236700-HW1 - DRY PART

The projects manages the dependency graph of the classes that have the @Inject and @Provides annotation.

Dagger cannot instantiate or inject classes that do not have neither @Inject nor @Provides annotations.

modules are being injected with @ContributesAndroidInjector to "providers modules"

these "provider modules" are being injected ActivityBuilder module.

ActivityBuilder is used here:

```
@Singleton
@Component(
    modules = [
        AndroidSupportInjectionModule::class,
        AppModule::class,
        ActivityBuilder::class,
        ServiceBuilder::class,
        ReceiverBuilder::class,
        AndroidWorkerInjectionModule::class]
)
interface AppComponent {

    @Component.Builder
    interface Builder {
        @BindsInstance
        fun application(application: Application): Builder

        fun build(): AppComponent
    }

    fun inject(app: RocketChatApplication)

    fun inject(service: MessageService)
}
```

to build component.

RocketChatClient:

RocketChatClient is injected and Configured using this :

```
@Provides
@PerFragment
fun provideRocketChatClient(
    factory: RocketChatClientFactory,
    @Named("currentServer") currentServer: String?
): RocketChatClient {
    return currentServer?.let { factory.get(it) }!!
}
```

in ChatRoomFragmentModule

```
@Module
abstract class ChatRoomFragmentProvider {

    @ContributesAndroidInjector(modules = [ChatRoomFragmentModule::class])
    @PerFragment
    abstract fun provideChatRoomFragment(): ChatRoomFragment
}
```

```
@PerActivity
@ContributesAndroidInjector(
    modules = [MainModule::class,
        ChatRoomsFragmentProvider::class,
        ServersBottomSheetFragmentProvider::class,
        SortingAndGroupingBottomSheetFragmentProvider::class,
        CreateChannelProvider::class,
        ProfileFragmentProvider::class,
        SettingsFragmentProvider::class,
        AdminPanelWebViewFragmentProvider::class,
        DirectoryFragmentProvider::class
    ]
)
```

abstract fun bindChatRoomActivity(): ChatRoomActivity

CreateChannelView is injected and Configured using this ::

CreateChannelView is Injected to CreateChannelPresenter with a constructor.

CreateChannelView is Configured using this function in CreateChannelModule:

```
@Provides
@PerFragment
fun createChannelView(fragment: CreateChannelFragment): CreateChannelView {
    return fragment
}
```

which is injected in CreateChannelProvider which is injected in ActivityBuilder

```
@Module
abstract class CreateChannelProvider {

    @ContributesAndroidInjector(modules = [CreateChannelModule::class])
    @PerFragment
    abstract fun provideCreateChannelFragment(): CreateChannelFragment
}
```

if we had an alternative implementation that we wanted to inject, the changes we would need to make to the DAGGER configuration are:

we need to change the modules that uses the previous implementation.