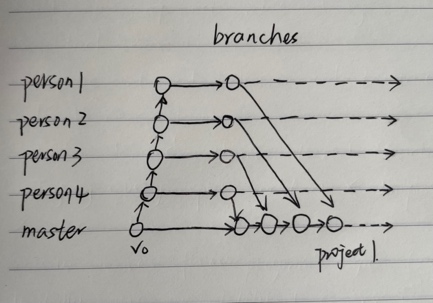
**Develop Document**

1. Github develop branches flow diagram:



1. Coding style uniform specifications: You can use tools to format your code.
   1. Back-end: google Go coding style
   2. Front-end: JavaScript beautifier
      1. **Class** name should be written in **PascalCase**, ex. TextNode
      2. **Functions** and **variables** should be written in **camelCase**, ex. writeTextNode
      3. **Constant** should written in **upper case**, ex. READING
      4. It’s better to use single line annotation.
      5. Explanation should be written in /\*\*…\*/ for functions, etc.(Front-end)

Ex./\*\*

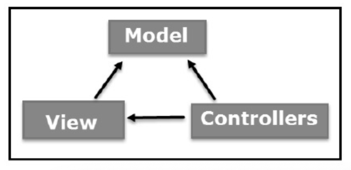
\*

\*@const

\*@type{string}

\*/

1. Other specifications:
   1. Everyone should have the responsibility to code review others code and requests change when he/she founds non-compliance of the rules
   2. Names should tell the meanings, ex. boolean isChangZhouAGirl
   3. Appropriate spaces, intentations, wraps and blank lines, ex. A binary operator must have a single blank column(space) both before and after/ Left curly bracket must have a space before it
2. Front-end develop structure: MVC framework



* 1. Model: The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.
  2. View: The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.
  3. Controller: Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.
  4. Three logical components should be independent

1. To be continued…