاربری و گذرواژه وارد شده با هم مطابقت دارد یا خیر و خروجی متناسب را می دهد.
- forget() : در سمت کاربری اگر کاربر گذرواژه خود را فراموش کرده باشد یک request شامل Email خود به این تابع می دهد. این تابع سرور را جست و جو می کند تا با توجه به ایمیل فرستاده شده کاربر و مشخصاتش را پیدا کند. سپس یک ایمیل به کاربر می فرستد که شامل گذرواژه اوست و مشکل حل می شود

signup ۵.۲

: شامل توابع زیر است:

- registration : از سمت کاربری و از سمت ZpForm به این تابع request داده می شود. این request شامل نام کاربری و گذرواژه و ایمیلی است که کاربر با آن ثبت نام می کند. در این تابع چک می شود که username مشابه در سرور موجود است یا خیر و اگر موجود نبود و نام کاربری و ایمیل معتبر بود کاربر در سرور ثبت می شود. سپس عدد رندومی تولید می شود و به عنوان verification code به ایمیل کاربر فرستاده می شود.
- verification_code : از کاربر بعد از ثبت نام خواسته می شود که verification code را وارد کند. با این کار Client یک request شامل کدی کاربر نوشته است را می دهد. حال در این تابع چک می شود که این کد درست است یا خیر و اگر درست بود کاربر اصطلاحا verified می شود.
- setting : وقتی کاربر بخواهد نام کاربری یا گذرواژه خود را تغییر دهد از کلاس ZpSetting استفاده می کند.از این کلاس یک request به این تابع می آید که شامل نام کاربری و گذرواژه جدید است. در این تابع چک می شود که نام کاربری که کاربر می خواهد قبلا توسط کاربر دیگری اخذ نشده باشد. در این صورت عملیات تغییر مشخصات کاربر را در سرور انجام می دهد.



شکل ۱۱: سرور Django

۳ جمع بندی امکانات برنامه

- امکان ایجاد حساب کاربری بر مبنای ایمیل
- تشخیص اصالت ادرس های ایمیل با سیستم ارسال کد تاییدیه به ایمیل کاربر
- امکان Login برای کاربر هایی که از پیش ثبت نام کرده اند.
- وجود امکان بازیابی گذرواژه در صورت فراموشی آن با ارسال اطلاعات به ایمیل کاربر
- وجود گزینه remember_me که با فعال کردن این گزینه کاربر در دفعات بعد که برنامه را اجرا می کند نیازی به Login ندارد و خود به خود وارد صفحه چت می شود.(مانند تلگرام)
- امکان فرستادن پیام
- محیط گرافیکی زیبا و بدون کندی
- امکان
- امکان delete کردن پیام. forward کردن پیام.

- نمایش notification وقتی پیام جدیدی دریافت می شود.
- تشخیص پیام seen و unseen و نمایش notification متناسب با آن(مانند تلگرام)
- امکان Chat With Yourself که به نوعی یک فضای ابری در اختیار هر کاربر قرار می دهد که پیام هایش را در آن جا ذخیره کند.(مانند تلگرام)
- تشخیص وجود یا عدم وجود اینترنت و اطلاع رسانی آن به کاربر. حتی اگر در حین چت اینترنت قطع شد و بعد از مدتی دوباره وصل شد برنامه به صورت اتومات خود را به اینترنت وصل میکند و به کارش ادامه می دهد.
- در حین تایپ پیام اندازه Field که پیام در آن تایپ می شود تابعی از تعداد خطوط پیام است و با افزایش خطوط بزرگتر و با کاهش خطوط کوچک تر می شود بنابراین کاربر به راحتی می تواند پیام با هر تعداد خطوط را بفرستد(الگو برداری از تلگرام)
- امکان جست و جو سایر کاربران از طریق نام کاربری آن ها بوسیله navigationbar تعبیه شده. به این صورت که در تمام طول مدتی که کاربر در حال تایپ کردن در Field جست و جو است. در هر لحظه و با توجه به چیزی که کاربر تا به آن لحظه نوشته است suggestion هایی نمایش داده می شود(تلگرام فاقد این قابلیت است)
- امکان مشاهده زمان اخرین فعالیت شروع چت با کاربری که در بند قبل یافت شده است.
- امکان تغییر نام کاربری و گذرواژه
- امکان logout به طوری که از آن به بعد حتی اگر کاربر remember me را نیز فعال کرده باشد در مراجعت بعدی حتما باید Loginذ کند.
- امکان mute notification کردن هر کاربر دلخواه.
- برنامه crossplatform است و روی MacOS و Ubuntu و Windows10 تست شده است.

تمام قابلیت های بالا در فیلم های ضمیمه تست شده است. همچنین تمام قابلیت های پایه ای یک پیام رسان را دارد که ممکن است در بالا ذکر نشده باشد.