CE Stack Exchange Project

Document No. 1

توسعه توسط گروه

Web*something*ers

تدوین

دانا افاضلی

۱۳۹۹/۰۶/۰۵

این متن شامل توضیحاتی درباره ساختار کلی پروژه وب اپلیکیشن Made In Lobby به همراه نسخه اولیه

رابط کاربری میان Front-end و Back-end می‌باشد

فهرست

ساختار کلی پروژه

پروژه به طور کلی متشکل از دو بخش Back-end و Front-end است که هرکدام در فولدر root پروژه به طور جداگانه قرار دارند. همچنین در فولدر root پروژه یک فولدر بنام Config وجود دارد که در این فولدر فایل‌های مربوط به تنظیمات پروژه اعم از API بین Front-end و Back-end قرار دارند.

Back-End

این قسمت از معماری MVC پیروی می‌کند و در نتیجه همانطور که انتظار می‌رود در فولدر Back-end سه فولدر Model، View و Controller قرار دارند که به تشریح هرکدام می‌پردازیم.

Model

این قسمت متشکل از دو بخش زیر است

- تعریف موجودات پروژه که از طریق آنها دیتابیس ساخته می‌شود و نحوه استفاده بقیه اجزاء برنامه مشخص می‌شود.
- پل ارتباطی Back-end (و در نتیجه کل پروژه) با دیتابیس در قسمت مدل قرار دارد. به این شکل که مدل تنها جایی از برنامه است که با دیتابیس (که یا روی همان دستگاه یا روی دستگاه دیگر قرار دارد) ارتباط برقرار می‌کند.

Controller

این قسمت مسئول پردازش و تحلیل درخواست‌های Front-end است، به این شکل که هنگام دریافت یک درخواست از طرف Front-end، پردازش‌های زیر رخ می‌دهد

۱. Token-based authentication: اول از همه باید درخواست دریافتی Authenticate شود. به این معنا که نخست توسط Middle-ware های موجود (مانند JWT و...) Token درخواست چک شده و در صورت اعتبار از این بخش عبور خواهد کرد. همچنین امکان پردازش های دیگری توسط Middle-ware وجود دارد.

۲. Access control: در صورت اعتبار اکانت، یک نقش برای آن اکانت در نظر گرفته می‌شود که با توجه به آن نقش، یک سری محدودیت برای هر اکامت پدید می‌آید. بخش Controller قبل از اینکه هر گونه اطلاعاتی را پردازش کند مسئول چک کردن این دسترسی‌ها است.
۳. Routing: در صورت وجود دسترسی، بخش Controller باید با توجه به URL دریافتی، سرویس مورد نظر را از بخش View صدا بزند و جواب آنرا به بخش Front-end بازگرداند.
    - توجه داشته باشید که فرآیند Routing میتواند در View نیز انجام پذیرد.

View

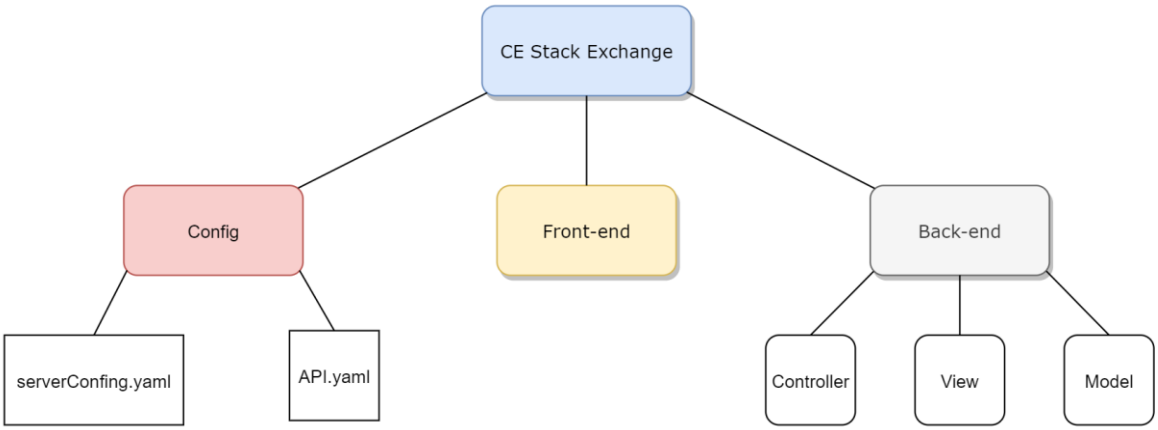
همانظور که در بالا نیز اشاره شد، بخش ویو مسئول پیاده سازی سرویس های مورد نظر Front-end است. به این شکل که یک API در اختیار Front-end قرار داده و اطلاعات مورد نظر را از Model به صورت آبجکت‌ها و یا استراکت‌های تعریف شده در Model، گرفته و به شکل Json یا هر نوع دیگری بازمی‌گرداند.

- توجه داشته باشید که فرآیند Routing میتواند در View نیز انجام پذیرد.

Front-end

هنگامی که یک کاربر از طریق کلاینت خود (مانند مرورگر) به IP یک سرویس درخواست ارسال می‌کند، این درخواست در قسمت Front-end آن سرویس دریافت و پردازش می‌شود. پس یک رابط کاربری اولیه باید از طرف Front-end برای استفاده کلاینت‌ها تعریف شود هرچند که درخواست اولیه تنها به خود IP آن سرویس است و در ادامه از طریق اجزای خود Front-end عملیات درخواست زدن اتفاق می‌افتد، اما مشخص کردن همین رابط برای راحتی کار خالی از لطف نیست.

بخش Front-end برای پردازشِ درخواست آمده، ابتدا فایل‌های Static لازم (مانند JS، CSS،HTML) را قرار داده و سپس با ارسال درخواست به Back-end، اطلاعات Dynamic مربوط به آن صفحه را دریافت کرده و در صفحه قرار می‌دهد سپس صفحه آماده شده را به کلاینت باز می‌گرداند.



Project overall structure

API بین Front-end و Back-end

همانطور که در بالا اشاره شد، وجود یک پروتکل ارتباطی یا یک رابط میان Front-end و Back-end الزامی است. در این قسمت بخشی از این API که برای فاز اول پروژه در نظر گرفته شده است تدوین شده است.

API Description

Account management

- Sign up a user: POST /signup

  Parameters: username، password، email، student code etc. (exact description can be checked from model)

  Format: Json

  Returns: message containing the response (either an error or successful notification or any kind of instruction)

- Login a user: POST /login

  Parameters: username، password

  Format: Json

  Returns: authentication token if successful or error if an error occurred.

- View basic information of a profile: GET /my/basic_Info
- View asked questions of a profile: GET /my/questions

Community management

- Get available communities: GET /communities
- Create a new Community: POST /communities/new
- Get recent posts of a Community: GET /communities/{id}

Post managements

- Get basic information of a question: GET /questions/{id}/basic_info
- Get answers to a question: GET /questions/{id}/answers

- Get basic information of an answer: GET /answers/{id}/basic_info

User actions

- Ask a question: POST /user/actions/ask
- Answer a question: POST /user/actions/answer/{id}
- Comment on a question: POST /user/actions/comment/on_question/{id}
- Comment on a comment: POST /user/actions/comment/on_comment/{id}

توجه کنید که این API در بخش Config/API.yaml قرار دارد که همانطور که پیداست از نوع yaml است. Yaml یک روش ذخیره و انتقال داده مانند Json است که به دلیل سادگی و فهم آسان، معمولا برای فایل‌های Config بکار می‌رود.