# Основы разработки информационных систем 2 курс 1 семестр - Требования по семестровой работе № 2

**Процесс сдачи работы и баллы**

Работа выполняется индивидуально. Сдать можно на паре или в дополнительное время (обсудить с преподавателем). Не тяните, распределяйте время, чтобы не было столпотворений во время сдачи!

На сдаче нужно показать код, продемонстрировать работу основного функционала, ответить на вопросы, связанные с программой.

При проверке будет оцениваться соответствие всем нижеперечисленным требованиям, реализация правильной архитектуры, умение применять используемые технологии, удобство пользования системой, защищенность системы.

**Идея сайта**

У сайта должна быть идея. Это не значит, что нужен функционал, которого нет ни на одном другом сайте. Можно взять функционал существующего портала и доработать его. Если сомневаетесь в сложности и новизне проекта, лучше уточните.

Примеры: не просто блог, а с прикреплением музыки (мультимедиа) к постам. Не просто магазин, а со сверкой цен с другим сайтом. Системы для записи на приём/регистрации в отеле. Торговые площадки-аукционы типа ebay.

**Основной фреймворк Flask**

Допускается использование вспомогательных библиотек, но не кода, который напрямую решает задачи бизнес-логики. При использовании любой библиотеки нужно хорошо понимать, для чего она, какие у неё возможности и логика использования.

**Интерфейс и дизайн**

Кратко: он должен быть удобным :D Без усложнений.

Необходимо использовать HTML5, CSS3. CSS-препроцессоры не запрещаются (LESS, SASS), но нужно знать их основные принципы. Если используете фреймворки (Bootstrap и прочие), должны уметь пояснить, что за классы используете, особенно классы сетки (grid).

Хороший пример: не перегруженный кнопками и цветами интерфейс, текст в текстовых полях, числа для чисел и т.п.

Плохой пример: выбор цвета в виде ввода текста, а не типом в теге input.

**Формы**

Необходимо корректно работать с формами: проверка данных на сервере, защита от повторной отправки там, где это нужно. Стилизация полей формы по вашему усмотрению. Нельзя использовать WTForms.

**Страницы**

* Нужно использовать шаблонизатор Jinja.
* В коде шаблонов нельзя напрямую вызывать Python-код (macro). Можно (нужно) типовые if/for/block/extends/двойные фигурные скобки.
* Должна быть развитая система шаблонов:
  + Базовый шаблон, содержащий заголовки, общее меню, подгруженные стили/скрипты.
  + Страницы находятся в папках сущностей. Т.е. для сущности Product существует папка, где хранятся страницы, которые работают только с этой сущностью.
* Как минимум, для одной из сущностей должен быть реализован полностью CRUD с интерфейсом, но не обязательно в виде админки.

**Авторизация/аутентификация**

* Сайт должен иметь самописную аутентификацию, авторизацию, регистрацию.
* Самописность предполагает создание куки/сессии.
* Использовать готовые flask\_login (LoginManager, UserMixin и т.п.) запрещено
* Роль на авторство: к примеру автор поста/товара может удалить/редактировать свою запись, а другой пользователь - нет.

**База данных**

* Сайт должен хранить данные в СУБД: sqlite3, MySQL, MSSQL, MariaDB или PostgreSQL
* Создание таблиц происходит путем выполнения sql-скрипта/запросами
* Запрещено использовать готовую реализацию ORM **(flask-sqlalchemy)**. Самописную можно, для удобной обработки сырых запросов. Но не обязательно писать свой ORM.
* Система должна работать, как минимум с тремя сущностями. Должны быть связи M2M (many-to-many), O2M (one-to-many), O2O (one-to-one).

**Система**

* Система должна быть построена по модели MVC. На самом деле Flask не реализует эту модель, это больше обертка для WSGI, но постараться разграничить модели, отображения и обработчики. Т.е. клиент через запросы вызывает обработчики, которые меняют модели, которые меняют отображение :)
* Необходимо придерживаться принципов SOLID. Весь код должен быть корректно поделен на пакеты. Крайне желательно выделить сервисы бизнес-логики.

**Javascript**

Необходимо применить немного JavaScript (можно jQuery). Обдуманное использование JS для скрытия объектов, перемещения и т.п., AJAX не обязательно, но положительно повлияет на оценку.

**Дополнительное задание**

Реализовать аутентификацию через oAuth-авторизацию. При этом все экшены, репозитории и прочее нужно прописать самим. Из готового кода, который можно использовать - это только генерация специфических токенов для конкретного oAuth-сервиса.

**Разбалловка**

Вариант суммы: 10 баллов

* Вёрстка, дизайн, шаблоны - 2
* Работа с БД и сущностями - 2
* Архитектура приложения - 2
* Проверки, вывод, обработка запроса, бизнес-логика - 2
* Отсутствие багов, комментарии к коду (падение сервера при использовании сайта, необработанные исключения) - 2