Coding Standards

The proposed coding standards for PHP is the one defined by the *PHP Framework Interop Group* that is a group trying to create standards for PHP coding.
Their standards are very easy to use and simple to understand. They all can be found here:

http://www.php-fig.org/

In particular we are interested in the PSR-1 standard and PSR-2 standard for general coding style. These are the most important points:

- take particular attention to file formats (UTF-8 without BOM) mixing file formats or including BOM characters may create unexpected behaviour of the webpages
- File endings must be only LF characters (Unix Style). While this may not be a problem for the webpages, it can be a mess while using a Version Control System like git (especially since we are using Windows, Mac and Linux in the team).
- NO TABS. Use only spaces for indenting code, 4 spaces
- The closing ?> tag MUST be omitted from files containing only PHP.
- PSR-2 guide contains all the details about using spaces, braces, etc.

There is no coding standard for CSS, HTML and Javascript. However I would recommend keeping the same standards proposed for PHP, in terms for indentation, line endings, spaces, etc.

HTML code must be HTML5 compliant, and possibly XML compliant. We should avoid using html attributes to give style to a page, using CSS instead.

Development Environment

I think everyone is free to use whatever IDE he likes for writing code, as long as it allows to follow the coding standards above.

If someone is still looking for a PHP IDE, I would suggest PHPStorm: it is a very good IDE, it is generally for pay, but it's free for students (you just need to provide your academic email address and you are given a free licence automatically).
It supports customization of the code style, so you can set it up for following PHP-FIG standards. The PHP-FIG standard is even built into PHPStorm, you can choose to use it in the settings > code style > PHP > set from...

GitHub Repository Standards

Each of the developers should have a proper git installation on his system to be able to push changes to the GitHub repository.

Follow these instructions if you need to install it: https://help.github.com/articles/set-up-git/ (if you are using linux you can just install git from your package manager)

Remember to setup your name and email address accordingly as shown in the guide.

Second step, you should setup the line-ending policy, in order to avoid making a mess between all the developers and their different OSes. You can follow this guide to setup it up:
https://help.github.com/articles/dealing-with-line-endings/
Most users will want to set the autocrlf setting to the *input* value as explained in this post:
http://stackoverflow.com/questions/3206843/how-line-ending-conversions-work-with-git-core-autocrlf-between-different-operat

For branches usage, the proposed structure is derived from this popular policy:
http://nvie.com/posts/a-successful-git-branching-model/
All git repos have a **master** branch by default. This will be used for publishing already stable/finished features.
Then we will have a **dev** branch, it will be used to actually develop the new code. dev can contain code that is still incomplete. This branch and the master one, are the only ones with a possibly infinite lifetime.

To avoid conflicts while each of us develops a feature, it is however recommended to have one branch (based on the dev branch) for each feature. When you need to start a new feature of the product, especially if you expect it to be huge or touch many files, you should create a new branch, with a proper name representing the feature you are implementing. This branch should contain only commits pertaining to its feature. When the feature is finished, the branch will be merged in dev to properly integrate it with the rest of the new commits and features. After this final merge, it can be deleted.