Приложение

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный аграрный университет имени П.А. Столыпина»

Университетский колледж агробизнеса

09.02.11 Разработка и управление программным обеспечением

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
по ПП.05.01 Производственная практика (по профилю специальности)

Обеспечивающее преподавание дисциплины подразделение	Инженерное отделение
Разработчик:	
Преподаватель	А.В. Кортусов

Омск
2026

СОДЕРЖАНИЕ

1. ОБЩИЕ ПОЛОЖЕНИЯ

1. Фонд оценочных средств (далее – ФОС) предназначен для контроля и оценки образовательных достижений обучающихся, освоивших программу ПП 05.02 Производственная практика.
2. ФОС включает оценочные материалы для проведения текущего контроля и промежуточной аттестации в форме зачета.
3.ФОС позволяет оценивать знания, умения, направленные на формирование компетенций
4. ФОС разработан на основании положений основной образовательной программы по специальности 09.02.11 Разработка и управление программным обеспечением и рабочей программы ПП 05.02 Производственная практика.
5. ФОС является обязательным обособленным приложением к рабочей программе.

2. ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ ИЗУЧЕНИЯ

Результаты обучения (освоенные навыки, умения)	**Показатели оценки образовательных результатов**
создания таблиц базы данных с определением структуры и типов данных для каждого атрибута	Обучающийся владеет навыками создания таблиц базы данных с определением структуры и типов данных для каждого атрибута
создания и удаления баз данных, восстановления баз данных, резервного копирования баз данных	Обучающийся владеет навыками создания и удаления баз данных, восстановления баз данных, резервного копирования баз данных
создания пользователей и назначения прав доступа	Обучающийся владеет навыками создания пользователей и назначения прав доступа
использования стандартных методов защиты объектов базы данных	Обучающийся владеет навыками использования стандартных методов защиты объектов базы данных
разработки и внедрения систем защиты баз данных от несанкционированного доступа разработки и внедрения систем резервного копирования и восстановления баз данных аудита безопасности баз данных	Обучающийся владеет навыками разработки и внедрения систем защиты баз данных от несанкционированного доступа разработки и внедрения систем резервного копирования и восстановления баз данных аудита безопасности баз данных
тестирования информационной системы	Обучающийся владеет навыками тестирования информационной системы
оптимизировать запросы и проводить мониторинг производительности базы данных	Обучающийся умеет оптимизировать запросы и проводить мониторинг производительности базы данных
создавать и удалять базы данных	Обучающийся умеет создавать и удалять базы данных
создавать пользователей и назначать права доступа	Обучающийся умеет создавать пользователей и назначать права доступа
обеспечивать безопасность и управлять доступом к данным	Обучающийся умеет обеспечивать безопасность и управлять доступом к данным
разрабатывать и внедрять системы защиты баз данных от несанкционированного доступа	Обучающийся умеет разрабатывать и внедрять системы защиты баз данных от несанкционированного доступа
разрабатывать и внедрять системы резервного копирования и восстановления баз данных	Обучающийся умеет разрабатывать и внедрять системы резервного копирования и восстановления баз данных
устанавливать и настраивать механизмы аутентификации и авторизации пользователей, создавать и управлять ролями и правами доступа к данным	Обучающийся умеет устанавливать и настраивать механизмы аутентификации и авторизации пользователей, создавать и управлять ролями и правами доступа к данным
осуществлять тестирование информационной системы	Обучающийся умеет осуществлять тестирование информационной системы

3. РАСПРЕДЕЛЕНИЕ ОЦЕНИВАНИЯ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ И ТИПОВ ОЦЕНОЧНЫХ МАТЕРИАЛОВ ПО ЭЛЕМЕНТАМ ЗНАНИЙ И УМЕНИЙ

Содержание курса	**Форма контроля**	**Умения**	**Навыки**
Текущий контроль			
Раздел 1. Организационный этап			
Прохождение вводного инструктажа. Получение и обсуждение задания на практику	Проверка отчета по учебной практике	Уо 02.01, Уо 02.02, Уо 07.01, Уо 09.01	-
Раздел 2. Основной этап			
Выполнение работ.	наблюдение и оценка в процессе практики; анализ отчетной документации; экспертная оценка выполнения индивидуальных заданий.	У 1.3.1, У 1.4.1, У 1.4.2, У 1.4.3, У 1.5.1, У 1.5.2, У 1.5.3, У 3.6.1, Уо 02.01, Уо 02.02, Уо 07.01, Уо 09.01	Н 1.3.1, Н 1.4.1, Н 1.4.2, Н 1.5.1, Н 1.5.2, Н 3.6.1
Раздел 3. Заключительный этап			
Оформление введения. Оформление основной части. Оформление заключения. Оформление списка использованных источников и приложений. Оформление отчета и приложений. Прохождение собеседования (зачет)	Проверка отчета по учебной практике, собеседование	Уо 02.01, Уо 02.02, Уо 07.01, Уо 09.01	-
Промежуточный контроль			
Зачет	Проверка отчета по учебной практике, собеседование	У 1.3.1, У 1.4.1, У 1.4.2, У 1.4.3, У 1.5.1, У 1.5.2, У 1.5.3, У 3.6.1, Уо 02.01, Уо 02.02, Уо 07.01, Уо 09.01	Н 1.3.1, Н 1.4.1, Н 1.4.2, Н 1.5.1, Н 1.5.2, Н 3.6.1

4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОЦЕНКИ УМЕНИЙ, НАВЫКОВ

4.1. Оценочные средства, применяемые для текущего контроля.

4.1.1 Разработайте динамическое веб-приложение с использованием Flask:

1. Создайте серверную часть с маршрутизацией.
2. Реализуйте работу с базой данных (SQLite/MySQL).
3. Разработайте шаблоны с использованием Jinja2.
4. Реализуйте аутентификацию и авторизацию пользователей.
5. Создайте административную панель.

Пример кода:

```
# app.py
from flask import Flask, render_template, request, redirect, url_for, session
import sqlite3
from werkzeug.security import generate_password_hash, check_password_hash
from functools import wraps

app = Flask(__name__)
app.secret_key = 'your-secret-key'

def get_db():
    conn = sqlite3.connect('blog.db')
    conn.row_factory = sqlite3.Row
    return conn

def init_db():
    with get_db() as conn:
        conn.execute('''
            CREATE TABLE IF NOT EXISTS users (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                username TEXT UNIQUE NOT NULL,
                password TEXT NOT NULL,
                role TEXT DEFAULT 'user'
            )
        ''')
        conn.execute('''
            CREATE TABLE IF NOT EXISTS posts (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                title TEXT NOT NULL,
                content TEXT NOT NULL,
                author_id INTEGER,
                created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                FOREIGN KEY (author_id) REFERENCES users(id)
            )
        ''')

def login_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
```

```
        if 'user_id' not in session:
            return redirect(url_for('login'))
        return f(*args, **kwargs)
    return decorated

def admin_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        if session.get('role') != 'admin':
            return 'Доступ запрещен', 403
        return f(*args, **kwargs)
    return decorated

@app.route('/')
def index():
    with get_db() as conn:
        posts = conn.execute('''
            SELECT posts.*, users.username
            FROM posts
            JOIN users ON posts.author_id = users.id
            ORDER BY created_at DESC
        ''').fetchall()
    return render_template('index.html', posts=posts)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = generate_password_hash(request.form['password'])

        with get_db() as conn:
            try:
                conn.execute('INSERT INTO users (username, password) VALUES (?, ?)',
                             (username, password))
                conn.commit()
                return redirect(url_for('login'))
            except sqlite3.IntegrityError:
                return 'Пользователь уже существует'

    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        with get_db() as conn:
            user = conn.execute('SELECT * FROM users WHERE username = ?',
                                (username,)).fetchone()
```

```
        if user and check_password_hash(user['password'], password):
            session['user_id'] = user['id']
            session['username'] = user['username']
            session['role'] = user['role']
            return redirect(url_for('index'))

        return 'Неверные данные'

    return render_template('login.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('index'))

@app.route('/posts/new', methods=['GET', 'POST'])
@login_required
def new_post():
    if request.method == 'POST':
        title = request.form['title']
        content = request.form['content']

        with get_db() as conn:
            conn.execute('''
                INSERT INTO posts (title, content, author_id)
                VALUES (?, ?, ?)
            ''', (title, content, session['user_id']))
            conn.commit()

        return redirect(url_for('index'))

    return render_template('new_post.html')

@app.route('/api/posts', methods=['GET'])
def api_posts():
    with get_db() as conn:
        posts = conn.execute('SELECT * FROM posts').fetchall()
    return [dict(post) for post in posts]

if __name__ == '__main__':
    init_db()
    app.run(debug=True)
html
<!-- templates/index.html -->
<!DOCTYPE html>
<html>
<head>
    <title>Блог</title>
    <style>
        * { margin: 0; padding: 0; box-sizing: border-box; }
        body { font-family: Arial, sans-serif; background: #f4f4f4; }
```

```
        .container { max-width: 800px; margin: 0 auto; padding: 20px; }
        header { background: #333; color: white; padding: 20px; }
        nav a { color: white; text-decoration: none; margin-right: 20px; }
        .post { background: white; padding: 20px; margin-bottom: 20px; border-radius: 5px; }
        .post h2 { margin-bottom: 10px; }
        .post .meta { color: #666; font-size: 14px; }
        .btn { display: inline-block; padding: 8px 20px; background: #333; color: white;
               text-decoration: none; border-radius: 5px; }
        .btn-primary { background: #007bff; }
        .flash { padding: 10px; margin: 10px 0; border-radius: 5px; }
        .flash-success { background: #d4edda; color: #155724; }
        .flash-error { background: #f8d7da; color: #721c24; }
    </style>
</head>
<body>
    <header>
        <div class="container">
            <h1>Мой блог</h1>
            <nav>
                <a href="{{ url_for('index') }}">Главная</a>
                <a href="{{ url_for('new_post') }}">Создать пост</a>
                {% if session.user_id %}
                    <span>Привет, {{ session.username }}</span>
                    <a href="{{ url_for('logout') }}">Выход</a>
                {% else %}
                    <a href="{{ url_for('login') }}">Вход</a>
                    <a href="{{ url_for('register') }}">Регистрация</a>
                {% endif %}
            </nav>
        </div>
    </header>

    <div class="container">
        {% for post in posts %}
        <div class="post">
            <h2>{{ post.title }}</h2>
            <p>{{ post.content[:200] }}...</p>
            <div class="meta">Автор: {{ post.username }} | {{ post.created_at }}</div>
        </div>
        {% endfor %}
    </div>

    <script>
        // Загрузка постов через API
        fetch('/api/posts')
            .then(response => response.json())
            .then(data => console.log('API Posts:', data));
    </script>
</body>
</html>
```

Форма выполнения: Исходный код, скриншоты работы, демонстрация функционала.

4.1.2. Интеграция с API и разработка мультимедийного контента

Разработайте одностраничное приложение с интеграцией API и мультимедийным контентом:

1. Интеграция с REST API (получение данных о погоде, новостях и т.д.).
2. Создание интерактивных графиков с использованием Chart.js.
3. Реализация поиска и фильтрации контента.
4. Разработка мультимедийной галереи с видеоплеером.

Пример кода:

```
// js/app.js - Single Page Application

// Получение данных о погоде
async function getWeather(city) {
    const apiKey = 'your-api-key';
    const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

    try {
        const response = await fetch(url);
        const data = await response.json();
        displayWeather(data);
    } catch (error) {
        console.error('Ошибка:', error);
    }
}

function displayWeather(data) {
    document.getElementById('weather').innerHTML = `
        <h3>Погода в ${data.name}</h3>
        <p>Температура: ${data.main.temp}°C</p>
        <p>Влажность: ${data.main.humidity}%</p>
        <p>Ветер: ${data.wind.speed} м/с</p>
        <img src="https://openweathermap.org/img/wn/${data.weather[0].icon}@2x.png"
             alt="Иконка погоды">
    `;
}

// Создание графика с Chart.js
function initChart() {
    const ctx = document.getElementById('myChart').getContext('2d');
    new Chart(ctx, {
        type: 'bar',
        data: {
            labels: ['Янв', 'Фев', 'Мар', 'Апр', 'Май', 'Июн'],
            datasets: [{
                label: 'Посещения',
                data: [120, 190, 300, 500, 200, 300],
                backgroundColor: 'rgba(54, 162, 235, 0.2)',
                borderColor: 'rgba(54, 162, 235, 1)',
```

```
            borderWidth: 1
        }]
    },
    options: {
        responsive: true,
        plugins: {
            title: {
                display: true,
                text: 'Статистика посещений'
            }
        }
    }
});
}

// Поиск и фильтрация
function initSearch() {
    const searchInput = document.getElementById('searchInput');
    const items = document.querySelectorAll('.list-item');

    searchInput.addEventListener('input', function() {
        const query = this.value.toLowerCase();
        items.forEach(item => {
            const text = item.textContent.toLowerCase();
            item.style.display = text.includes(query) ? 'block' : 'none';
        });
    });
}

// Инициализация
document.addEventListener('DOMContentLoaded', () => {
    getWeather('Moscow');
    initChart();
    initSearch();
});
```

Форма выполнения: Исходный код, демонстрация работы с API, скриншоты.

4.1.3. Развертывание и тестирование веб-приложения

Выполните развертывание и тестирование веб-приложения:

1. Настройка сервера для production-окружения.
2. Настройка базы данных и подключение.
3. Сборка статических файлов.
4. Тестирование производительности.
5. Документирование процесса развертывания.

Пример конфигурации:

```
# config.py
import os

class Config:
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'dev-key'
```

```
    DATABASE_URL = os.environ.get('DATABASE_URL') or 'sqlite:///app.db'
    DEBUG = False

class DevelopmentConfig(Config):
    DEBUG = True
    DATABASE_URL = 'sqlite:///dev.db'

class ProductionConfig(Config):
    DEBUG = False
    # Настройки для production

# gunicorn.conf.py
bind = "0.0.0.0:8000"
workers = 4
worker_class = "sync"
accesslog = "/var/log/app/access.log"
errorlog = "/var/log/app/error.log"
loglevel = "info"
bash
# deploy.sh
#!/bin/bash
# Скрипт развертывания

echo "Начало развертывания..."

# Установка зависимостей
pip install -r requirements.txt

# Инициализация базы данных
python init_db.py

# Сборка статики
npm run build

# Перезапуск сервера
sudo systemctl restart myapp

echo "Развертывание завершено!"
```

Форма выполнения: Конфигурационные файлы, инструкция по развертыванию, скриншоты.

4.2. Оценочные средства, применяемые для промежуточной аттестации по итогам изучения дисциплины

Зачет проводится по завершении производственной практики.

Промежуточная аттестация в форме зачета осуществляется по результатам сдачи отчета по практике и с учетом текущего контроля успеваемости при выполнении всех видов текущего контроля.

Обучающиеся, не выполнившие виды работ, предусмотренные рабочей программой; пропустившие более 50% практики без уважительной причины, не допускаются к зачету.

Промежуточная аттестация таких лиц проводится только после прохождения ими всех видов текущего контроля.

V. ПОКАЗАТЕЛИ ОЦЕНКИ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ

Уровень сформиро-ванности компетенций	**Оценка**	**Критерии оценивания по видам работ**	
		тестирование (процент правильных ответов)	**прочие виды работ по практике**
Высокий	Отлично	90-100%	Обучающийся глубоко и прочно усвоил теоретический и освоил практический материал. Дает логичные и грамотные ответы. Демонстрирует знание не только основного, но и дополнительного материала, быстро ориентируется, отвечая на дополнительные вопросы. Свободно справляется с поставленными задачами, аргументировано и верно обосновывает принятые решения.
Повышенный	Хорошо	70-89%	Обучающийся твердо знает программный материал, грамотно и по существу излагает его. Не допускает существенных неточностей при ответах на вопросы, правильно применяет теоретические положения при решении практических задач, владеет навыками и приемами их выполнения.
Базовый	Удовлетворительно	50-69%	Обучающийся демонстрирует знания только основного материала, но не усвоил его детали, испытывает затруднения при решении практических задач. В ответах на поставленные вопросы допускает неточности. Дает определения понятий, неискажающие их смысл. Нарушает последовательность изложения программного материала.
Не сформирована	Неудовлетворительно	0-49%	Обучающийся не знает, не выполняет или неправильно выполняет большую часть учебного материала. Допускает ошибки в формулировке определений, искажающие их смысл, беспорядочно и неуверенно излагает материал. Ответы на дополнительные вопросы отсутствуют. Не выполняет задания.

ЛИСТ РАССМОТРЕНИЙ И ОДОБРЕНИЙ

рабочей программы

ПП 05.02 Производственная практика (по профилю специальности)

в составе ООП **09.02.11 Разработка и управление программным обеспечением**

1)Рассмотрена и одобрена:
а)На заседании предметно-цикловой методической комиссии протокол № 6 от 10.03.2026 г. Председатель ПЦМК _____ С.М. Нурбаева
б) На заседании методического совета протокол №4 от 21.04.2026 г. Председатель методического совета _____ М.В. Иваницкая
2)Рассмотрена и одобрена внешним экспертом
а) директор ООО «Сатори Партнер» А.Б. Мальцев