Technical guide of Readbo:  
How Readbo works on the backend.

Server side information:

Readbo was programmed for Apache, PHP 5.3 and MySQL.  
I have used Code Igniter as the framework, it is MVC and fairly usual structure.

Client side information:

Readbo client side is using jQuery 1.7. There is lots of Javascript files (30), this is needed for all the interface of Readbo (dialog boxes, ajax, display as quickly as possible all the news sent by PHP, twitter, facebook, etc… avoiding changing pages as much as possible).

One quick note about Javascript, it is being entirely compiled before being released on production, this is to create minified files and as less files as possible (1 per page).

What happens on the backend?

I’ll quickly go over a few important things that happen on the backend, here are some MySQL tables I’ll be referring to:

**Users**: contains a row per user subscribed on the website, contains a user\_id  
**Feeds**: contains a row per RSS, Facebook or Twitter feed, contains a feed\_id  
**Subscriptions**: A user can subscribe to a specific feed (example Digg.com), there will be only one feed digg.com with a feed\_id of 14, but hundreds of users can subscribe to the same feed, each subscription will have its row here. Contains a feed\_id to refer to the feed, and a user\_id to refer to the user which will be subscribing.  
**Items**: contains a row per news, example for CNN.com, each news on the website will have its own row in this table. The row contains the item\_sid that is a unique id to identify a news. The rest of the information is the date, the title, the content, the author etc…  
**Lus**: table per item that is read by a user, lets say “Dave” reads “Harry Potter came out on DVD” from CNN.com, lus will store the item\_id of the news, and the user that read it. So that the same news does not get shown again to “Dave”

User flow:

1. User signs up -> new row added to table **Users** with id 45
2. User subscribes to CNN.com -> new row added to table **Feeds** if the Feed was never created before (Feed\_id 85) -> new row added to **Subscriptions** with feed\_id 85 and user\_id 45
3. *--- A process runs that grabs all the news from CNN.com, I’ll talk about it below ---*
4. The list of news is being sent back to the users grabbed from **Items** table, being sent back to the user and displayed through Javascript
5. User can view all the items, he clicks on an item which marks it as read -> An Ajax request is being sent to the server, the server knows a user read a news, and marks it as read by create a row in the table **Lus**.

Process grabbing news:

There is a cron job (schedule processes) that runs every 2 minutes to grab the latest news from all the feeds. This is what was happening:

1. A specific feed is being parsed, example CNN.com
2. From the **Feeds** database, it was grabbing some data, the RSS url (example for CNN: <http://rss.cnn.com/rss/cnn_topstories.rss>), next\_update (a field that determines when CNN.com should be next updated, to not update too often)
3. Grabbing all the news from the RSS url, and generating an item\_sid (unique id) for every news, if the item\_sid is already in **Items** table, this means that the news was already parsed, if not creating a new row in Items.
4. After a feed is being considered parsed, it is being reviewed, has there been any new items added? If not then it wasn’t useful to parse it, then it should be updated less often. A specific field in the table Feeds is being changed, called avg\_priority that analysis when was the last time that a feed was updated. Depending on this data, the field next\_update is being updated so that the process will know when to parse this feed again. Some feeds might only get updated a week later.