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 УП.03.01 Учебная практика

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

Омск
2026

СОДЕРЖАНИЕ

1. ОБЩИЕ ПОЛОЖЕНИЯ 3
2. ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ ИЗУЧЕНИЯ 4
3. РАСПРЕДЕЛЕНИЕ ОЦЕНИВАНИЯ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ И ТИПОВ ОЦЕНОЧНЫХ МАТЕРИАЛОВ ПО ЭЛЕМЕНТАМ ЗНАНИЙ И УМЕНИЙ 6
4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ 7
5. ПОКАЗАТЕЛИ ОЦЕНКИ РЕЗУЛЬТАТОВ ОСВОЕНИЯ УЧЕБНОЙ ПРАКТИКИ 15

1. ОБЩИЕ ПОЛОЖЕНИЯ

1. Фонд оценочных средств (далее – ФОС) предназначен для контроля и оценки образовательных достижений обучающихся, освоивших программу УП 03.01 Учебная 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 УП 03.01 Учебная практика
5. ФОС является обязательным обособленным приложением к рабочей программе.

2. ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ ИЗУЧЕНИЯ

Результаты обучения (освоенные навыки, умения)	Показатели оценки образовательных результатов
анализа деятельности организации	Обучающийся владеет навыками анализа деятельности организации
выявления потребностей в автоматизации	Обучающийся владеет навыками выявления потребностей в автоматизации
составления описания требований	Обучающийся владеет навыками составления описания требований
разработки общей схемы работы системы	Обучающийся владеет навыками разработки общей схемы работы системы
определения состава и связей частей	Обучающийся владеет навыками определения состава и связей частей
проектирования экранных форм и отчетов	Обучающийся владеет навыками проектирования экранных форм и отчетов
описания правил обработки информации	Обучающийся владеет навыками описания правил обработки информации
подбора технических средств	Обучающийся владеет навыками подбора технических средств
тестирования информационной системы	Обучающийся владеет навыками тестирования информационной системы
подготовки документов для разработчиков	Обучающийся владеет навыками подготовки документов для разработчиков
согласования решений с заказчиком	Обучающийся владеет навыками согласования решений с заказчиком
оценки сроков и стоимости создания	Обучающийся владеет навыками оценки сроков и стоимости создания
учета требований безопасности	Обучающийся владеет навыками учета требований безопасности
проводить изучение деятельности организации	Обучающийся умеет проводить изучение деятельности организации
выявлять потребности в автоматизации	Обучающийся умеет выявлять потребности в автоматизации
составлять описание требований к будущей системе	Обучающийся умеет составлять описание требований к будущей системе
разрабатывать общую схему работы системы	Обучающийся умеет разрабатывать общую схему работы системы
определять состав и взаимосвязи частей системы	Обучающийся умеет определять состав и взаимосвязи частей системы
проектировать экранные формы и отчеты	Обучающийся умеет проектировать экранные формы и отчеты
описывать правила обработки информации	Обучающийся умеет описывать правила обработки информации
определять необходимый состав технических средств	Обучающийся умеет определять необходимый состав технических средств
осуществлять тестирование информационной системы	Обучающийся умеет осуществлять тестирование информационной системы

готовить документы для разработчиков	Обучающийся умеет готовить документы для разработчиков
согласовывать проектные решения с заказчиком	Обучающийся умеет согласовывать проектные решения с заказчиком
оценивать сроки и стоимость создания системы	Обучающийся умеет оценивать сроки и стоимость создания системы
учитывать требования безопасности информации	Обучающийся умеет учитывать требования безопасности информации

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

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 05.01	-
Раздел 2. Основной этап			
Решение прикладных задач.	наблюдение и оценка в процессе практики; анализ отчетной документации; экспертная оценка выполнения индивидуальных заданий.	У 3.1.1, У 3.1.2, У 3.2.1, У 3.2.2, У 3.3.1, У 3.4.1, У 3.4.2, У 3.5.1, У 3.6.1, У 3.7.1, У 3.7.2, У 3.8.1, У 3.8.2	Н 3.1.1, Н 3.1.2, Н 3.2.1, Н 3.2.2, Н 3.3.1, Н 3.4.1, Н 3.4.2, Н 3.5.1, Н 3.6.1, Н 3.7.1, Н 3.7.2, Н 3.8.1, Н 3.8.2, Н 3.8.3
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 05.01	-
Промежуточный контроль			
Зачет	Проверка отчета по учебной практике, собеседование	У 3.1.1, У 3.1.2, У 3.2.1, У 3.2.2, У 3.3.1, У 3.4.1, У 3.4.2, У 3.5.1, У 3.6.1, У 3.7.1, У 3.7.2, У 3.8.1, У 3.8.2, Уо 02.01, Уо 02.02, Уо 05.01	Н 3.1.1, Н 3.1.2, Н 3.2.1, Н 3.2.2, Н 3.3.1, Н 3.4.1, Н 3.4.2, Н 3.5.1, Н 3.6.1, Н 3.7.1, Н 3.7.2, Н 3.8.1, Н 3.8.2, Н 3.8.3

4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОЦЕНКИ УМЕНИЙ, НАВЫКОВ

4.1. Оценочные средства, применяемые для текущего контроля.

4.1.1. Проектирование структуры информационной системы

На основе технического задания разработайте структуру информационной системы для учета книг в библиотеке:

1. Проведите анализ предметной области.
2. Разработайте ER-диаграмму в HeidiSQL (не менее 4 связанных таблиц).
3. Создайте SQL-скрипты для создания базы данных, таблиц и связей.
4. Заполните таблицы тестовыми данными (не менее 10 записей в каждой таблице).

Структура базы данных:

Таблица	Поля
Книги	id (PK), название, автор, год_издания, жанр, количество_экземпляров
Читатели	id (PK), ФИО, дата_рождения, телефон, email, адрес
Выдача	id (PK), id_книги (FK), id_читателя (FK), дата_выдачи, дата_возврата, статус
Жанры	id (PK), название

Пример SQL-скрипта в HeidiSQL:

```
CREATE DATABASE библиотека;
USE библиотека;

CREATE TABLE Жанры (
    id INT PRIMARY KEY AUTO_INCREMENT,
    название VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE Книги (
    id INT PRIMARY KEY AUTO_INCREMENT,
    название VARCHAR(200) NOT NULL,
    автор VARCHAR(100) NOT NULL,
    год_издания INT,
    id_жанра INT,
    количество_экземпляров INT DEFAULT 1,
    FOREIGN KEY (id_жанра) REFERENCES Жанры(id)
```

```
);

CREATE TABLE Читатели (
    id INT PRIMARY KEY AUTO_INCREMENT,
    ФИО VARCHAR(100) NOT NULL,
    дата_рождения DATE,
    телефон VARCHAR(20),
    email VARCHAR(100),
    адрес VARCHAR(200)
);

CREATE TABLE Выдача (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_книги INT NOT NULL,
    id_читателя INT NOT NULL,
    дата_выдачи DATE NOT NULL,
    дата_возврата DATE,
    статус ENUM('выдана', 'возвращена', 'просрочена') DEFAULT 'выдана',
    FOREIGN KEY (id_книги) REFERENCES Книги(id),
    FOREIGN KEY (id_читателя) REFERENCES Читатели(id)
);
```

Форма выполнения: ER-диаграмма в HeidiSQL, SQL-скрипты создания БД и заполнения данными.

4.1.2. Разработка серверной части на Python (Flask)

Разработайте серверную часть информационной системы на Flask:

1. Настройте подключение к базе данных через PyMySQL.
2. Реализуйте CRUD-операции для всех таблиц.
3. Создайте API-эндпоинты для работы с данными.
4. Реализуйте валидацию входных данных.

Пример кода:

```
# app.py
from flask import Flask, request, jsonify
import pymysql
import pymysql.cursors

app = Flask(__name__)

def get_db_connection():
    return pymysql.connect(
        host='localhost',
        user='root',
        password='',
        database='библиотека',
        cursorclass=pymysql.cursors.DictCursor,
        charset='utf8mb4'
    )

# Получение всех книг
@app.route('/api/books', methods=['GET'])
```

```
def get_books():
    conn = get_db_connection()
    try:
        with conn.cursor() as cursor:
            cursor.execute('''
                SELECT Книги.*, Жанры.название as жанр
                FROM Книги
                LEFT JOIN Жанры ON Книги.id_жанра = Жанры.id
            ''')
            books = cursor.fetchall()
            return jsonify(books)
    finally:
        conn.close()

# Добавление книги
@app.route('/api/books', methods=['POST'])
def add_book():
    data = request.json
    required = ['название', 'автор']
    if not all(field in data for field in required):
        return jsonify({'error': 'Не все поля заполнены'}), 400

    conn = get_db_connection()
    try:
        with conn.cursor() as cursor:
            cursor.execute('''
                INSERT INTO Книги (название, автор, год_издания, id_жанра,
количество_экземпляров)
                VALUES (%s, %s, %s, %s, %s)
            ''', (data['название'], data['автор'], data.get('год_издания'),
                  data.get('id_жанра'), data.get('количество_экземпляров', 1)))
            conn.commit()
            return jsonify({'id': cursor.lastrowid, 'message': 'Книга добавлена'}), 201
    finally:
        conn.close()

# Выдача книги
@app.route('/api/loans', methods=['POST'])
def create_loan():
    data = request.json
    if 'id_книги' not in data or 'id_читателя' not in data:
        return jsonify({'error': 'Укажите книгу и читателя'}), 400

    conn = get_db_connection()
    try:
        with conn.cursor() as cursor:
            # Проверка наличия книги
            cursor.execute('SELECT количество_экземпляров FROM Книги WHERE id = %s',
(data['id_книги'],))
            book = cursor.fetchone()
            if not book or book['количество_экземпляров'] <= 0:
```

```
            return jsonify({'error': 'Книги нет в наличии'}), 400

        # Проверка читателя
        cursor.execute('SELECT id FROM Читатели WHERE id = %s', (data['id_читателя'],))
        if not cursor.fetchone():
            return jsonify({'error': 'Читатель не найден'}), 400

        # Создание записи о выдаче
        cursor.execute('''
            INSERT INTO Выдача (id_книги, id_читателя, дата_выдачи, статус)
            VALUES (%s, %s, CURDATE(), 'выдана')
        ''', (data['id_книги'], data['id_читателя']))

        # Уменьшение количества экземпляров
        cursor.execute('UPDATE Книги SET количество_экземпляров = количество_экземпляров - 1 WHERE id = %s',
                       (data['id_книги'],))
        conn.commit()
        return jsonify({'message': 'Книга выдана'}), 201
    finally:
        conn.close()

if __name__ == '__main__':
    app.run(debug=True)
```

Форма выполнения: Исходный код серверной части, скриншоты работы API через Postman.

4.1.3. Тестирование и отладка информационной системы

Выполните тестирование разработанной системы:

1. Разработайте тестовые сценарии для всех API-эндпоинтов.
2. Протестируйте корректные и некорректные запросы.
3. Выявите и исправьте ошибки.

Пример тестов:

```
# test_api.py
import pytest
import json
from app import app

@pytest.fixture
def client():
    app.config['TESTING'] = True
    with app.test_client() as client:
        yield client

def test_get_books(client):
    """Тест получения списка книг"""
    response = client.get('/api/books')
    assert response.status_code == 200
    data = json.loads(response.data)
    assert isinstance(data, list)
```

```
def test_add_book(client):
    """Тест добавления книги"""
    response = client.post('/api/books', json={
        'название': 'Война и мир',
        'автор': 'Лев Толстой',
        'год_издания': 1869,
        'количество_экземпляров': 3
    })
    assert response.status_code == 201
    data = json.loads(response.data)
    assert 'id' in data

def test_add_book_invalid(client):
    """Тест добавления книги без обязательных полей"""
    response = client.post('/api/books', json={'название': 'Тест'})
    assert response.status_code == 400

def test_create_loan(client):
    """Тест выдачи книги"""
    # Сначала добавляем книгу
    add_resp = client.post('/api/books', json={
        'название': 'Тестовая книга',
        'автор': 'Тест',
        'количество_экземпляров': 1
    })
    book_id = json.loads(add_resp.data)['id']

    response = client.post('/api/loans', json={
        'id_книги': book_id,
        'id_читателя': 1
    })
    assert response.status_code == 201
```

Форма выполнения: Код тестов, отчет о результатах тестирования.

4.1.4. Разработка пользовательского интерфейса

Разработайте простой веб-интерфейс для информационной системы:

```
<!-- templates/index.html -->
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Библиотека</title>
    <style>
        body { font-family: Arial; margin: 20px; }
        table { border-collapse: collapse; width: 100%; }
        th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
        th { background: #4CAF50; color: white; }
        .form-group { margin: 10px 0; }
        input, select { padding: 5px; width: 200px; }
```

```
        button { padding: 8px 16px; background: #4CAF50; color: white; border: none; cursor: pointer; }
        .error { color: red; }
        .success { color: green; }
    </style>
</head>
<body>
    <h1>Библиотека</h1>

    <div id="books">
        <h2>Список книг</h2>
        <table id="booksTable">
            <thead>
                <tr><th>ID</th><th>Название</th><th>Автор</th><th>Жанр</th><th>В наличии</th></tr>
            </thead>
            <tbody></tbody>
        </table>
    </div>

    <div id="addBook">
        <h2>Добавить книгу</h2>
        <form id="bookForm">
            <input type="text" id="title" placeholder="Название" required>
            <input type="text" id="author" placeholder="Автор" required>
            <input type="number" id="year" placeholder="Год издания">
            <input type="number" id="stock" placeholder="Количество" value="1">
            <button type="submit">Добавить</button>
        </form>
        <div id="message"></div>
    </div>

    <div id="loanBook">
        <h2>Выдать книгу</h2>
        <form id="loanForm">
            <input type="number" id="bookId" placeholder="ID книги" required>
            <input type="number" id="readerId" placeholder="ID читателя" required>
            <button type="submit">Выдать</button>
        </form>
        <div id="loanMessage"></div>
    </div>

    <script>
        // Загрузка списка книг
        function loadBooks() {
            fetch('/api/books')
                .then(response => response.json())
                .then(data => {
                    const tbody = document.querySelector('#booksTable tbody');
                    tbody.innerHTML = '';
                    data.forEach(book => {
```

```
            tbody.innerHTML += `
                <tr>
                    <td>${book.id}</td>
                    <td>${book.название}</td>
                    <td>${book.автор}</td>
                    <td>${book.жанр || '-'}</td>
                    <td>${book.количество_экземпляров}</td>
                </tr>
            `;
        });
    });
}

// Добавление книги
document.getElementById('bookForm').addEventListener('submit', function(e) {
    e.preventDefault();
    const data = {
        название: document.getElementById('title').value,
        автор: document.getElementById('author').value,
        год_издания: parseInt(document.getElementById('year').value) || null,
        количество_экземпляров: parseInt(document.getElementById('stock').value) || 1
    };

    fetch('/api/books', {
        method: 'POST',
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify(data)
    })
    .then(response => response.json())
    .then(result => {
        document.getElementById('message').textContent = result.message || result.error;
        if (result.id) {
            this.reset();
            loadBooks();
        }
    });
});

// Выдача книги
document.getElementById('loanForm').addEventListener('submit', function(e) {
    e.preventDefault();
    const data = {
        id_книги: parseInt(document.getElementById('bookId').value),
        id_читателя: parseInt(document.getElementById('readerId').value)
    };

    fetch('/api/loans', {
        method: 'POST',
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify(data)
    })
```

```
            .then(response => response.json())
            .then(result => {
                document.getElementById('loanMessage').textContent = result.message || result.error;
                if (result.message) {
                    this.reset();
                    loadBooks();
                }
            });
        });

        loadBooks();
    </script>
</body>
</html>
```

Форма выполнения: HTML-шаблоны, скриншоты работы интерфейса.

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

Зачет проводится по завершении учебной практики на последнем аудиторном занятии.

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 УЧЕБНОЙ ПРАКТИКИ

Уровень сформированности компетенций	**Оценка**	**Критерии оценивания по видам работ**	
		тестирование (процент правильных ответов)	**прочие виды работ**
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
УП 03.01 Учебная практика

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