iOS 升级优化记录

一、总览

本次升级优化的重点分为两个方面：

1、优化聊天模块

(1) 重构聊天界面

(2) 优化聊天消息收发存储逻辑

(3) 添加新功能

2、 优化部分性能问题

二、详解

1、优化聊天模块

在当今网络发达信息大爆炸的时代，手机几乎是人手一部，而手机中必不可少的 APP 就是社交类 APP。所以聊天功能在一个软件项目中显得尤为重要，这次我们重点优化了项目中的聊天功能，主要内容是：

（1）重构聊天界面

① 重构聊天 ViewController：聊天 ViewController 也就是用户与好友聊天的主界面，使用频率高。原有的单聊与群聊界面是完全独立的，不利于修改。现将共有功能抽离出来，供不同页面继承，减少冗余代码量，方便维护。在 MVC 设计模式中，ViewController 充当一个 CPU 的功能，负责处理 View 和 Model 的事件。优化后聊天控制器更好提现了它的责任。

② 重构聊天 View：按照开发中一个很重要的原则——降低耦合性。低耦合体现的就是程序设计中的模块化设计，将两个元素之间的联系、影响降低到最小。我们将消息键盘进行封装，实现工厂模式。将消息 Cell 重新布局，实现按类创建，使得代码逻辑结构更加清晰，并且利于后期拓展，同时也带给用户更好的操作体验。

③ 优化聊天 Model：在消息数据处理和 UI 呈现上，如果计算数据的同时还要处理 UI 的话，会有很多弊端，比如滑动消息列表会卡顿。对此，我们按照 MVC 的设计模式，在消息模型里面添加计算 Cell 高度的方法，当每次 Cell 需要高度以及内部布局的时候就可以直接调用，不需要进行重复计算。

（2）优化聊天消息收发存储逻辑

① 发送接收聊天消息：用户只需轻轻点击消息“发送”按钮，对方便可收到这条消息。看似简单的操作，但其中的罗逻辑却非常复杂。这其中需要先将消息按照不同类型封装成 socket 可传递的对象，待对方收到消息后，再将其转成 APP 可显示的消息体。原先在的代码这一系列的操作处理上，逻辑上存在不合理的现象，逻辑复杂程度高。优化过后，按照不同类型将消息封装成对象，逻辑结构更加清晰明了。

② 存储聊天消息：消息在数据库中的读取操作，可谓是重中之重。在聊天界面从数据库获取消息时，也存在在逻辑上过于复杂的问题。除此之外，sessionList 的更新逻辑也有不合理的情况。针对以上情况，对相应代码进行了调整，减少多余的查询修改次数，合并功能相似的查询修改方法。同时也新增加了部分新的查询方法，提高效率。

（3）添加新功能

① 文字分解：连续撞击文字消息，可按照单个个体显示，用户可自由选择某一部分文字或者多选，再出复制转发翻译搜索（弹窗形式）。

② 文字放大：用户在编辑好要发送的文字后，长按住发送键就会出现上划提示，向上划可以随机改变文字的大小，选择好之后再点击发送就可以发送出去，聊天界面可显示设置完的文字。

③ 文字颜色：用户在编辑好要发送的文字后，长按住发送键就会出现左划提示，左划则可以选择文字的颜色。选择好之后可以发送出去，聊天界面可显示设置完的文字。

④ 添加语音播放进度条：在播放语音消息时，用户可随意推动进度条，调整播放语音进度。

⑤ 快速切换聊天对象：当用户在聊天界面，只要点击特定的按钮，就能滑出左侧的头像栏目，用户在不需要退出当前的聊天界面的情况下，只需要点击相应自己想查看的其他用户，点击他们头像即可，便可快速切换聊天界面。

2、优化部分性能问题

（1）在 APPTab 主界面，添加了快速快捷加号菜单。点击加号菜单，动态弹出添加好友、添加群组、搜索群组、二维码四个按钮，可快速进入相应功能。

（2）Notification 通知中心是很常用的一种观察者模式，项目中有不少地方在使用。不过经过仔细研究代码之后，发现相同的注册中心会重复接受到相同的通知，从而会进行相同的操作，做无用功。所以，现将发现的重复操作进行了修改，减少了重复操作。

（3）APP 中用到不少第三方 SDK，为了更加便捷的更换 APPKEY，将所有第三方 APPKEY 整理到一个宏定义文件。

（4）在 Bolock 中使用 self 强引用对象，有可能会循环引用，产生内存泄漏的问题，所以将发现的强引用改为弱引用。