Disappearing messages

- ability to turn on/off disappearing messages feature per conversation, specify ttl
- use preferences framework, preference affects local deletion of both sent and received messages
- special chat item on change
- in direct chat - chat item can be interacted with to agree or disagree with preference change, updates preference accordingly
    - how does party that offered preference change learn about disagreement? (maybe just no preference update from contact is ok, since it's still not on if it's not mutual)
    - how does it learn about disagreement on ttl? (it's on already - so it works for both but differently if there's no agreement)
    - single updating chat item or per event? (probably per event is better since they can be spaced in time)
- in group - set by owner
- should it be allowed to be configured globally?
- change of setting shouldn't prevent previous disappearing messages from being deleted

Design

- add `delete_at` field to `chat_items` table, index `idx_chat_items_delete_at`
- add `disappearingItems :: TMap ChatItemId (Async ())` to ChatController (use `Weak ThreadId`?)
- new background process that periodically scans for disappearing messages bound to be deleted during next 30 minutes:
    - add `cleanupManager :: TVar (Async ())` to ChatController
    - periodically gets items to schedule for deletion based on delete_at field
    - for items to be deleted in next 30 minutes - add thread to disappearingItems - thread delays until deleteAt date, then deletes and sends CRChatItemDeleted to view
    - for items past current time - delete in bulk

    - race condition between bulk deletion of expired items on start and opening a chat with them - they should be removed from chat view once deleted - don't optimize for bulk deletion and create threads? create multiple CRs after bulk deletion? create single chat response with all ids?

- when chat item is deleted locally, either by user or via "delete for everyone" feature, kill thread and remove from map
- when MsgContent chat item is sent or marked read, add thread to disappearingItems based on chat preference
- UI shows timer based on chat item's createdAt date and deleteAt date

***

Preference agreement:

- new preference types?

  ``` haskell data DisappearingMessagesPreference = DisappearingMessagesPreference { allow :: FeatureAllowed, ttl :: Int }

  data DisappearingMessagesGroupPreference = DisappearingMessagesGroupPreference { enable :: GroupFeatureEnabled, ttl :: Int }

  -- requires changing functions and types using Preference and GroupPreference ```
- chat items to contain old and new preference value

***

Maybe agreement shouldn't be via preferences framework, but ad-hoc? For example:

- new protocol messages `XMsgTtlOffer ttl`, `XMsgTtlAgree ttl`, `XMsgTtlOff`
- for direct chats on XMsgTtlOffer contact `disappearingMessages` fields is updated to
- for direct chats on XMsgTtlAgree check ttl equals offered, then turn on
- for group chats only XMsgTtlAgree has to be sent, should only be accepted from owner
- XMsgTtlOff turns off unconditionally, for group chats should only be accepted from owner
- types:
  ```

``` haskell data DisappearingMessagesState = DMSOff | DMSOffered ttl | DMSAgreed ttl

data Contact = Contact { ... disappearingMessagesState :: DisappearingMessagesState, ... }

data GroupInfo = GroupInfo { ... disappearingMessagesState :: DisappearingMessagesState, ... }

-- make part of ChatSettings? ```
```