2021/22 учебный год

Всероссийская олимпиада студентов «Я – профессионал»

Демонстрационный вариант

задания **полуфинала**

по направлению «Программная инженерия»

Категория участия: «Магистратура/специалитет»

(для поступающих в аспирантуру)

Задание 1. Определение тематики текста (максимум 30 баллов)

Предложить методику определения темы некоторого текста (например, поста).

Требования к структуре оформления решения с указанием критериев оценивания и максимального количества баллов за каждую часть решения:

1. Введение (изложить главную идею решения) - до 1 балла.
2. Основная часть:

2.1. Описание технологии определения темы с кратким описанием методов машинного обучения, используемых для этой цели. В описании использовать одну из процессных нотаций – до 5 баллов.

2.2. Описание алгоритма вычисления ошибки - до 5 баллов.

2.3. Описание программного обеспечения, используемого для автоматизации вышеописанных действий – до 3 баллов.

2.4. Программная реализация решения – до 8 баллов.

2.5. Демонстрация решения на тестовом примере с использованием программной реализации – до 7 баллов.

3. Заключение (выводы) – до 1 балла.

Задание 2. Заметки для магазина (максимум 30 баллов)

Необходимо реализовать RESTFul API сервис работы с товарами и категориями товаров некоторого магазина для панели администратора магазина.

Возможные действия:

Метод	**Endpoint**	**Пояснение**	**Описание**

POST	/products	В запросе передается body в формате JSON:
		{
		“name”: “string”,
		“description”: “string”,
		“category_id”:number
		}
		
		В ответе ожидается новая заметка в формате JSON:
		{
		“id”: number,
		“category_id”: number,
		“name”: “string”,
		“description”: “string”
		}
			Добавление товара с возможностью указания названия (name), описания (description) и идентификатора категории (category_id)
			
			Заголовок - **не** обязательный параметр

GET	/products	В ответе ожидается полный список продуктов в формате JSON:
		[
		{
		“id”: number,
		“category_id”: number,
		“name”: “string”,
		“description”: “string”
		}
		]
			Получение списка всех товаров

GET	/products/${id}	ID запрашиваемого товара как path-параметр: например: **/products/1**
		
		В ответе ожидается конкретный товар в формате JSON:
		{
		“id”: number,
		“category_id”: number,
		“name”: “string”,
		“description”: “string”
		}
			Получение товара по его идентификатору

PUT	/products/${id}	ID товара, который необходимо отредактировать, передается как path-параметр: например:	Редактирование товара по его идентификатору Для редактирования доступны: название, описание, категория

		/products/1
		
		Редактируемые параметры передаются в body запроса в формате JSON:
		
		{
		“id”: number,
		“category_id”: number,
		“name”: “string”,
		“description”: “string”
		}	

GET	/products?query=string	В ответе ожидается список товаров, удовлетворяющих поисковому запросу в формате JSON:
		[
		{
		“id”: number,
		“category_id”: number,
		“name”: “string”,
		“description”: “string”
		}
		]
			Получение списка всех товаров, удовлетворяющих поисковому запросу (т.е. запрос содержится в наименовании (name), описание (description) или НАЗВАНИИ категории

DELETE	/products/${id}	ID товара, который необходимо удалить, передается как path-параметр, например: **/products/1**	Удаление товара по его идентификатору

POST	/categories	В запросе передается body в формате JSON:
		{
		“name”: “string”,
		“description”: “string”,
		“parent_id”:number, nullable
		}
		
		В ответе ожидается новая заметка в формате JSON:
		{
		“id”: number,
		“parent_id”: number,
		“name”: “string”,
		“description”: “string”
		}
			Добавление категории с возможностью указания названия (name), описания (description) и идентификатора категории родителя (parent_id)
			
			Заголовок - **не** обязательный параметр

GET	/categories	В ответе ожидается полный список продуктов в формате JSON: [{	Получение списка всех категорий. Массив подкатегорий содержит объект, аналогичный родителю

		“id”: number, “subCategories”: array, “name”: “string”, “description”: “string” }]	

GET	/categories/${id}	ID запрашиваемой категории как path-параметр: например: **/categories/1**
		
		В ответе ожидается конкретная категория в формате JSON:
		{
		“id”: number,
		“name”: “string”,
		“description”: “string”
		}
			Получение категории по её идентификатору

PUT	/caregories/${id}	ID товара, который необходимо отредактировать, передается как path-параметр: например: **/catrgories/1**
		
		Редактируемые параметры передаются в body запроса в формате JSON:
		
		{
		“id”: number,
		“parent_id”: number,
		“name”: “string”,
		“description”: “string”
		}
			Редактирование категории по его идентификатору
			
			Для редактирования доступны: название, описание, категория родитель

GET	/categories?query=string	В ответе ожидается список категорий, удовлетворяющих поисковому запросу в формате JSON: [{	Получение списка всех категорий, удовлетворяющих поисковому запросу (т.е. запрос содержится в наименовании (name), описание (description)

		“id”: number, “name”: “string”, “description”: “string” }]	
DELETE	/categories/${id}	ID категории, который необходимо удалить, передается как path-параметр, например: **/categories/1**	Удаление категории по её идентификатору. Необходима защита, запрещающая удаление категории при наличии у неё неудаленных подкатегорий

При доступных и адекватных запросах код ответа должен быть равен 200, 201 или 202. При недоступных или неадекватных запросах сервис должен возвращать соответствующие сообщения об ошибках, код ответа должен отличаться от 200, 201 или 202.

Можно использовать любые open source библиотеки.

Проверка будет производиться автоматизированным тестирующим ПО. Необходимо также приложить исходный код в виде архива и выгрузить его в любой git-репозиторий (github, gitlab, bitbucket) с предоставлением публичного всеобщего доступа и приложить ссылку.

Критерии оценки:

1. Проверка автоматизирующим ПО.
2. Описание технического решения (генерация документации на проект в одной из общеиспользуемых спецификаций, рекомендуется Swagger).

Требования к структуре оформления решения с указанием максимального количества баллов за каждую часть решения:

1. Результаты проверки автотестом – до 15 баллов.
2. Описание технического решения:

2.1. Схема решения, описание процессов, сценария использования, описание информационных потоков и предлагаемых технологий в решении (техническое решение) – до 3 баллов.

2.2. Описание алгоритма (алгоритмов) в виде блок-схемы – до 3 баллов.

2.3. Реализация алгоритма (алгоритмов) на любом языке программирования – до 3 баллов.

2.4. Оценка точности и скорости работы алгоритма (алгоритмов) – до 3 баллов.

2.5. Описание документации API в одной из общеиспользуемых спецификаций (рекомендуется Swagger) – до 3 баллов.

Задание 3. Онлайн-сервис городских библиотек (максимум 40 баллов)

Сети городских библиотек для организации онлайн-сервиса выбора книг необходима информационная система, облегчающая сбор, хранение и выдачу заказов читателей.

В основе системы должна лежать реляционная база данных, содержащая информацию, необходимую для ввода и хранения следующих данных:

- Сведений о библиотеках, включая наименование, контактные данные (адрес, телефон, e-mail), ФИО руководителя. Ответственный за ввод информации – пользователь с ролью «Администратор».
- Сведений о книгах, хранящихся в библиотеках, включая наименование, автора, дату выпуска, издательство, категорию («Десткая», «Подростковая», «Взрослая»), тип («Художественная», «Нон-фикшн»), количество книг в каждом статусе («Свободна», «В обработке», «Зарезервирована», «Выдана»). Ответственный за ввод информации – пользователь с ролью «Библиотекарь».
- Сведений о сотрудниках библиотек, включая их паспортные данные, должность, логин и пароль. Ответственный за первоначальный ввод информации – пользователь с ролью «Администратор».
- Сведений о зарегистрированных читателях, включая ФИО, адрес проживания, эл почту, телефон (необязательно), логин и пароль, максимальное количество книг в заказе. Ответственный за ввод информации – пользователь с ролью «Читатель», за исключением максимального количества книг в заказе (по умолчанию 10).
- Сведений о заказах читателей, включая дату заказа, перечень книг в заказе, статус заказа («В обработке», «Готов к выдаче», «Выдан», «Получен в библиотеку»), дату последней смены статуса, дедлайн получения заказа читателем (дата смены статуса на «Готов к выдаче» плюс две недели), дедлайн возврата заказа в библиотеку (дата смены статуса на «Готов к выдаче» плюс три месяца), наименование библиотеки обработки заказа, назначение сотрудника библиотеки на обработку отдельного заказа. Ответственный за ввод информации – пользователь с ролью «Читатель», за исключением полей статуса заказа и ФИО сотрудника библиотеки (ответственный за ввод информации – пользователь с ролью «Библиотекарь»).

Информационная система должна поддерживать вывод следующей информации для пользователей в зависимости от роли:

- Для пользователя с ролью «Читатель» система предоставляет возможность расширенного поиска книг для добавления в заказ по наименованию, автору, году выпуска, издательству, категории и типу книги. Система также дает возможность наблюдать историю заказов, удалять, создавать и редактировать заказы, а также настраивать оповещение о смене статуса заказа по почте.
- Для пользователя с ролью «Библиотекарь» система отображает перечень взятых им в обработку заказов с возможностью фильтрации по имени читателя, статусе заказа, дате заказа. Также система отображает каталог книг с фильтрацией по любому из полей.
- Для пользователя с ролью «Администратор» система отображает по каждому библиотекарю заказы в различном статусе, выполняемые сотрудником в текущий момент времени.

Необходимо:

1) Используя любую общепринятую нотацию, изобразить схему инфологической модели предметной области.
2) Используя любую общепринятую нотацию, изобразить схему даталогической модели базы данных, удовлетворяющую третьей нормальной форме, с выделением первичных и внешних ключей, типа и направления связей.
3) Изобразить прототипы web-страниц работы пользователей с ролями «Администратор», «Читатель» и «Библиотекарь» с системой. Прототипы должны иллюстрировать возможности ввода и вывода доступной пользователям информации.
4) Описать алгоритм снижения максимального количества книг в заказе для читателя до 7 книг – в случае возврата одного любого заказа в библиотеку после расчетного срока, до 4 книг – в случае возврата двух заказов в библиотеку после расчетного срока, до 1 книги – в случае возврата трех и более заказов в библиотеку после расчетного срока.
5) Описать алгоритм расчета самого раннего времени возврата заданной книги в библиотеку в случае ее выдачи в нескольких заказах читателям.
6) Используя операторы языка SQL, написать запрос для вывода списка сотрудников с указанием места работы (наименования библиотеки), у которых за последний месяц было не менее 10 заказов.

Требования к структуре оформления решения с указанием критериев оценивания и максимального количества баллов за каждую часть решения:

1. Введение - до 2 баллов.
2. Основная часть:

2.1. Схема решения, описание процессов, описание инфологической модели предметной области – до 4 баллов.

2.2. Структура базы данных, отражающая специфику предметной области. БД должна соответствовать третьей нормальной форме, не быть перегруженной дублированием, учитывать сущности-справочники, не иметь ошибок по связям – до 6 баллов.

2.3. Прототипы визуальных интерфейсов – до 5 баллов.

2.4. Описание алгоритмов (в виде схемы алгоритмов, программного кода или псевдокода) – до 7 баллов.

2.5. Описание SQL-запроса – до 4 баллов.

2.6. Программная реализация решения – до 7 баллов.

2.7. Демонстрация решения на тестовом примере с использованием программной реализации – до 3 баллов.

3. Заключение (выводы) – до 2 баллов.