**Questão 1**

Dois estudantes, Lucas e Clara, discutem sobre a função do arquivo package.json em um projeto Node.js.

Lucas diz: "O arquivo package.json serve para listar as dependências do projeto, como express e mongoose."

Clara responde: "Na verdade, ele também define scripts de execução e outras configurações do projeto."

Analise as afirmações:

I) Lucas está correto ao dizer que package.json lista dependências.  
II) Clara está correta ao dizer que o arquivo contém configurações e scripts.  
III) O arquivo package.json não é necessário em um projeto Node.js.

Está correto apenas o que se afirma em:

A) Apenas I e II  
B) Apenas II e III  
C) Apenas I  
D) I, II e III

**Resposta Correta:** A) Apenas I e II  
**Explicação:** O package.json lista as dependências e pode definir scripts e outras configurações do projeto. Ele é essencial para gerenciar o ambiente Node.js.

**Questão 2**

Qual das alternativas define corretamente o papel da biblioteca bcrypt em uma aplicação Node.js?

A) Fazer consultas ao banco de dados MongoDB.  
B) Criptografar senhas antes de armazená-las.  
C) Gerar tokens de autenticação.  
D) Gerenciar requisições HTTP no servidor.

**Resposta Correta:** B) Criptografar senhas antes de armazená-las.

**Explicação:**

A biblioteca bcrypt é utilizada em aplicações Node.js para criptografar senhas antes de armazená-las no banco de dados. Ela permite transformar senhas em "hashes", o que significa que elas são convertidas para uma forma criptografada e não podem ser facilmente convertidas de volta para o texto original. Isso aumenta a segurança, pois mesmo que alguém acesse o banco de dados, as senhas não estarão em texto claro.

**Questão 3**

O código a seguir cria uma estrutura básica de coleção (ou modelo) usando o pacote mongoose:

const mongoose = require('mongoose');

const Filme = mongoose.model("Filme", mongoose.Schema({

titulo: String,

sinopse: String

}));

Sobre esse código, considere as proposições:

I) Esse código define um modelo Filme com os campos titulo e sinopse.  
II) Cada instância do modelo Filme representa um item na coleção filmes do MongoDB.  
III) É necessário definir titulo e sinopse como obrigatórios para que o modelo funcione.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas II e III  
C) Apenas I e III  
D) I, II e III

**Resposta Correta:** A) Apenas I e II

**Explicação:**

1. **Proposição I** está correta, pois o código define um modelo Filme com os campos titulo e sinopse, ambos do tipo String. O modelo Filme será usado para criar instâncias de documentos com essa estrutura no MongoDB.
2. **Proposição II** também está correta, pois cada instância do modelo Filme representará um item na coleção filmes do MongoDB, com base no nome do modelo (Filme), que é automaticamente convertido para plural (filmes) no MongoDB.
3. **Proposição III** está incorreta, pois não é obrigatório definir titulo e sinopse como campos obrigatórios para que o modelo funcione. Sem o uso de required: true na definição dos campos, titulo e sinopse podem ser omitidos ao criar um documento.

**Questão 4**

Considere o código abaixo para um endpoint em Node.js utilizando o pacote Mongoose:

const express = require('express');

const mongoose = require('mongoose');

const app = express();

app.use(express.json());

const Filme = mongoose.model("Filme", mongoose.Schema({

titulo: { type: String, required: true },

sinopse: { type: String }

}));

app.post('/filmes', async (req, res) => {

const { titulo, sinopse } = req.body;

try {

const novoFilme = new Filme({ titulo, sinopse });

await novoFilme.save();

res.status(201).json(novoFilme);

} catch (error) {

res.status(400).json({ error: 'Falha ao salvar filme' });

}

});

Analise as proposições:

I) A propriedade required: true para o campo titulo impede que o filme seja salvo sem um título.  
II) Caso o campo titulo não seja fornecido, o servidor retornará um erro 400 devido à validação do Mongoose.  
III) O código cria um novo item na coleção Filme do MongoDB e o devolve como resposta JSON ao cliente.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas II e III  
C) Apenas I e III  
D) I, II e III

**Resposta Correta:** D) I, II e III  
**Explicação:** A propriedade required: true impede a criação de um documento sem titulo. A falta desse campo gera um erro 400, e o código retorna o documento recém-criado ao cliente em formato JSON.

**Questão 5**

Considere o seguinte código no back-end para listar todos os filmes e cadastrar novos filmes no MongoDB usando Mongoose e Express:

app.get("/filmes", async (req, res) => {

const filmes = await Filme.find();

res.json(filmes);

});

app.post("/filmes", async (req, res) => {

const { titulo, sinopse } = req.body;

const novoFilme = new Filme({ titulo, sinopse });

await novoFilme.save();

res.json(await Filme.find());

});

Sobre o código acima, analise as proposições:

I) O endpoint GET /filmes retorna a lista completa de filmes presentes no MongoDB em formato JSON.  
II) Após salvar um novo filme, o endpoint POST /filmes devolve a lista completa de filmes atualizada.  
III) Caso ocorra um erro ao salvar o filme no MongoDB, o código retornará uma lista vazia para o cliente.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas II e III  
C) Apenas I e III  
D) I, II e III

**Resposta Correta:** A) Apenas I e II  
**Explicação:** O endpoint GET /filmes devolve todos os filmes, e o POST /filmes salva um novo filme e retorna a lista atualizada. No entanto, se ocorrer um erro ao salvar o filme, o código lançará uma exceção, e a lista vazia não será enviada; a execução será interrompida.

**Questão 6**

Observe o código para configuração inicial de um servidor:

const express = require('express');

const app = express();

app.use(express.json());

app.listen(3000, () => console.log("Servidor em execução"));

Sobre o código, analise as proposições:

I) O código define que o servidor deve ser executado na porta 3000.  
II) express.json() permite que o servidor processe requisições com dados no formato JSON.  
III) Da forma definida pelo código, o servidor só poderá responder a requisições do tipo POST.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas I  
C) Apenas II e III  
D) I, II e III

**Resposta Correta:** A) Apenas I e II  
**Explicação:** O código define o servidor na porta 3000 e usa express.json() para processar dados JSON. A proposição III está incorreta; o servidor pode responder a outros tipos de requisições além de POST.

**Questão 7**

Ao criar um modelo com Mongoose, o código a seguir é válido:

const Filme = mongoose.model("Filme", mongoose.Schema({

titulo: { type: String, required: true },

sinopse: String

}));

Sobre o código acima, considere as proposições:

I) Filme representa uma coleção MongoDB chamada filmes.  
II) O campo titulo é obrigatório em cada documento.  
III) O campo sinopse será convertido para o tipo String automaticamente.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas II e III  
C) Apenas I  
D) I, II e III

**Resposta Correta:** A) Apenas I e II  
**Explicação:** O nome do modelo Filme é pluralizado para filmes no MongoDB, e titulo é obrigatório. sinopse deve ser fornecido como String.

**Questão 8**

Analise o código abaixo no backend para lidar com uma requisição POST:

app.post('/filmes', async (req, res) => {

const { titulo, sinopse } = req.body;

const novoFilme = new Filme({ titulo, sinopse });

await novoFilme.save();

res.status(201).json(novoFilme);

});

Considere as proposições:

I) O código salva um novo filme no banco de dados.  
II) res.status(201) indica sucesso e criação de um novo recurso.  
III) A resposta contém apenas o status HTTP e nenhuma informação adicional.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas I e III  
C) Apenas II e III  
D) I, II e III

**Resposta Correta:** A) Apenas I e II  
**Explicação:** O código cria e salva um novo filme e responde com 201 Created, além de enviar os dados do novo filme em JSON para o cliente.

**Questão 9**

Um estudante implementa a função abaixo para buscar filmes a partir do frontend:

async function carregarFilmes() {

try {

const resposta = await axios.get('/filmes');

console.log(resposta.data);

} catch (erro) {

console.error("Erro ao carregar filmes:", erro.message);

}

}

Sobre o código acima, analise as proposições:

I) A função exibe filmes no console ao obter a resposta do servidor.  
II) Caso o servidor retorne erro, a mensagem será exibida no console.  
III) catch só é executado se a requisição falhar.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas II e III  
C) Apenas I e III  
D) I, II e III

**Resposta Correta:** D) I, II e III  
**Explicação:** A função exibe dados no console se a resposta for bem-sucedida e exibe uma mensagem de erro se a requisição falhar.

**Questão 10**

Sobre o uso de axios.post para enviar dados ao servidor, considere as afirmações:

I) axios.post(url, data) envia os dados como JSON no corpo da requisição.  
II) O backend deve estar configurado para aceitar JSON com express.json().  
III) axios.post é usado para requisições GET e POST.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas I e III  
C) Apenas II  
D) I, II e III

**Resposta Correta:** A) Apenas I e II  
**Explicação:** axios.post envia dados como JSON, e express.json() deve estar configurado no backend para aceitar esse formato. axios.post não é usado para requisições GET.

**Questão 11**

Um desenvolvedor deseja obter uma lista de filmes e exibi-los em uma tabela HTML. O backend possui o seguinte endpoint:

app.get("/filmes", async (req, res) => {

const filmes = await Filme.find();

res.json(filmes);

});

E o frontend possui o código para obter os filmes:

async function carregarFilmes() {

const resposta = await axios.get('http://localhost:3000/filmes');

return resposta.data;

}

Considere as proposições:

I) axios.get envia uma requisição GET para o servidor, que retorna a lista de filmes.  
II) res.json(filmes) converte a lista de filmes em um objeto JSON para ser enviado ao frontend.  
III) return resposta.data faz com que a função carregarFilmes() retorne os filmes enviados como resposta do servidor.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas II e III  
C) Apenas I e III  
D) I, II e III

**Resposta Correta:** D) I, II e III  
**Explicação:** axios.get envia a requisição, res.json(filmes) converte a lista para JSON, e return resposta.data permite retornar os filmes para exibição no frontend.

**Questão 12**

O código abaixo define um modelo Filme no backend usando Mongoose:

const mongoose = require('mongoose');

const Filme = mongoose.model("Filme", mongoose.Schema({

titulo: String,

sinopse: String

}));

Sobre o código, analise as proposições:

I) Filme representa uma coleção no MongoDB chamada filmes.  
II) titulo e sinopse são campos do tipo String.  
III) Filme é necessário apenas para operações de consulta no MongoDB.

Está correto o que se afirma em:

A) Apenas I e II  
B) Apenas II e III  
C) Apenas I  
D) I, II e III

**Resposta Correta:** A) Apenas I e II  
**Explicação:** Filme é o modelo da coleção filmes, e titulo e sinopse são do tipo String. Filme é necessário para operações de leitura e escrita.