بسمه تعالی

پروژه پایانی بوتکمپ جنگو کوئرا

سامانه انتخاب واحد دانشگاه

پروژه پایانی بوتکمپ کوئرا یک سامانه جامع برای خدمات دانشگاهی برای دانشجویان از جمله انتخاب واحد، ترمیم واحد (حذف و اضافه)، ارائه دروس و نمایش آن برای دانشجو، کارنامه و وضعیت تحصیل دانشجو، حذف اضطراری دروس و ... می‌باشد که قصد داریم با استفاده از فریمورک جنگو و Django Rest Framework و دیتابیس PostgreSQL و همچنین ابزارهای دیگری که در متن پروژه آورده شده است، یک API سرور برای این سامانه طراحی کنیم.

توجه داشته باشید که بخش بیشتر طراحی پروژه بر عهده‌ی تیم‌تان است و در این کار منتورها به شما کمک خواهند کرد.

در ابتدا موجودیت‌ها (Entity) توضیح داده می‌شوند و سپس معماری سیستم و API های سامانه و در آخر نیز یک سری موارد مورد انتظار مطرح می‌شود که شما در پروژه باید حتما این موارد را پیاده‌سازی کنید و اگر امکانات بیشتری علاوه بر موارد گفته شده در پروژه داشته باشید بر اساس نظر تیم تصحیح نمره‌ی امتیازی به تیم تعلق خواهد گرفت.

1. موجودیت‌های سامانه

موارد گفته شده در هر موجودیت باید حتما در سامانه وجود داشته باشد ولی شما هم می‌توانید بر حسب نیاز خود یک سری موارد دیگر هم به این موجودیت‌ها اضافه کنید.

به این مورد هم توجه داشته باشید که موارد زیر یک سری موجودیت و صفات آن‌ها هستند و لزوما مدل دیتابیس شما نباید مانند این موارد باشد.

a. کاربر

- نام و نام خانوادگی
- شماره دانشجویی / استادی / کاربری
- رمز عبور

- عکس پروفایل
- ایمیل
- تلفن همراه
- کد ملی
- جنسیت
- تاریخ تولد

b. دانشجو

- سال ورودی
- ترم ورودی
- معدل
- دانشکده
- رشته تحصیلی
- دروس گذرانده
- دروس در حال گذراندن
- استاد راهنما
- وضعیت نظام وظیفه (در صورت پسر بودن)
- سنوات

c. استاد

- دانشکده
- رشته
- تخصص
- مرتبه
- دروس تدریسی گذشته

d. مدیر IT

e. معاون آموزشی

- دانشکده
- رشته

f. درس مصوب

- نام درس
- دانشکده ارائه دهنده
- پیش‌نیازها
- هم‌نیازها
- تعداد واحد درس
- نوع درس (عمومی - تخصصی - پایه - اختیاری - ...)

g. درس ترمی

- روز و زمان کلاس
- تاریخ و زمان امتحان
- مکان امتحان
- استاد درس
- ظرفیت درس
- ترم تحصیلی

h. درس - دانشجو

- وضعیت درس
- نمره دانشجو
- ترم اخذ شده

i. ترم

- نام ترم
- دانشجوها و اساتیدی که در ترم ثبت‌نام کردند
- لیست دروس ترمی
- زمان شروع و پایان انتخاب واحد
- زمان شروع و پایان کلاس‌ها
- زمان شروع و پایان ترمیم
- زمان پایان حذف اضطراری
- زمان شروع امتحانات
- زمان اتمام ترم

j. دانشکده

- نام دانشکده

k. رشته

- نام رشته
- گروه آموزشی
- دانشکده
- تعداد واحد
- مقطع

l. درخواست ثبت‌نام (انتخاب واحد)

- دانشجوی درخواست دهنده
- دروس درخواستی
- وضعیت تایید

m. درخواست ترمیم (حذف و اضافه)

- دانشجو

- دروس حذف
- دروس اضافه
- وضعیت تایید

n. درخواست تجدیدنظر

- دانشجو
- درس
- متن تجدیدنظر
- پاسخ تجدیدنظر

o. درخواست حذف اضطراری دانشجو

- دانشجو
- درس
- نتیجه‌ی درخواست
- توضیحات دانشجو
- توضیحات معاون آموزشی

p. درخواست حذف ترم دانشجو

- دانشجو
- ترم
- نتیجه‌ی درخواست (با سنوات یا بدون سنوات)
- توضیحات دانشجو
- توضیحات معاون آموزشی

q. درخواست اشتغال به تحصیل ویژه پسران

- دانشجو
- فایل اشتغال به تحصیل

- ترم تحصیلی
- محل صدور گواهی

2. اندپوینت‌های سامانه

در این قسمت URL ها و متدهای هر API به شما داده می‌شود ولی شما می‌توانید از URL ای که بنظرتان بهتر است استفاده کنید و اجباری به استفاده از URL پیشنهادی نیست اما لاجیک هر API باید مطابق خواسته‌ی صورت پروژه باشد. شما باید هر API را ورژن بزنید. اگر ابهامی در چگونگی پیاده‌سازی ورژن زدن API دارید سرچ کنید :)

تمام API های لیست هم باید دارای pagination باشند یعنی صفحه به صفحه باشند و تعداد رکوردها در هر پیج هم دیفالت ۱۰ است و کاربر می‌تواند در هدر یا کوئری پارامتر آن را تغییر دهد.

a. ورود کاربر و احراز هویت

در این API شما باید با استفاده از Token کاربر را احراز هویت کنید. (الزامی به استفاده از Token خود DRF نیست و هر نوع Token Authentication ی قابل قبول است)

- POST /users/login/
- POST /users/logout/

تغییر رمز عبور (فراموشی رمز عبور) با endpoint های زیر انجام میشود. ابتدا باید درخواست تغییر به بکند ارسال شود و بکند یک ایمیل حاوی توکن یا یک کد یکبار مصرف به ایمیل کاربر با استفاده از سلری ارسال می‌کند. سپس کاربر با فرستادن کد یکبار مصرف دریافت شده به همراه رمز عبور جدید میتواند عملیات تغییر رمز را انجام دهد. (ذخیره موقت توکن کاربر در ردیس انجام شود)

- POST /users/change-password-request/
- POST /users/change-password-action/

b. اندپوینت‌های مدیر IT

- POST /admin/professors/
- GET /admin/professors/

با استفاده از این دو API، مدیر IT یک استاد را ایجاد و لیست اساتید را دریافت می‌کند.

فیلدهای موجود در لیست به عهده‌ی خودتان است.

API لیست باید فیلتر بر اساس نام و نام خانوادگی، شماره استادی، کدملی، دانشکده، رشته، مرتبه برای استاد داشته باشد.

- GET /admin/professor/{pk}/
- PUT /admin/professor/{pk}/
- DELETE /admin/professor/{pk}/

این سه API اطلاعات یک استاد را می‌گیرد، اطلاعات را تغییر می‌دهد و یک استاد را پاک می‌کند.

(می‌توانید بجای pk از slug یا هر چیز که مد نظرتان است استفاده کنید)

- POST /admin/students/
- GET /admin/students/

با استفاده از این دو API، مدیر IT یک دانشجو را ایجاد و لیست دانشجویان را دریافت می‌کند.

فیلدهای موجود در لیست به عهده‌ی خودتان است.

API لیست باید فیلتر بر اساس نام و نام خانوادگی، شماره دانشجویی، کدملی، دانشکده، رشته، سال ورودی و وضعیت نظام وظیفه برای دانشجو داشته باشد.

- GET /admin/student/{pk}/
- PUT /admin/student/{pk}/
- DELETE /admin/student/{pk}/

این سه API اطلاعات یک دانشجو را می‌گیرد، اطلاعات را تغییر می‌دهد و یک دانشجو را پاک می‌کند.

(می‌توانید بجای pk از slug یا هر چیز که مد نظرتان است استفاده کنید)

- POST /admin/assistants/
- GET /admin/assistants/

با استفاده از این دو API، مدیر IT یک معاون آموزشی را ایجاد و لیست معاونین آموزشی را دریافت می‌کند.

فیلدهای موجود در لیست به عهده‌ی خودتان است.

API لیست باید فیلتر بر اساس نام و نام خانوادگی، شماره کارمندی، کدملی، دانشکده، رشته برای معاون داشته باشد.

- GET /admin/assistant/{pk}/
- PUT /admin/assistant/{pk}/
- DELETE /admin/assistant/{pk}/

این سه API اطلاعات یک معاون آموزشی را می‌گیرد، اطلاعات را تغییر می‌دهد و یک معاون آموزشی را پاک می‌کند.

(می‌توانید بجای pk از slug یا هر چیز که مد نظرتان است استفاده کنید)

- POST /admin/faculties/
- GET /admin/faculties/

با استفاده از این دو API، مدیر IT یک دانشکده را ایجاد و لیست دانشکده‌ها را دریافت می‌کند.

فیلدهای موجود در لیست به عهده‌ی خودتان است.

- GET /admin/faculty/{pk}/
- PUT /admin/faculty/{pk}/
- DELETE /admin/faculty/{pk}/

این سه API اطلاعات یک دانشکده را می‌گیرد، اطلاعات را تغییر می‌دهد و یک دانشکده را پاک می‌کند.

(می‌توانید بجای pk از slug یا هر چیز که مد نظرتان است استفاده کنید)

- POST /admin/term/
- GET /admin/term/
- GET /admin/term/{pk}/
- PUT /admin/term/{pk}/
- DELETE /admin/term/{pk}/

پنج API بالا برای ساخت و دریافت لیست و دریافت جزییات و تغییر و پاک کردن یک ترم است.

(می‌توانید بجای pk از slug یا هر چیز که مد نظرتان است استفاده کنید)

c. اندپوینت‌های مربوط به دروس

- POST /subjects/
- GET /subjects/
- GET /subjects/{pk}/
- PUT /subjects/{pk}/
- DELETE /subjects/{pk}/

اندپوینت اول یک درس مصوب را ایجاد می‌کند و تنها مدیر IT و معاون آموزشی **همان** دانشکده می‌توانند دروس آن دانشکده را ایجاد کند.

اندپوینت دوم لیست دروس را نشان می‌دهد که بر اساس دانشکده و نام فیلتر می‌شود. این لیست برای همه قابل نمایش است.

اندپوینت سوم جزییات یک درس را نشان می‌دهد که برای همه قابل نمایش است.

اندپوینت چهارم و پنجم هم مربوط به تغییر و پاک کردن یک درس است که تنها مدیر IT و معاون آموزشی **همان** دانشکده می‌توانند این عملیات‌ها را انجام دهند.

- POST /courses/

- GET /courses/
- GET /courses/{pk}/
- PUT /courses/{pk}/
- DELETE /courses/{pk}/

اندپوینت اول یک درس ترمی را ایجاد می‌کند و تنها مدیر IT و معاون آموزشی **همان** دانشکده می‌تواند دروس آن دانشکده را ایجاد کند و امکان ایجاد هر درس ترمی پس از پایان زمان ترمیم تا انتهای ترم وجود ندارد اما درس مصوب هر زمانی می‌تواند ایجاد شود. اگر نوع درس عملی باشد، امتحانی برای آن وجود نخواهد داشت.

اندپوینت دوم لیست دروس را نشان می‌دهد که بر اساس دانشکده و نام و ترم فیلتر می‌شود. این لیست برای همه قابل نمایش است.

اندپوینت سوم جزییات یک درس را نشان می‌دهد که برای همه قابل نمایش است.

اندپوینت چهارم و پنجم هم مربوط به تغییر و پاک کردن یک درس است که تنها مدیر IT و معاون آموزشی **همان** دانشکده می‌توانند این عملیات‌ها را انجام دهند.

d. اندپوینت‌های دانشجو و استاد

معاون آموزشی می‌تواند با استفاده از اندپوینت‌های زیر همانند ادمین، لیست اساتید و دانشجویان دانشکده خود را ببیند و روی آن لیست‌ها فیلترهای گفته شده را اعمال کند. همچنین می‌تواند جزییات استاد و دانشجو دانشکده خود را مانند ادمین مشاهده کند.

- GET /students/

- GET /students/{pk}/
- GET /professors/
- GET /professors/{pk}/

دانشجو و استاد نیز می‌توانند صرفا اطلاعات مربوط به خودشان را ببینند و اصلاح کنند، اما حق ندارند شماره دانشجویی و استادی خود را اصلاح کنند.

- PUT /students/{pk}/
- PUT /professors/{pk}/

e. اندپوینت‌های ترم و دروس قابل اخذ

هر دانشجو و استاد می‌تواند با استفاده از URL های زیر لیست و جزییات ترم‌ها را مشاهده کند.

- GET /terms/
- GET /term/{pk}/

همچنین دانشجو می‌تواند با استفاده از URL زیر دروس قابل اخذ خودش را مشاهده کند.

- GET /student/{pk/me}/my-courses/

در این API باید با توجه به وضعیت پیشنیازها و همنیازها این لیست را نمایش دهد.

گزارش لیست دروس گذرانده توسط دانشجو و وضعیت گذراندن هر نوع درس توسط دانشجو.

- GET /student/{pk/me}/pass-courses-report/

این API توسط ادمین، معاون آموزشی همان دانشکده و استاد راهنمای دانشجو نیز قابل مشاهده است.

دروس در حال گذراندن هر دانشجو در هر ترم نیز در گزارش زیر قابل مشاهده است که این API توسط ادمین، معاون آموزشی همان دانشکده و استاد راهنمای دانشجو نیز قابل مشاهده است.

- GET /student/{pk/me}/term-courses/

تعداد سنوات باقیمانده هم با API زیر به دانشجو نشان داده می‌شود.

- GET /student/{pk/me}/remaining-terms/

f. اندپوینت‌های انتخاب واحد

برای انتخاب واحد، در زمان شروع انتخاب واحد دانشجو می‌تواند با اندپوینت زیر یک فرم انتخاب واحد ایجاد کند و اگر در زمانی جز انتخاب واحد اقدام به این کار کند با خطا مواجه شود. با اندپوینت دوم جزییات آن و دروسی که انتخاب کرده را مشاهده کند.

- POST /student/{pk/me}/course-selection/create/
- GET /student/{pk/me}/course-selection/

در ادامه با اندپوینت‌های زیر یک درس را می‌توان برداشت و یا یک درس را حذف کرد. اندپوینت اول تغییرات را اعمال نمیکند و فقط خطاها را در صورت وجود نشان می‌دهد و اندپوینت دوم تغییرات را اعمال می‌کند.

- POST /student/{pk/me}/course-selection/check/
- POST /student/{pk/me}/course-selection/submit/

دقت کنید که درخواست‌های زیر به صورت **اتمیک** اجرا می‌شوند و اگر یک درس با خطا مواجه شود کلا تغییرات هیچ درسی اعمال نمی‌شود.

در موارد بالا باید به خطاهای زیر توجه کنید:

- درس پیشنیاز حتما باید در وضعیت قبول باشد.
- درس تکراری یا پاس شده نمی‌توان برداشت.
- درس تکمیل را نمی‌توان اخذ کرد.
- درس همنیاز را نمی‌توان زودتر از درسی که همنیاز آن شده حذف کرد.
- در صورت معدل ترم پیش بالای ۱۷ دانشجو حق دارد ۲۴ واحد اخذ کند و در غیر این صورت ۲۰ واحد در یک انتخاب واحد می‌تواند اخذ کند.
- تداخل زمانی در امتحان و کلاس نباید وجود داشته باشد.
- دانشجو در صورت داشتن سنوات می‌تواند انتخاب واحد کند.
- تنها دروس مرتبط به رشته را می‌توان برداشت.
- و هر خطایی که بنظر شما منطقی می‌باشد باید پیاده‌سازی شود.

می‌توان یک درس را در انتظار گذاشت و در صورت اینکه یک نفر این درس را حذف کند، دانشجوی در صف انتظار می‌تواند این درس را اخذ کند، برای این کار می‌توانید از ردیس استفاده کنید.

پس از اتمام کار، دانشجو قبل از پایان زمان انتخاب واحد، می‌تواند فرم خود را تایید و به استاد راهنما ارسال کند. پس از آن دیگر حق تغییر دروس اخذ شده‌ی خود را نخواهد داشت.

- POST /student/{pk/me}/course-selection/send-form/

استاد نیز با اندپوینت زیر می‌تواند لیست و جزییات انتخاب واحد هر دانشجوی خود را مشاهده کند.

- GET /professor/{pk/me}/students-selection-forms/
- GET /professor/{pk/me}/students-selection-forms/{s-pk}/

سپس با API زیر می‌تواند انتخاب واحد را رد و یا تایید کند. در صورت رد کردن استاد، فرم دوباره برای دانشجو تا پایان انتخاب واحد باز می‌شود. تا بتواند تغییرات لازم را اعمال کند و دوباره برای استاد ارسال کند. در صورت تایید نیز دروس به دروس انتخابی در ترم جاری اضافه می‌شوند.

- POST /professor/{pk/me}/students-selection-forms/{s-pk}/

سپس با تایید استاد، یک ایمیل توسط سلری به دانشجو ارسال می‌شود که استاد این دروس را تایید کرد و برنامه هفتگی دانشجو را برایش ارسال می‌کند. در صورتی که دانشجو تا پایان انتخاب واحد فرم را ارسال نکند، سلری این فرم را تایید و برای استاد ارسال می‌کند.

g. اندپوینت‌های ترمیم (حذف و اضافه)

برای ترمیم، در زمان آن دانشجو می‌تواند با اندپوینت زیر یک فرم ترمیم ایجاد کند و اگر در زمانی جز ترمیم اقدام به این کار کند با خطا مواجه شود. با اندپوینت دوم جزییات آن و دروسی که انتخاب کرده را مشاهده کند.

- POST /student/{pk/me}/course-substitution/create/
- GET /student/{pk/me}/course-substitution/

در ادامه با اندپوینت‌های زیر یک درس را می‌توان برداشت و یا یک درس را حذف کرد. اندپوینت اول تغییرات را اعمال نمیکند و فقط خطاها را در صورت وجود نشان می‌دهد و اندپوینت دوم تغییرات را اعمال می‌کند.

- POST /student/{pk/me}/course-substitution/check/
- POST /student/{pk/me}/course-substitution/submit/

دقت کنید که درخواست‌های زیر به صورت **اتمیک** اجرا می‌شوند و اگر یک درس با خطا مواجه شود کلا تغییرات هیچ درسی اعمال نمی‌شود.

در موارد بالا باید به خطاهای زیر توجه کنید:

- درس پیشنیاز حتما باید در وضعیت قبول باشد.
- درس تکراری یا پاس شده نمی‌توان برداشت.
- درس تکمیل را نمی‌توان اخذ کرد.
- درس همنیاز را نمی‌توان زودتر از درسی که همنیاز آن شده حذف کرد.

- در صورت معدل ترم پیش بالای ۱۷ دانشجو حق دارد ۲۴ واحد اخذ کند و در غیر این صورت ۲۰ واحد در یک انتخاب واحد می‌تواند اخذ کند.
- تداخل زمانی در امتحان و کلاس نباید وجود داشته باشد.
- دانشجو تنها می‌تواند ۶ واحد حذف و ۶ واحد اضافه کند و حداکثر دو درس را می‌تواند حذف و دو درس را اضافه کند.
- تنها دروس مرتبط به رشته را می‌توان برداشت.
- و هر خطایی که بنظر شما منطقی می‌باشد باید پیاده‌سازی شود.

می‌توان یک درس را در انتظار گذاشت و در صورت اینکه یک نفر این درس را حذف کند، دانشجوی در صف انتظار می‌تواند این درس را اخذ کند، برای این کار می‌توانید از ردیس استفاده کنید.

پس از اتمام کار، دانشجو قبل از پایان زمان ترمیم، می‌تواند فرم خود را تایید و به استاد راهنما ارسال کند. پس از آن دیگر حق تغییر دروس ترمیم شده‌ی خود را نخواهد داشت.

- POST /student/{pk/me}/course-substitution/send-form/

استاد نیز با اندپوینت زیر می‌تواند لیست و جزییات ترمیم هر دانشجوی خود را مشاهده کند.

- GET /professor/{pk/me}/students-substitution-forms/

- GET /professor/{pk/me}/students-substitution-forms/{s-pk}/

سپس با API زیر می‌تواند ترمیم را رد و یا تایید کند. در صورت رد کردن استاد، فرم دوباره برای دانشجو تا پایان ترمیم باز می‌شود. تا بتواند تغییرات لازم را اعمال کند و دوباره برای استاد ارسال کند. در صورت تایید نیز دروس به دروس انتخابی در ترم جاری اضافه می‌شوند.

- POST /professor/{pk/me}/students-substitution-forms/{s-pk}/

سپس با تایید استاد، یک ایمیل توسط سلری به دانشجو ارسال می‌شود که استاد این فرم را تایید کرد و برنامه هفتگی دانشجو را برایش ارسال می‌کند.
در صورتی که دانشجو تا پایان ترمیم فرم را ارسال نکند، سلری این فرم را تایید و برای استاد ارسال می‌کند.

h. اندپوینت‌های دروس دانشجو

پس از انتخاب واحد و تایید آن دانشجو می‌تواند دروس برداشته شده در این ترم به اضافه‌ی زمان برگزاری کلاس آن را دریافت کند.

- GET /student/{pk/me}/class-schedule/

همچنین برنامه امتحانات را هم می‌تواند با استفاده از اندپوینت زیر دریافت کند.

- GET /student/{pk/me}/exam-schedule/

تاریخچه‌ی این اندپوینت‌ها نگه داشته می‌شود. به طور مثال اگر یک زمان در کوئری پارامتر ارسال کنیم. سرور باید برنامه زمانی آن زمان را به ما نشان بدهد.

i. اندپوینت‌های حذف اضطراری

دانشجو می‌تواند هر ترم یک درس را قبل از ددلاین حذف، حذف اضطراری کند.

- POST /student/{me,pk}/courses/{c-pk}/emergency-remove/
- GET /student/{me,pk}/courses/{c-pk}/emergency-remove/
- PUT /student/{me,pk}/courses/{c-pk}/emergency-remove/
- DELETE /student/{me,pk}/courses/{c-pk}/emergency-remove/

معاون آموزشی دانشکده می‌تواند این را تایید یا رد کند. در صورت تایید یک ایمیل به دانشجو ارسال می‌شود (توسط سلری) و این را اطلاع می‌دهد و آن درس از دروس ترمی دانشجو حذف می‌شود و در کارنامه وضعیت حذف می‌خورد و نمره‌ای ثبت نمی‌شود. اگر هم رد شود یک ایمیل ارسال می‌شود و دلیل رد را به اطلاع می‌رساند.

- GET /assistant/{me,pk}/emergency-remove/
- GET /assistant/{me,pk}/emergency-remove/{s-pk}/
- POST /assistant/{me,pk}/emergency-remove/{s-pk}/

دو اندپوینت اول لیست و جزییات یک فرم حذف تک درس را نمایش می‌دهد و اندپوینت سوم آن را تایید یا رد می‌کند.

j. اندپوینت‌های حذف ترم

دانشجو می‌تواند یک ترم را حذف کند.

- POST /student/{me,pk}/remove-term/
- PUT /student/{me,pk}/remove-term/
- GET /student/{me,pk}/remove-term/
- DELETE /student/{me,pk}/remove-term/

معاون آموزشی دانشکده می‌تواند این را تایید یا رد کند. در صورت تایید یک ایمیل به دانشجو ارسال می‌شود (توسط سلری) و این را اطلاع می‌دهد و آن ترم و تمام دروس آن حذف می‌شوند. اگر هم رد شود یک ایمیل ارسال می‌شود و دلیل رد را به اطلاع می‌رساند. معاون می‌تواند بدون سنوات یک ترم را حذف کند که به ترم‌های سنواتی یک دانشجو این اضافه نمی‌شود.

- GET /assistant/{me,pk}/remove-term/
- GET /assistant/{me,pk}/remove-term/{s-pk}/
- POST /assistant/{me,pk}/remove-term/{s-pk}/

دو اندپوینت اول لیست و جزییات یک فرم حذف ترم را نمایش می‌دهد و اندپوینت سوم آن را تایید یا رد می‌کند.

k. اندپوینت‌های نمره‌دهی استاد

پس از شروع زمان امتحانات تا پایان ترم، اساتید باید نمرات دانشجویان را وارد کنند.

اساتید یک لیست از دانشجویان و نمره‌ی آن در آن درس توسط اندپوینت زیر ارسال می‌کنند.

- POST /professor/{pk,me}/courses/{c-pk}/scores/

همچنین استاد می‌تواند یک فایل اکسل را که یک ستون آن دانشجو و ستون دیگر آن نمره است را آپلود کند و شما باید نمره را از آن استخراج کنید. (پروتکل و طراحی این فیچر به عهده‌ی تیم است)

پس از نمره دهی استاد وضعیت درس به تایید نشده در کارنامه در می‌آید و دانشجویان حق تقاضای تجدید نظر خواهند داشت.

ا. اندپوینت‌های تقاضای تجدید نظر

دانشجو پس از آنکه نمرات او وارد شد می‌تواند برای یک درس تقاضای تجدید نظر بدهد.

- POST /student/{pk,me}/courses/{c-pk}/appeal-request/
- GET /student/{pk,me}/courses/{c-pk}/appeal-request/
- PUT /student/{pk,me}/courses/{c-pk}/appeal-request/
- DELETE /student/{pk,me}/courses/{c-pk}/appeal-request/

استاد نیز می‌تواند به آن پاسخ دهد و نمره را اصلاح کند.

- GET /professor/{pk,me}/courses/{c-pk}/appeal-requests/
- GET /professor/{pk,me}/courses/{c-pk}/appeal-requests/{pk}/
- POST /professor/{pk,me}/courses/{c-pk}/appeal-requests/{pk}/

در دو اندپوینت اول لیست و جزییات تقاضاهای تجدید نظر را میبیند و در اندپوینت سوم به آن پاسخ می‌دهد و نمره را آپدیت می‌کند. (شما می‌توانید در این API نمره را آپدیت نکنید و از API نمره دهی استفاده کنید)

سپس استاد با اندپوینت زیر نمرات خود را تایید می‌کند و وضعیت به تایید استاد تغییر می‌کند.

- POST /professor/{pk,me}/courses/{c-pk}/approve/

در قسمت بعد هم معاون آموزشی دانشکده دروس تایید شده‌ی اساتید را میبند و تایید می‌زند.

- GET /assistant/{pk,me}/courses/{c-pk}/prof-approved/
- GET /assistant/{pk,me}/courses/{c-pk}/prof-approved/{pk}/
- POST /assistant/{pk,me}/courses/{c-pk}/prof-approved/{pk}/

دو اندپوینت اول لیست و جزییات یک درسی که استاد تایید کرده را نمایش می‌دهد و اندپوینت آخر هم معاون رد یا تایید می‌کند. در صورت رد کردن معاون استاد باید نمرات را مجددا اصلاح و تایید کند و در صورت تایید وضعیت درس برای دانشجو به قبول یا مردود تغییر می‌کند.

m. اندپوینت‌های تقاضای اشتغال به تحصیل پسران

ابتدا دانشجوی پسر یک درخواست ایجاد می‌کند.

- POST /student/{pk,me}/studying-evidence/

- GET /student/{pk,me}/studying-evidence/
- PUT /student/{pk,me}/studying-evidence/
- DELETE /student/{pk,me}/studying-evidence/

سپس معاون آموزشی دانشکده آن را تایید می‌کند و در صورت تایید ایمیلی به دانشجو به همراه pdf گواهی اشتغال به تحصیل ارسال می‌شود (با استفاده از سلری)

- GET /assistant/{pk,me}/studying-evidence/
- GET /assistant/{pk,me}/studying-evidence/{pk}/
- POST /assistant/{pk,me}/studying-evidence/{pk}/

3. قسمت‌های دیگر

a. پنل ادمین

پنل ادمین باید کامل و قابل استفاده باشد. تمام مدل‌ها در آن رجیستر شده باشند و دیزاین آن به عهده‌ی تیم شماست. فیلدهای سرچ و ... باید درست کار کنند. مدیر IT می‌تواند به تمام مدل‌ها دسترسی داشته باشد اما معاون آموزشی به تمام قسمت‌هایی که گفته شده دسترسی دارد. استاد و دانشجو نیز هیچ دسترسی به پنل ادمین ندارند.

مدل دانشجو و استاد را باید بتوان با یک excel در پنل ادمین import کرد تا ادمین به راحتی بتواند دانشجو و استاد بسازد.

از آنجا که URL مدیر admin است پس URL پنل ادمین را باید به دلخواه خودتان تغییر دهید.

b. گیت

شما باید حتما از گیت استفاده کنید و برای هر فیچر یک pull request بزنید و روی هر pr باید یکسری pipline اجرا شود که در ادامه توضیح خواهیم داد. برای کامیت‌ها و برنچ‌ها از یک فرمت خاص و یکپارچه استفاده کنید.

c. ترنسلیشن

باید تمامی رشته‌هایی که شما در بکند خود دارید با استفاده از ترنسلیشن جنگو ترجمه شود و بتوان API را فارسی یا انگلیسی کال کرد.

d. سلری

شما باید یک سلری داشته باشید و هر جا که گفته شد، از آن استفاده کنید. برای broker هم از ردیس یا rabbitmq می‌توانید استفاده کنید.

e. ردیس

برای کش کردن در جاهایی که گفته شده باید استفاده شود و همچنین هر جا که احساس می‌کنید کش کردن لازم است و به پرفورمنس کمک می‌کند از آن استفاده کنید.

f. پکیج منیجر

از خود pip می‌توانید استفاده کنید ولی اگر می‌خواهید حرفه‌ای تر عمل کنید، از poetry یا ابزارهای مشابه هم می‌توانید استفاده کنید.

g. تست

تمام متدها و API ها باید تست شوند! پوشش تست شما نمره‌ی قابل توجهی دارد.

h. اجرای تست و دیپلویمنت

شما باید یک pipeline برای اجرای تست داشته باشید و یک CI/CD برای تست و دیپلوی باید بسازید. از ابزار دلخواه خودتان می‌توانید استفاده کنید ولی پیشنهاد ما GitHub Actions است. پس یعنی هر وقت هر pr ی که زده

می‌شود، باید تست‌ها روی آن اجرا شود و همچنین فرمت آن هم با pep8 چک شود. در صورت پاس شدن تست‌ها می‌توانید برنچ را مرج کنید. پس از مرج هم کانتینرهای شما دوباره بیلد و دیپلوی می‌شوند.

i. داکر

شما باید با استفاده از ابزار داکر تمام کامپوننت‌های خود را داکرایز کنید. یک Dockerfile برای جنگو باید بنویسید و یک docker-compose برای تمامی سرویس‌ها باید داشته باشید.

j. مستندات API ها

تمامی API های شما باید مستند شده باشد. پیشنهاد ما Swagger است و باید بتوان هر API که مدنظر است با داکیومنت شما به راحتی کال شود. می‌توانید از پکیج‌هایی که به صورت اتوماتیک داکیومنت می‌سازند هم استفاده کنید. (مثل django-rest-swagger)

k. ذخیره فایل‌ها

شما باید از دیتابیس MinIO برای ذخیره عکس‌های پروفایل و فایل‌های مدنظرتان استفاده کنید.

l. پروکسی

شما باید از یک proxy که پیشنهاد ما Nginx است استفاده کنید. تنظیمات آن به عهده‌ی خود شماست ولی باید به درستی و در هر شرایط کار کند.

m. متغیرهای محیطی

شما باید متغیرهای محیطیتان (Enviroment Variable) را در یک فایل جدا بریزید و در محیط‌های مختلف (مثل پروداکشن) مقادیر مختلفی به آن بدهید.

n. لاگ

مدیریت خطاها به‌صورت کامل باید انجام شود و از استاندارد RFC 7231 پیروی شود. همچنین در خود سیستم نیز در هنگام خطاها از common log format استفاده باید شود. سیستم باید لاگ‌های خود را در یک فایل ذخیره کند و لاگ‌ها باید در چند سطح باشند. (Log Levels را سرچ کنید)

در پایان این آخرین تمرین شما در این بوتکمپ است. تمام تلاش خود را بکنید که پروژه را کامل به پایان برسانید و تمام نکات آن را یادبگیرید و در کار از آن‌ها استفاده کنید.

موفق و موید باشید :)