# HASKELLOG

A Blog using Haskell

consists of

a Restful API and a dynamic web application

Documentation & Manual

**Description**

**Haskell blog** is project which uses Restful API as an intermediate between dynamic web pages (rendered with Haskell) and database (SQLite) ,the API controls and manages the web pages also it maintain and manipulate the database.

**Components**

1. Restful API.
2. Web application.
3. Database.

**Definitions**

**RESTful API(Representational State Transfer):** It's an architectural style that defines a set of constraints and properties based on HTTP. Web Services that conform to the REST architectural style, or REST-ful web services, provide interoperability between computer systems on the Internet. REST-compliant web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations. Other kinds of web services, expose their own arbitrary sets of operations.

**SQLite:** is a database engine and relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

**Development tools**

* **Stack project manager:** It's a development tool for the entire Haskell development cycle from installing the compiler to the testing and the benchmarking.
* **GHC v8.0.2:** it is an open source, compiler and interactive environment for the functional language Haskellin addition to some libraries.

**Framework**

**Spock is a lightweight Haskell web framework**

* + - * Used for rapid design
      * Applications supported by Spock are easily deployed using stack
      * Open sourced on Github

**Project main files**

**blog.cfg :** configuration file contain db name, port, blog name, etc.

**haskellog.cabal :** contains info about the package, some of these info are needed to build the package.

**haskellog.db : sqlite** database file which contains the encrypted data using cryptohash library.

**stack.yaml :** file contains human readable data serialization lang. Used for configuring files.

**static/ :** directory contains all the static files like .js & .css files

**src/** : directory contains all the source code of the project

**Main.hs** : main haskell function to run.

**Model/** : contain the abstraction data types

**Web/** : directory separates the view related files from the control files

**Blog.hs** : first and only file to be imported by main.hs ,this file controls all the functionalities of the web site, almost access all the haskell source files

**Utils.hs :** responsible for start and end Conn. With the DB server ,also handle the sessions which start and end through logging in&out processes

**View/** : directory contain the rendered html files using blaze-html lib which explained below

**Source code Hierarchy**

In src directory

├── Main.hs

├── Model

│ ├── CoreTypes.hs

│ └── ResponseTypes.hs

└── Web

├── Actions

│ └── User.hs

├── Blog.hs

├── Forms

│ ├── Common.hs

│ ├── Login.hs

│ ├── Post.hs

│ └── Register.hs

├── Utils.hs

└── Views

├── Home.hs

└── Site.hs

**Libraries functions**

* **Blaze-html class**
  + it is a library in Haskell to render .html files to .hs files
  + A blazingly fast HTML combinator library for the Haskell programming language
  + It supports html5 ,html4 ,Xhtml1
* **Monad class**
  + Used to create sequential instructions from the functional ones using do keyword
  + Used in almost all the functional programming languages (scala, huskell ,F#..etc.)
  + Based on binding the return of each line to the parameter on the consecutive one

* **Aeson class**
  + It's a fast Haskell library for working with JSON data.
  + JSON parsing and encoding library optimized for ease of use and high performance.
* **Crypto.hash class**
  + A collection of crypto hashes, with a practical incremental and one-pass, pure APIs, with performance close to the fastest implementations available in other languages.
  + this library is now deprecated in favor of "cryptonite", which is a superset of this, and also have more functionality
* **Byte String class**
  + An efficient compact, immutable byte string type (both strict and lazy) suitable for binary or 8-bit character data, suitable for high performance use, both in terms of large data quantities, or high speed requirements.
  + it is easy to convert code from using String to ByteString.

**Code documentation**