DOCUMENTAZIONE PROGETTO EXTRACORSO TECNOLOGIE WEB

Sistema di Gestione dell'Assistenza Tecnica Online

RIFERIMENTI

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione
Anno Accademico 2024-2025
Corso di Tecnologie WEB
Docente: Prof. Alessandro Cucchiarelli

Identificativo del Gruppo

- **Numero Gruppo:** 51
- **URL Progetto:** tweban.dii.univpm.it/~grp_51/laraProject/public

Componenti del Gruppo

Matricola	Cognome	Nome
1095256	Mariucci	Filippo

Contributo al Progetto

Essendo un progetto individuale, il contributo è del **100%** per Filippo Mariucci.

DESCRIZIONE DEL SITO

Il progetto realizza un **Sistema di Gestione dell'Assistenza Tecnica Online** per un'azienda che produce elettrodomestici innovativi. Il sistema permette la gestione di un servizio di supporto post-vendita attraverso una piattaforma web che connette l'azienda produttrice con i centri di assistenza tecnica distribuiti sul territorio.

Settore Merceologico Scelto

Produzione di Elettrodomestici Smart - L'azienda "TechSuportPro" si specializza nella produzione di elettrodomestici intelligenti per la casa moderna, inclusi:

- Lavatrici smart con connettività IoT
- Lavastoviglie a risparmio energetico
- Forni multifunzione con controllo remoto
- Frigoriferi con display touchscreen
- Aspirapolvere robotici
- Climatizzatori intelligenti

Funzionalità Principali Implementate

Livello 1 - Accesso Pubblico

- **Informazioni aziendali**: Visualizzazione di nome, localizzazione, logo, contatti dell'azienda
- **Catalogo prodotti pubblico**: Schede tecniche complete con foto, descrizione, note d'uso e modalità d'installazione
- **Ricerca prodotti avanzata**: Sistema di ricerca con supporto wildcards (es. "lav*" per lavatrici, lavastoviglie)
- **Elenco centri assistenza**: Informazioni complete sui centri di assistenza tecnica sul territorio

Livello 2 - Tecnici dei Centri di Assistenza

- **Accesso alle informazioni sui malfunzionamenti**: Visualizzazione completa dei problemi riscontrati per ogni prodotto
- **Database soluzioni tecniche**: Accesso alle procedure di risoluzione per ogni malfunzionamento
- **Ricerca malfunzionamenti**: Sistema di ricerca per parole chiave nelle descrizioni dei problemi

Livello 3 - Staff Tecnico Aziendale

- **Gestione malfunzionamenti**: Inserimento, modifica e cancellazione di malfunzionamenti e relative soluzioni
- **Aggiornamento database**: Capacità di mantenere aggiornato il database delle problematiche tecniche

Livello 4 - Amministratore

- **Gestione completa prodotti**: CRUD completo su tutti i prodotti del catalogo
- **Gestione utenti**: Amministrazione di tecnici, staff e altri amministratori
- **Gestione centri assistenza**: Amministrazione completa dei centri di assistenza

Funzionalità Opzionali Implementate

1. **Ripartizione gestione prodotti**: Ogni membro dello staff può gestire solo i prodotti a lui assegnati

2. **Archivio centri assistenza**: Sistema completo di gestione dei centri di assistenza esterni

---

ORGANIZATION CHART - MAPPA DEL SITO

```
  SISTEMA ASSISTENZA TECNICA
├──   Homepage (/)
│   ├── Informazioni aziendali
│   ├── Link alla documentazione
│   └── Menu di navigazione principale
│
├──   Informazioni Azienda (/azienda)
│   ├── Storia dell'azienda
│   ├── Mission e Vision
│   └── Dati di contatto
│
├──   Contatti (/contatti)
│   ├── Form di contatto
│   ├── Mappa sede principale
│   └── Informazioni di supporto
│
├──   CATALOGO PRODOTTI
│   ├──   Lista Prodotti (/prodotti)
│   │   ├── Filtri per categoria
│   │   ├── Ricerca avanzata con wildcards
│   │   └── Paginazione risultati
│   │
│   ├──   Ricerca Prodotti (/prodotti/search)
│   │   ├── Ricerca per nome/descrizione
│   │   ├── Supporto wildcards ("lav*")
│   │   └── Filtri avanzati
│   │
│   ├──   Dettaglio Prodotto (/prodotti/{id})
│   │   ├── [PUBBLICO] Scheda tecnica base
│   │   ├── [TECNICI] + Malfunzionamenti
│   │   └── [STAFF/ADMIN] + Gestione contenuti
│   │
│   └──   Prodotti per Categoria (/prodotti/categoria/{cat})
│
├──   CENTRI ASSISTENZA
│   ├──   Lista Centri (/centri-assistenza)
│   │   ├── Visualizzazione su mappa
│   │   ├── Ricerca per località
│   │   └── Informazioni di contatto
```

```
│   │
│   ├──   Ricerca per Località (/centri-assistenza/cerca/{loc})
│   └──   Dettaglio Centro (/centri-assistenza/{id})
│
├──  AUTENTICAZIONE
│   ├──   Login (/login)
│   │   ├── Form di accesso
│   │   ├── Validazione credenziali
│   │   └── Reindirizzamento basato su ruolo
│   │
│   └──   Logout (/logout)
│
├──  AREE RISERVATE
│   ├──   AREA TECNICI (Livello 2)
│   │   ├── Dashboard tecnico
│   │   ├── Accesso completo a malfunzionamenti
│   │   └── Ricerca soluzioni tecniche
│   │
│   ├──   AREA STAFF (Livello 3)
│   │   ├── Dashboard staff
│   │   ├── Gestione malfunzionamenti/soluzioni
│   │   └── Aggiornamento database tecnico
│   │
│   └──   AREA AMMINISTRATORE (Livello 4)
│       ├── Dashboard amministrativo
│       ├── Gestione completa prodotti
│       ├── Gestione utenti sistema
│       └── Gestione centri assistenza
│
├──  GESTIONE MALFUNZIONAMENTI
│   ├──   Lista Malfunzionamenti (/malfunzionamenti)
│   ├──   Nuovo Malfunzionamento (/malfunzionamenti/create)
│   ├──   Modifica Malfunzionamento (/malfunzionamenti/{id}/edit)
│   └──   Ricerca Malfunzionamenti
│
├──  GESTIONE UTENTI (Solo Admin)
│   ├──   Lista Utenti (/admin/utenti)
│   ├──   Nuovo Utente (/admin/utenti/create)
│   ├──   Modifica Utente (/admin/utenti/{id}/edit)
│   └──   Cancellazione Utente
│
├──  API ENDPOINTS
│   ├──   /api/prodotti/search (Ricerca AJAX)
│   ├──   /api/prodotti/{id}/malfunzionamenti (Lista malfunzionamenti)
```

```
│   ├──   /api/centri/cerca (Ricerca centri)
│   └──   /api/utenti (Gestione utenti via AJAX)
│
└──  DOCUMENTAZIONE
    └──   File PDF (/documentazione)
        ├── Descrizione progetto
        ├── Specifiche tecniche
        └── Istruzioni d'uso
```

Flussi di Navigazione Principali

1. Utente Non Registrato (Pubblico)

```
Homepage → Catalogo Prodotti → Dettaglio Prodotto (base)
       → Centri Assistenza → Ricerca Centro
       → Informazioni Azienda
```

2. Tecnico Centro Assistenza

```
Login → Dashboard Tecnico → Catalogo Completo → Malfunzionamenti → Soluzioni
```

3. Staff Aziendale

```
Login → Dashboard Staff → Gestione Malfunzionamenti → CRUD Soluzioni Tecniche
```

4. Amministratore

```
Login → Dashboard Admin → Gestione Prodotti/Utenti/Centri → Configurazione Sistema
```

SOLUZIONI ADOTTATE

1. Architettura del Sistema

Framework e Tecnologie

- **Backend**: Laravel 12 con architettura MVC
- **Frontend**: Blade Templates + CSS custom + JavaScript/jQuery
- **Database**: MySQL con relazioni ottimizzate
- **Autenticazione**: Sistema custom con middleware per livelli di accesso

Struttura del Database

```
/* Tabella Prodotti - Gestione catalogo completo */
CREATE TABLE prodotti (
```

```
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(255) NOT NULL,                    /* Nome commerciale del pro
    descrizione TEXT,                              /* Descrizione dettagliata
    categoria ENUM('lavatrici', 'lavastoviglie', 'forni', 'frigoriferi', 'aspi
    prezzo DECIMAL(10,2),                          /* Prezzo di listino */
    foto VARCHAR(255),                             /* Path dell'immagine prodot
    note_tecniche TEXT,                            /* Specifiche tecniche d'uso
    modalita_installazione TEXT,                   /* Istruzioni di installazio
    staff_assegnato_id BIGINT,                    /* FK per assegnazione prodot
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    FOREIGN KEY (staff_assegnato_id) REFERENCES users(id)
);

/* Tabella Malfunzionamenti - Database problematiche tecniche */
CREATE TABLE malfunzionamenti (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    prodotto_id BIGINT NOT NULL,                   /* FK verso prodotti */
    titolo VARCHAR(255) NOT NULL,                  /* Titolo breve del problema
    descrizione TEXT NOT NULL,                     /* Descrizione dettagliata d
    gravita ENUM('bassa', 'media', 'alta', 'critica') DEFAULT 'media',
    frequenza INT DEFAULT 0,                       /* Numero di segnalazioni ri
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    FOREIGN KEY (prodotto_id) REFERENCES prodotti(id) ON DELETE CASCADE,
    INDEX idx_prodotto_gravita (prodotto_id, gravita)  /* Indice per performan
);

/* Tabella Soluzioni - Procedure di risoluzione */
CREATE TABLE soluzioni (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    malfunzionamento_id BIGINT NOT NULL,           /* FK verso malfunzionamenti
    descrizione TEXT NOT NULL,                     /* Procedura di risoluzione
    strumenti_necessari TEXT,                      /* Lista strumenti/parti di
    tempo_stimato INT,                            /* Tempo stimato in minuti */
    livello_difficolta ENUM('facile', 'medio', 'difficile', 'esperto') DEFAULT
    efficacia_percentuale DECIMAL(5,2),           /* Percentuale di successo de
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    FOREIGN KEY (malfunzionamento_id) REFERENCES malfunzionamenti(id) ON DELET
);

/* Tabella Utenti - Sistema di autenticazione multi-livello */
CREATE TABLE users (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,          /* Username alfanumerico (no
    password VARCHAR(255) NOT NULL,                /* Hash bcrypt della passwor
```

```
    nome VARCHAR(100) NOT NULL,
    cognome VARCHAR(100) NOT NULL,
    data_nascita DATE,                          /* Solo per tecnici */
    ruolo ENUM('tecnico', 'staff', 'admin') NOT NULL,
    specializzazione VARCHAR(255),              /* Solo per tecnici */
    centro_assistenza_id BIGINT,                /* FK per tecnici dei centri
    attivo BOOLEAN DEFAULT TRUE,                /* Flag per abilitazione acco
    ultimo_accesso TIMESTAMP NULL,             /* Tracking ultimo login */
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    FOREIGN KEY (centro_assistenza_id) REFERENCES centri_assistenza(id)
);

/* Tabella Centri Assistenza - Rete territoriale */
CREATE TABLE centri_assistenza (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(255) NOT NULL,                 /* Ragione sociale del centro
    indirizzo VARCHAR(255) NOT NULL,
    citta VARCHAR(100) NOT NULL,
    provincia CHAR(2) NOT NULL,                 /* Codice provincia (es. AN,
    cap VARCHAR(5) NOT NULL,
    telefono VARCHAR(20),
    email VARCHAR(255),
    coordinate_lat DECIMAL(10, 7),              /* Latitudine per mappe */
    coordinate_lng DECIMAL(10, 7),              /* Longitudine per mappe */
    orari_apertura JSON,                        /* Orari settimanali in forma
    servizi_offerti SET('riparazione', 'manutenzione', 'installazione', 'consu
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    INDEX idx_citta (citta),                    /* Indice per ricerche geogra
    INDEX idx_provincia (provincia)
);
```

2. Sistema di Autenticazione e Autorizzazione

Middleware Personalizzato per Livelli di Accesso

```
<?php
/*
 * Middleware CheckUserLevel - Gestione autorizzazioni multi-livello
 * Implementa il controllo degli accessi secondo le specifiche del progetto
 */
class CheckUserLevel
{
    public function handle($request, Closure $next, ...$levels)
    {
```

```
        // Verifica se l'utente è autenticato
        if (!Auth::check()) {
            return redirect()->route('login')->with('error', 'Accesso richiest
        }

        $user = Auth::user();

        // Controllo del livello di autorizzazione
        // Livello 1: pubblico (non serve autenticazione)
        // Livello 2: tecnici
        // Livello 3: staff
        // Livello 4: admin

        $userLevel = $this->getUserLevel($user->ruolo);
        $requiredLevels = array_map('intval', $levels);

        if (!in_array($userLevel, $requiredLevels)) {
            abort(403, 'Accesso non autorizzato per il tuo livello utente');
        }

        return $next($request);
    }

    /**
     * Converte il ruolo utente nel corrispondente livello numerico
     */
    private function getUserLevel($ruolo)
    {
        return match($ruolo) {
            'tecnico' => 2,
            'staff' => 3,
            'admin' => 4,
            default => 1
        };
    }
}
```

Controller di Autenticazione

```
<?php
/*
 * AuthController - Gestione login/logout con validazione
 */
class AuthController extends Controller
{
    /**
```

```
     * Visualizza il form di login
     */
    public function showLogin()
    {
        // Se l'utente è già autenticato, reindirizza alla dashboard appropria
        if (Auth::check()) {
            return $this->redirectToDashboard(Auth::user());
        }

        return view('auth.login');
    }

    /**
     * Gestisce l'autenticazione dell'utente
     */
    public function login(Request $request)
    {
        // Validazione dei dati di input
        $credentials = $request->validate([
            'username' => 'required|string|max:50',
            'password' => 'required|string|min:8'
        ]);

        // Tentativo di autenticazione
        if (Auth::attempt($credentials)) {
            $request->session()->regenerate(); // Prevenzione session fixation

            $user = Auth::user();

            // Aggiornamento timestamp ultimo accesso
            $user->update(['ultimo_accesso' => now()]);

            // Reindirizzamento basato sul ruolo utente
            return $this->redirectToDashboard($user);
        }

        // Login fallito - messaggio di errore generico per sicurezza
        return back()->withErrors([
            'username' => 'Credenziali non valide.',
        ])->onlyInput('username');
    }

    /**
     * Determina la dashboard di destinazione in base al ruolo
     */
    private function redirectToDashboard($user)
    {
```

```
        return match($user->ruolo) {
            'admin' => redirect()->route('admin.dashboard'),
            'staff' => redirect()->route('staff.dashboard'),
            'tecnico' => redirect()->route('tecnico.dashboard'),
            default => redirect()->route('home')
        };
    }
}
```

3. Sistema di Ricerca Avanzata con Wildcards

Implementazione Ricerca Prodotti

```
<?php
/*
 * ProdottoController - Gestione catalogo e ricerca avanzata
 */
class ProdottoController extends Controller
{
    /**
     * Ricerca prodotti con supporto wildcards secondo specifiche progetto
     * Supporta pattern come "lav*" per lavatrici, lavastoviglie, lavelli
     */
    public function search(Request $request)
    {
        $query = $request->input('q', '');
        $categoria = $request->input('categoria', '');

        // Costruzione query di ricerca
        $prodotti = Prodotto::query();

        if (!empty($query)) {
            // Gestione wildcard "*" - ammesso solo come ultimo carattere
            if (str_ends_with($query, '*')) {
                $searchTerm = rtrim($query, '*'); // Rimuove l'asterisco finale
                $prodotti->where(function($q) use ($searchTerm) {
                    $q->where('nome', 'LIKE', $searchTerm . '%')      // Nome
                      ->orWhere('descrizione', 'LIKE', $searchTerm . '%'); //
                });
            } else {
                // Ricerca normale (senza wildcard)
                $prodotti->where(function($q) use ($query) {
                    $q->where('nome', 'LIKE', '%' . $query . '%')        // Nor
                      ->orWhere('descrizione', 'LIKE', '%' . $query . '%'); //
                });
            }
```

```
        }

        // Filtro per categoria se specificato
        if (!empty($categoria)) {
            $prodotti->where('categoria', $categoria);
        }

        // Ordinamento e paginazione
        $risultati = $prodotti->orderBy('nome')
                              ->paginate(12) // 12 prodotti per pagina
                              ->withQueryString(); // Mantiene i parametri di r

        return view('prodotti.search', compact('risultati', 'query', 'categori
    }
}
```

4. Sistema AJAX per Interazioni Dinamiche

Ricerca Malfunzionamenti in Tempo Reale

```
/**
 * Sistema AJAX per ricerca dinamica malfunzionamenti
 * Utilizza jQuery per chiamate asincrone al server
 */
$(document).ready(function() {
    // Handler per la ricerca in tempo reale dei malfunzionamenti
    $('#search-malfunzionamenti').on('input', function() {
        const searchTerm = $(this).val();
        const prodottoId = $(this).data('prodotto-id');

        // Debouncing per ridurre le chiamate al server
        clearTimeout(window.searchTimeout);
        window.searchTimeout = setTimeout(function() {
            searchMalfunzionamenti(searchTerm, prodottoId);
        }, 300); // Attesa di 300ms dopo l'ultimo carattere digitato
    });

    /**
     * Funzione di ricerca AJAX dei malfunzionamenti
     */
    function searchMalfunzionamenti(term, prodottoId) {
        // Mostra indicatore di caricamento
        $('#loading-indicator').show();
        $('#malfunzionamenti-container').addClass('loading');

        $.ajax({
```

```
        url: '/api/malfunzionamenti/search',
        method: 'GET',
        data: {
            q: term,                    // Termine di ricerca
            prodotto_id: prodottoId     // ID del prodotto per filtrare
        },
        dataType: 'json',
        success: function(response) {
            // Nasconde indicatore di caricamento
            $('#loading-indicator').hide();
            $('#malfunzionamenti-container').removeClass('loading');

            // Aggiorna il contenuto della pagina
            updateMalfunzionamentiList(response.data);

            // Aggiorna contatore risultati
            $('#risultati-count').text(response.data.length + ' risultati
        },
        error: function(xhr, status, error) {
            console.error('Errore nella ricerca:', error);

            // Nasconde indicatore e mostra messaggio di errore
            $('#loading-indicator').hide();
            $('#malfunzionamenti-container').removeClass('loading');

            showErrorMessage('Errore durante la ricerca. Riprova.');
        }
    });
}

/**
 * Aggiorna dinamicamente la lista dei malfunzionamenti
 */
function updateMalfunzionamentiList(malfunzionamenti) {
    const container = $('#malfunzionamenti-list');
    container.empty(); // Svuota il contenuto precedente

    if (malfunzionamenti.length === 0) {
        container.append('<p class="no-results">Nessun malfunzionamento tr
        return;
    }

    // Genera HTML per ogni malfunzionamento trovato
    malfunzionamenti.forEach(function(item) {
        const card = `
            <div class="malfunzionamento-card" data-id="${item.id}">
                <div class="card-header">
```

```
                        <h4>${item.titolo}</h4>
                        <span class="gravita gravita-${item.gravita}">${item.g
                    </div>
                    <div class="card-body">
                        <p>${item.descrizione}</p>
                        <div class="card-footer">
                            <span class="frequenza">Segnalazioni: ${item.frequ
                            <button class="btn-visualizza-soluzione" data-malf
                                Visualizza Soluzione
                            </button>
                        </div>
                    </div>
                </div>
            `;
            container.append(card);
        });

        // Riattiva i handler per i nuovi elementi
        attachSolutionHandlers();
    }
});
```

5. Gestione File e Upload Immagini

Sistema Upload Foto Prodotti

```
<?php
/*
 * Sistema di gestione upload immagini prodotti con validazione e ottimizzazio
 */
class ImageUploadService
{
    private $allowedExtensions = ['jpg', 'jpeg', 'png', 'webp'];
    private $maxFileSize = 5242880; // 5MB in bytes
    private $uploadPath = 'uploads/prodotti/';

    /**
     * Gestisce l'upload di un'immagine prodotto
     */
    public function uploadProductImage($file, $prodottoId)
    {
        try {
            // Validazione del file
            $this->validateImage($file);

            // Generazione nome file univoco
```

```
        $extension = $file->getClientOriginalExtension();
        $filename = 'prodotto_' . $prodottoId . '_' . time() . '.' . $exte

        // Creazione directory se non esiste
        $fullUploadPath = public_path($this->uploadPath);
        if (!file_exists($fullUploadPath)) {
            mkdir($fullUploadPath, 0755, true);
        }

        // Spostamento file nella directory di destinazione
        $file->move($fullUploadPath, $filename);

        // Ottimizzazione dell'immagine per il web
        $this->optimizeImage($fullUploadPath . $filename);

        // Ritorna il path relativo per il database
        return $this->uploadPath . $filename;

    } catch (Exception $e) {
        throw new ImageUploadException('Errore durante l\'upload: ' . $e->
    }
}

/**
 * Validazione dell'immagine caricata
 */
private function validateImage($file)
{
    // Controllo se il file è stato caricato correttamente
    if (!$file->isValid()) {
        throw new ImageUploadException('File non valido o corrotto');
    }

    // Controllo estensione
    $extension = strtolower($file->getClientOriginalExtension());
    if (!in_array($extension, $this->allowedExtensions)) {
        throw new ImageUploadException('Formato file non supportato. Usa:
    }

    // Controllo dimensione file
    if ($file->getSize() > $this->maxFileSize) {
        throw new ImageUploadException('File troppo grande. Massimo 5MB co

    // Controllo tipo MIME per sicurezza aggiuntiva
    $allowedMimes = ['image/jpeg', 'image/png', 'image/webp'];
    if (!in_array($file->getMimeType(), $allowedMimes)) {
```

```
            throw new ImageUploadException('Tipo di file non valido');
        }
    }

    /**
     * Ottimizza l'immagine per ridurre dimensioni mantenendo qualità
     */
    private function optimizeImage($imagePath)
    {
        // Carica l'immagine in base al tipo
        $imageInfo = getimagesize($imagePath);
        $imageType = $imageInfo[2];

        switch ($imageType) {
            case IMAGETYPE_JPEG:
                $image = imagecreatefromjpeg($imagePath);
                break;
            case IMAGETYPE_PNG:
                $image = imagecreatefrompng($imagePath);
                break;
            case IMAGETYPE_WEBP:
                $image = imagecreatefromwebp($imagePath);
                break;
            default:
                return; // Tipo non supportato per l'ottimizzazione
        }

        // Ridimensiona se troppo grande (max 1200x1200)
        $width = imagesx($image);
        $height = imagesy($image);

        if ($width > 1200 || $height > 1200) {
            $ratio = min(1200 / $width, 1200 / $height);
            $newWidth = intval($width * $ratio);
            $newHeight = intval($height * $ratio);

            $resized = imagecreatetruecolor($newWidth, $newHeight);

            // Mantieni trasparenza per PNG
            if ($imageType == IMAGETYPE_PNG) {
                imagealphablending($resized, false);
                imagesavealpha($resized, true);
            }

            imagecopyresampled($resized, $image, 0, 0, 0, 0, $newWidth, $newHe

            // Salva l'immagine ottimizzata
```

```
            switch ($imageType) {
                case IMAGETYPE_JPEG:
                    imagejpeg($resized, $imagePath, 85); // Qualità 85%
                    break;
                case IMAGETYPE_PNG:
                    imagepng($resized, $imagePath, 6); // Compressione livello
                    break;
                case IMAGETYPE_WEBP:
                    imagewebp($resized, $imagePath, 85); // Qualità 85%
                    break;
            }

            imagedestroy($resized);
        }

        imagedestroy($image);
    }
}
```

6. Sistema di Validazione Dati

Validazione Form con Regole Custom

```
<?php
/*
 * Request classes per validazione dati con regole personalizzate
 */

// Validazione per creazione/modifica prodotti
class ProdottoRequest extends FormRequest
{
    /**
     * Determina se l'utente è autorizzato a fare questa richiesta
     */
    public function authorize()
    {
        // Solo admin e staff possono gestire i prodotti
        return Auth::check() && in_array(Auth::user()->ruolo, ['admin', 'staff
    }

    /**
     * Regole di validazione per i prodotti
     */
    public function rules()
    {
        $rules = [
```

```
            'nome' => 'required|string|max:255|unique:prodotti,nome',
            'descrizione' => 'required|string|min:50',
            'categoria' => 'required|in:lavatrici,lavastoviglie,forni,frigorif
            'prezzo' => 'required|numeric|min:0|max:999999.99',
            'note_tecniche' => 'required|string|min:20',
            'modalita_installazione' => 'required|string|min:30',
            'foto' => 'nullable|image|mimes:jpeg,png,jpg,webp|max:5120' // 5MB
        ];

        // Se stiamo modificando, escludiamo il prodotto corrente dalla valida
        if ($this->route('prodotto')) {
            $rules['nome'] = 'required|string|max:255|unique:prodotti,nome,' .
        }

        return $rules;
    }

    /**
     * Messaggi di errore personalizzati
     */
    public function messages()
    {
        return [
            'nome.required' => 'Il nome del prodotto è obbligatorio',
            'nome.unique' => 'Esiste già un prodotto con questo nome',
            'descrizione.min' => 'La descrizione deve essere di almeno 50 cara
            'categoria.in' => 'Categoria non valida',
            'prezzo.numeric' => 'Il prezzo deve essere un numero valido',
            'prezzo.min' => 'Il prezzo non può essere negativo',
            'foto.image' => 'Il file deve essere un\'immagine',
            'foto.mimes' => 'Formati supportati: JPEG, PNG, JPG, WEBP',
            'foto.max' => 'L\'immagine non può superare i 5MB'
        ];
    }
}

// Validazione per malfunzionamenti
class MalfunzionamentoRequest extends FormRequest
{
    public function authorize()
    {
        return Auth::check() && Auth::user()->ruolo === 'staff';
    }

    public function rules()
    {
        return [
```

```
            'prodotto_id' => 'required|exists:prodotti,id',
            'titolo' => 'required|string|max:255',
            'descrizione' => 'required|string|min:30',
            'gravita' => 'required|in:bassa,media,alta,critica',
            'soluzione.descrizione' => 'required|string|min:50',
            'soluzione.strumenti_necessari' => 'nullable|string',
            'soluzione.tempo_stimato' => 'nullable|integer|min:1|max:1440', //
            'soluzione.livello_difficolta' => 'required|in:facile,medio,diffic
        ];
    }
}
```

7. Performance e Ottimizzazione Database

Query Ottimizzate con Eager Loading

```
<?php
/*
 * Ottimizzazioni database per migliorare le performance
 */

class ProdottoService
{
    /**
     * Carica prodotti con relazioni ottimizzate per evitare N+1 queries
     */
    public function getProdottiConMalfunzionamenti($perPage = 12)
    {
        return Prodotto::with([
            'malfunzionamenti' => function($query) {
                // Carica solo malfunzionamenti attivi ordinati per gravità
                $query->orderByRaw("FIELD(gravita, 'critica', 'alta', 'media',
                      ->limit(5); // Limita ai 5 più importanti per performanc
            },
            'malfunzionamenti.soluzioni' => function($query) {
                // Carica solo la soluzione più efficace
                $query->orderByDesc('efficacia_percentuale')->limit(1);
            },
            'staffAssegnato:id,nome,cognome' // Carica solo campi necessari
        ])
        ->select(['id', 'nome', 'categoria', 'foto', 'staff_assegnato_id']) //
        ->paginate($perPage);
    }

    /**
     * Ricerca ottimizzata con indici full-text
```

```
     */
    public function searchProdotti($term, $categoria = null)
    {
        $query = Prodotto::query();

        // Usa indice full-text se disponibile, altrimenti LIKE ottimizzato
        if (!empty($term)) {
            if (str_ends_with($term, '*')) {
                $searchTerm = rtrim($term, '*');
                $query->where(function($q) use ($searchTerm) {
                    $q->where('nome', 'LIKE', $searchTerm . '%')
                      ->orWhere('descrizione', 'LIKE', $searchTerm . '%');
                });
            } else {
                // Ricerca full-text quando possibile
                $query->whereRaw(
                    "MATCH(nome, descrizione) AGAINST(? IN BOOLEAN MODE)",
                    [$term . '*']
                );
            }
        }

        if ($categoria) {
            $query->where('categoria', $categoria);
        }

        return $query->with('staffAssegnato:id,nome,cognome')
                     ->orderByRaw('MATCH(nome, descrizione) AGAINST(?) DESC', [
                     ->paginate(12);
    }
}

/*
 * Indici per ottimizzazione performance
 */
// Migration per aggiungere indici ottimizzati
Schema::table('prodotti', function (Blueprint $table) {
    // Indice composito per ricerche per categoria
    $table->index(['categoria', 'nome']);

    // Indice full-text per ricerche testuali
    $table->fullText(['nome', 'descrizione'], 'prodotti_search_fulltext');
});

Schema::table('malfunzionamenti', function (Blueprint $table) {
    // Indice composito per query frequenti
    $table->index(['prodotto_id', 'gravita', 'created_at']);
```

```
    // Indice per ricerche testuali nei malfunzionamenti
    $table->fullText(['titolo', 'descrizione'], 'malfunzionamenti_search_fullt
});
```

8. Sicurezza e Protezione Dati

Middleware di Sicurezza Custom

```
<?php
/*
 * Sistema di sicurezza avanzato per proteggere l'applicazione
 */

class SecurityMiddleware
{
    /**
     * Protezione contro attacchi CSRF, XSS e SQL Injection
     */
    public function handle($request, Closure $next)
    {
        // Rate limiting per prevenire attacchi bruteforce
        $this->applyRateLimit($request);

        // Sanitizzazione input per prevenire XSS
        $this->sanitizeInput($request);

        // Validazione headers di sicurezza
        $this->validateSecurityHeaders($request);

        $response = $next($request);

        // Aggiunta headers di sicurezza alla risposta
        $this->addSecurityHeaders($response);

        return $response;
    }

    /**
     * Applica rate limiting basato su IP e utente
     */
    private function applyRateLimit($request)
    {
        $key = 'rate_limit:' . $request->ip();
        $attempts = Cache::get($key, 0);
```

```
        // Massimo 100 richieste per ora per IP
        if ($attempts >= 100) {
            abort(429, 'Troppo molte richieste. Riprova più tardi.');
        }

        Cache::put($key, $attempts + 1, 3600); // 1 ora
    }

    /**
     * Sanitizza input per prevenire XSS
     */
    private function sanitizeInput($request)
    {
        $input = $request->all();

        array_walk_recursive($input, function(&$value) {
            if (is_string($value)) {
                // Rimuove tag HTML pericolosi mantenendo quelli sicuri
                $value = strip_tags($value, '<p><br><strong><em><ul><ol><li>')

                // Escape caratteri speciali
                $value = htmlspecialchars($value, ENT_QUOTES, 'UTF-8');
            }
        });

        $request->merge($input);
    }

    /**
     * Aggiunge headers di sicurezza HTTP
     */
    private function addSecurityHeaders($response)
    {
        $response->headers->set('X-Content-Type-Options', 'nosniff');
        $response->headers->set('X-Frame-Options', 'DENY');
        $response->headers->set('X-XSS-Protection', '1; mode=block');
        $response->headers->set('Strict-Transport-Security', 'max-age=31536000
        $response->headers->set('Content-Security-Policy', "default-src 'self'

        return $response;
    }
}

/*
 * Protezione password con hashing sicuro
 */
class SecurePasswordService
```

```
{
    /**
     * Genera hash sicuro della password
     */
    public static function hashPassword($password)
    {
        // Verifica complessità password
        if (!self::isPasswordComplex($password)) {
            throw new InvalidArgumentException('Password non sufficientemente c
        }

        // Usa bcrypt con cost factor alto per sicurezza
        return password_hash($password, PASSWORD_BCRYPT, ['cost' => 12]);
    }

    /**
     * Verifica complessità password
     */
    private static function isPasswordComplex($password)
    {
        // Minimo 8 caratteri, almeno una maiuscola, una minuscola, un numero
        return strlen($password) >= 8 &&
               preg_match('/[A-Z]/', $password) &&
               preg_match('/[a-z]/', $password) &&
               preg_match('/[0-9]/', $password);
    }

    /**
     * Verifica password contro hash
     */
    public static function verifyPassword($password, $hash)
    {
        return password_verify($password, $hash);
    }
}
```

STRUMENTI UTILIZZATI

HTML - Struttura delle Pagine

- **Semantic HTML5**: Uso di tag semantici (`<header>` , `<nav>` , `<main>` , `<section>` , `<article>` , `<aside>` , `<footer>`)
- **Accessibilità**: Attributi ARIA, alt text per immagini, struttura logica per screen reader
- **Form avanzati**: Validazione lato client, input types specifici (`email` , `tel` , `date`)

- **Microdata**: Schema.org markup per SEO e ricerca

HTML - Elementi Multimediali

- **Gestione immagini responsive**: Uso di `srcset` e `sizes` per ottimizzazione mobile
- **Lazy loading**: Caricamento ritardato delle immagini per performance
- **Ottimizzazione formati**: Supporto WebP con fallback JPEG/PNG
- **Gallery dinamiche**: Implementazione carousel e lightbox per foto prodotti

CSS - Utilizzo degli Stili

- **CSS Grid e Flexbox**: Layout responsive moderni
- **Custom Properties**: Variabili CSS per temi consistenti
- **Media queries**: Design completamente responsive (mobile-first approach)
- **Animazioni CSS**: Transizioni fluide e micro-interazioni
- **SCSS/Sass**: Preprocessore per codice CSS organizzato e modulare

```
/* Esempio di CSS avanzato utilizzato */
:root {
    --primary-color: #2563eb;
    --secondary-color: #64748b;
    --success-color: #059669;
    --warning-color: #d97706;
    --danger-color: #dc2626;
    --text-primary: #1e293b;
    --text-secondary: #64748b;
    --bg-primary: #ffffff;
    --bg-secondary: #f8fafc;
    --border-color: #e2e8f0;
    --shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.1);
}

/* Layout Grid Responsive */
.products-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 2rem;
    padding: 2rem;
}

/* Animazioni per feedback utente */
.card {
    transition: transform 0.2s ease, box-shadow 0.2s ease;
}

.card:hover {
```

```
    transform: translateY(-2px);
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
}

/* Responsive design con mobile-first */
@media (max-width: 768px) {
    .products-grid {
        grid-template-columns: 1fr;
        gap: 1rem;
        padding: 1rem;
    }
}
```

JavaScript - Funzioni Complesse

- **Modular JavaScript**: Organizzazione in moduli ES6
- **Event handling avanzato**: Gestione eventi con delegation
- **Manipolazione DOM**: Creazione dinamica di elementi
- **Gestione stato**: State management per interfaccia reattiva
- **Validazione client-side**: Controlli in tempo reale sui form

```
/**
 * Modulo per gestione ricerca avanzata prodotti
 */
class ProductSearchManager {
    constructor() {
        this.searchInput = document.getElementById('product-search');
        this.filterContainer = document.getElementById('filters');
        this.resultsContainer = document.getElementById('results');
        this.loadingIndicator = document.getElementById('loading');

        this.debounceTimer = null;
        this.currentFilters = {};

        this.initializeEventListeners();
    }

    /**
     * Inizializza event listeners per la ricerca
     */
    initializeEventListeners() {
        // Ricerca con debouncing
        this.searchInput.addEventListener('input', (e) => {
            clearTimeout(this.debounceTimer);
            this.debounceTimer = setTimeout(() => {
                this.performSearch(e.target.value);
```

```
        }, 300);
    });

    // Gestione filtri categoria
    this.filterContainer.addEventListener('change', (e) => {
        if (e.target.type === 'checkbox') {
            this.updateFilters(e.target.name, e.target.checked);
        }
    });

    // Gestione paginazione dinamica
    document.addEventListener('click', (e) => {
        if (e.target.classList.contains('pagination-link')) {
            e.preventDefault();
            this.loadPage(e.target.dataset.page);
        }
    });
}

/**
 * Esegue ricerca con chiamata AJAX
 */
async performSearch(query) {
    try {
        this.showLoading(true);

        const searchParams = new URLSearchParams({
            q: query,
            ...this.currentFilters
        });

        const response = await fetch(`/api/prodotti/search?${searchParams}
        const data = await response.json();

        this.updateResults(data.products);
        this.updatePagination(data.pagination);

    } catch (error) {
        console.error('Errore nella ricerca:', error);
        this.showError('Errore durante la ricerca');
    } finally {
        this.showLoading(false);
    }
}

/**
 * Aggiorna risultati di ricerca nel DOM
```

```
     */
    updateResults(products) {
        // Svuota contenitore precedente
        this.resultsContainer.innerHTML = '';

        if (products.length === 0) {
            this.resultsContainer.innerHTML = '<p class="no-results">Nessun pr
            return;
        }

        // Crea cards per ogni prodotto
        products.forEach(product => {
            const productCard = this.createProductCard(product);
            this.resultsContainer.appendChild(productCard);
        });

        // Applica animazione di ingresso
        this.animateResults();
    }

    /**
     * Crea card HTML per un prodotto
     */
    createProductCard(product) {
        const card = document.createElement('div');
        card.className = 'product-card';
        card.innerHTML = `
            <div class="product-image">
                <img src="${product.foto}" alt="${product.nome}" loading="lazy
            </div>
            <div class="product-info">
                <h3>${product.nome}</h3>
                <p class="category">${product.categoria}</p>
                <p class="price">€${product.prezzo}</p>
                <a href="/prodotti/${product.id}" class="btn btn-primary">Dett
            </div>
        `;

        return card;
    }
}

// Inizializzazione quando DOM è pronto
document.addEventListener('DOMContentLoaded', () => {
    new ProductSearchManager();
});
```

jQuery - Visualizzazione Elementi e AJAX

Gestione Dinamica Interfaccia

```
/**
 * Sistema jQuery per gestione dinamica malfunzionamenti
 */
$(document).ready(function() {

    /**
     * Caricamento dinamico lista malfunzionamenti
     */
    function loadMalfunzionamenti(prodottoId) {
        const container = $('#malfunzionamenti-container');

        // Mostra skeleton loader durante caricamento
        container.html(`
            <div class="loading-skeleton">
                <div class="skeleton-item"></div>
                <div class="skeleton-item"></div>
                <div class="skeleton-item"></div>
            </div>
        `);

        $.ajax({
            url: `/api/prodotti/${prodottoId}/malfunzionamenti`,
            method: 'GET',
            dataType: 'json',
            success: function(response) {
                renderMalfunzionamenti(response.data);
                initializeMalfunzionamentHandlers();
            },
            error: function(xhr, status, error) {
                container.html(`
                    <div class="error-message">
                        <i class="icon-warning"></i>
                        <p>Errore nel caricamento dei malfunzionamenti</p>
                        <button class="btn-retry" onclick="loadMalfunzionament
                            Riprova
                        </button>
                    </div>
                `);
            }
        });
    }
```

```
/**
 * Rendering HTML dei malfunzionamenti
 */
function renderMalfunzionamenti(malfunzionamenti) {
    const container = $('#malfunzionamenti-container');

    if (malfunzionamenti.length === 0) {
        container.html('<p class="no-content">Nessun malfunzionamento regi
        return;
    }

    let html = '<div class="malfunzionamenti-grid">';

    malfunzionamenti.forEach(function(item) {
        html += `
            <div class="malfunzionamento-card" data-id="${item.id}">
                <div class="card-header">
                    <h4 class="malfunzionamento-title">${item.titolo}</h4>
                    <span class="gravita-badge gravita-${item.gravita}">
                        ${item.gravita.toUpperCase()}
                    </span>
                </div>
                <div class="card-body">
                    <p class="malfunzionamento-description">${item.descriz
                    <div class="malfunzionamento-stats">
                        <span class="stat-item">
                            <i class="icon-alert"></i>
                            Frequenza: ${item.frequenza}
                        </span>
                        <span class="stat-item">
                            <i class="icon-clock"></i>
                            ${formatDate(item.created_at)}
                        </span>
                    </div>
                </div>
                <div class="card-footer">
                    <button class="btn btn-outline view-solution"
                            data-malfunzionamento-id="${item.id}">
                        <i class="icon-solution"></i>
                        Visualizza Soluzione
                    </button>
                </div>
            </div>
        `;
    });

    html += '</div>';
```

```
    container.html(html);

    // Applica animazione fadeIn
    $('.malfunzionamento-card').hide().fadeIn(600);
}

/**
 * Modal per visualizzazione soluzioni
 */
function showSolutionModal(malfunzionamentoId) {
    const modal = $('#solution-modal');
    const modalBody = modal.find('.modal-body');

    // Mostra loader nel modal
    modalBody.html('<div class="text-center"><i class="spinner"></i> Carica
    modal.fadeIn(300);

    $.ajax({
        url: `/api/malfunzionamenti/${malfunzionamentoId}/soluzioni`,
        method: 'GET',
        success: function(response) {
            renderSolutionContent(response.data, modalBody);
        },
        error: function() {
            modalBody.html('<div class="error">Errore nel caricamento della
        }
    });
}

/**
 * Event handlers per malfunzionamenti
 */
function initializeMalfunzionamentHandlers() {
    // Click su visualizza soluzione
    $('.view-solution').off('click').on('click', function() {
        const malfunzionamentoId = $(this).data('malfunzionamento-id');
        showSolutionModal(malfunzionamentoId);
    });

    // Hover effects
    $('.malfunzionamento-card').hover(
        function() {
            $(this).addClass('hovered').find('.view-solution').addClass('b
        },
        function() {
            $(this).removeClass('hovered').find('.view-solution').removeCl
        }
```

```
        );
    }

    /**
     * Chiusura modal con animazione
     */
    $('.modal-close, .modal-overlay').on('click', function() {
        $('#solution-modal').fadeOut(300);
    });

    // Chiusura modal con ESC
    $(document).on('keydown', function(e) {
        if (e.keyCode === 27) { // ESC key
            $('#solution-modal').fadeOut(300);
        }
    });
});
```

Laravel - Uso Componenti di Base

Architettura MVC Completa

- **Models Eloquent**: Relazioni complesse, mutators, accessors, scopes
- **Controllers RESTful**: Organizzazione logica delle funzionalità
- **Blade Templates**: Template engine con componenti riutilizzabili
- **Middleware personalizzato**: Gestione autorizzazioni multi-livello
- **Service Providers**: Registrazione servizi personalizzati
- **Artisan Commands**: Comandi custom per manutenzione

```
<?php
/**
 * Model Prodotto con relazioni ottimizzate
 */
class Prodotto extends Model
{
    protected $fillable = [
        'nome', 'descrizione', 'categoria', 'prezzo',
        'foto', 'note_tecniche', 'modalita_installazione',
        'staff_assegnato_id'
    ];

    protected $casts = [
        'prezzo' => 'decimal:2',
        'created_at' => 'datetime',
        'updated_at' => 'datetime'
    ];
```

```
    /**
     * Relazione con malfunzionamenti
     */
    public function malfunzionamenti()
    {
        return $this->hasMany(Malfunzionamento::class)
                    ->orderByRaw("FIELD(gravita, 'critica', 'alta', 'media', 'b
    }

    /**
     * Relazione con staff assegnato
     */
    public function staffAssegnato()
    {
        return $this->belongsTo(User::class, 'staff_assegnato_id')
                    ->where('ruolo', 'staff');
    }

    /**
     * Scope per prodotti per categoria
     */
    public function scopeByCategoria($query, $categoria)
    {
        return $query->where('categoria', $categoria);
    }

    /**
     * Scope per ricerca testuale
     */
    public function scopeSearch($query, $term)
    {
        return $query->where(function($q) use ($term) {
            $q->where('nome', 'LIKE', "%{$term}%")
              ->orWhere('descrizione', 'LIKE', "%{$term}%");
        });
    }

    /**
     * Accessor per URL foto
     */
    public function getFotoUrlAttribute()
    {
        return $this->foto ? asset($this->foto) : asset('images/no-image.png')
    }

    /**
```

```
     * Accessor per prezzo formattato
     */
    public function getPrezzoFormattedAttribute()
    {
        return '€ ' . number_format($this->prezzo, 2, ',', '.');
    }
}

/**
 * Controller con pattern Repository
 */
class ProdottoController extends Controller
{
    protected $prodottoService;

    public function __construct(ProdottoService $prodottoService)
    {
        $this->prodottoService = $prodottoService;

        // Middleware per autorizzazioni
        $this->middleware('auth')->except(['index', 'show', 'search']);
        $this->middleware('check.user.level:4')->only(['create', 'store', 'des
        $this->middleware('check.user.level:3,4')->only(['edit', 'update']);
    }

    /**
     * Lista prodotti con paginazione
     */
    public function index(Request $request)
    {
        $prodotti = $this->prodottoService->getPaginatedProducts(
            $request->input('categoria'),
            $request->input('search'),
            12 // Prodotti per pagina
        );

        $categorie = $this->prodottoService->getCategorie();

        return view('prodotti.index', compact('prodotti', 'categorie'));
    }

    /**
     * Creazione nuovo prodotto
     */
    public function store(ProdottoRequest $request)
    {
        DB::beginTransaction();
```

```
        try {
            $prodotto = $this->prodottoService->createProdotto($request->valida

            DB::commit();

            return redirect()
                ->route('prodotti.show', $prodotto)
                ->with('success', 'Prodotto creato con successo');

        } catch (Exception $e) {
            DB::rollback();

            return back()
                ->withInput()
                ->with('error', 'Errore nella creazione del prodotto: ' . $e->
        }
    }
}
```

Laravel - Gestione Autorizzazioni

Sistema Autorizzazioni Multi-livello

```
<?php
/**
 * Sistema completo di autorizzazioni basato su ruoli e policies
 */

// Policy per gestione autorizzazioni prodotti
class ProdottoPolicy
{
    /**
     * Determina se l'utente può visualizzare tutti i prodotti
     */
    public function viewAny(User $user)
    {
        return true; // Tutti possono vedere la lista pubblica
    }

    /**
     * Determina se l'utente può vedere i dettagli completi (con malfunzioname
     */
    public function view(User $user, Prodotto $prodotto)
    {
        // Tecnici e staff possono vedere tutto
```

```
        return in_array($user->ruolo, ['tecnico', 'staff', 'admin']);
    }

    /**
     * Determina se l'utente può creare prodotti
     */
    public function create(User $user)
    {
        return $user->ruolo === 'admin';
    }

    /**
     * Determina se l'utente può modificare un prodotto
     */
    public function update(User $user, Prodotto $prodotto)
    {
        // Admin può modificare tutto
        if ($user->ruolo === 'admin') {
            return true;
        }

        // Staff può modificare solo prodotti assegnati
        if ($user->ruolo === 'staff') {
            return $prodotto->staff_assegnato_id === $user->id;
        }

        return false;
    }

    /**
     * Determina se l'utente può eliminare un prodotto
     */
    public function delete(User $user, Prodotto $prodotto)
    {
        // Solo admin può eliminare
        return $user->ruolo === 'admin';
    }

    /**
     * Determina se l'utente può gestire malfunzionamenti del prodotto
     */
    public function manageMalfunzionamenti(User $user, Prodotto $prodotto)
    {
        if ($user->ruolo === 'admin') {
            return true;
        }
```

```
        if ($user->ruolo === 'staff') {
            return $prodotto->staff_assegnato_id === $user->id;
        }

        return false;
    }
}

// Gate personalizzato per controlli complessi
class AuthServiceProvider extends ServiceProvider
{
    public function boot()
    {
        $this->registerPolicies();

        // Gate per controllo accesso dashboard
        Gate::define('access-dashboard', function (User $user, $level) {
            $userLevel = match($user->ruolo) {
                'tecnico' => 2,
                'staff' => 3,
                'admin' => 4,
                default => 1
            };

            return $userLevel >= $level;
        });

        // Gate per gestione centri assistenza
        Gate::define('manage-centri', function (User $user) {
            return $user->ruolo === 'admin';
        });

        // Gate per gestione utenti
        Gate::define('manage-users', function (User $user) {
            return $user->ruolo === 'admin';
        });
    }
}

/**
 * Middleware per controllo livelli di accesso
 */
class CheckAccessLevel
{
    public function handle($request, Closure $next, $level)
    {
        if (!Auth::check()) {
```

```
            return redirect()->route('login');
        }

        if (!Gate::allows('access-dashboard', (int)$level)) {
            abort(403, 'Accesso non autorizzato');
        }

        return $next($request);
    }
}

/**
 * Blade directive per controlli autorizzazione nel template
 */
// In AppServiceProvider::boot()
Blade::directive('canLevel', function ($level) {
    return "<?php if(Gate::allows('access-dashboard', $level)): ?>";
});

Blade::directive('endcanLevel', function () {
    return '<?php endif; ?>';
});
```

ASPETTI ORGANIZZATIVI

1. Articolazione Complessiva del Progetto

Struttura Directory Laravel Ottimizzata

```
/laraProject
├── app/
│   ├── Http/
│   │   ├── Controllers/          # Controller organizzati per funzionalità
│   │   │   ├── Auth/             # Autenticazione
│   │   │   ├── Admin/            # Area amministrativa
│   │   │   ├── Staff/            # Area staff
│   │   │   └── Api/              # API endpoints
│   │   ├── Middleware/           # Middleware personalizzati
│   │   ├── Requests/             # Form request validation
│   │   └── Resources/            # API resources
│   ├── Models/                   # Eloquent models
│   ├── Services/                 # Business logic
│   ├── Policies/                 # Authorization policies
│   └── Providers/                # Service providers
```

```
├── resources/
│   ├── views/
│   │   ├── layouts/              # Template base
│   │   ├── components/           # Componenti riutilizzabili
│   │   ├── auth/                 # Pagine autenticazione
│   │   ├── prodotti/             # Gestione prodotti
│   │   ├── malfunzionamenti/     # Gestione malfunzionamenti
│   │   └── admin/                # Interface amministrative
│   ├── scss/                     # Stili SASS organizzati
│   └── js/                       # JavaScript modulare
├── public/
│   ├── css/                      # CSS compilati
│   ├── js/                       # JavaScript compilati
│   ├── images/                   # Immagini statiche
│   ├── uploads/                  # File caricati dagli utenti
│   └── docs/                     # Documentazione PDF
├── database/
│   ├── migrations/               # Schema database
│   ├── seeders/                  # Dati di test
│   └── factories/                # Model factories
└── routes/
    ├── web.php                   # Route web principali
    ├── api.php                   # Route API
    └── admin.php                 # Route amministrative
```

Metodologia di Sviluppo

- **Approccio MVC**: Separazione netta tra logica, presentazione e dati
- **Repository Pattern**: Astrazione dell'accesso ai dati
- **Service Layer**: Incapsulamento della business logic
- **Component-Based**: Interfaccia modulare e riutilizzabile
- **API-First**: Progettazione con separazione frontend/backend

Gestione Configurazioni

```php
<?php
// config/database.php - Configurazione database secondo specifiche
return [
    'default' => 'mysql',
    'connections' => [
        'mysql' => [
            'driver' => 'mysql',
            'host' => $HOST,              // Variabile dal file connect.php
            'database' => $DB,            // Nome database gruppo
            'username' => $USER,          // Username MySQL gruppo
```

```
            'password' => $PASSWORD,      // Password MySQL gruppo
            'charset' => 'utf8mb4',
            'collation' => 'utf8mb4_unicode_ci',
            'prefix' => '',
            'strict' => true,
            'engine' => 'InnoDB',
        ],
    ],
];

// database/seeders/DatabaseSeeder.php - Utenti di sistema richiesti
class DatabaseSeeder extends Seeder
{
    public function run()
    {
        // Password: dNWRdNWR (primi 4 caratteri SSH ripetuti)
        $password = Hash::make('dNWRdNWR');

        // Utente Tecnico (Livello 2)
        User::create([
            'username' => 'tecntecn',
            'password' => $password,
            'nome' => 'Mario',
            'cognome' => 'Rossi',
            'ruolo' => 'tecnico',
            'data_nascita' => '1985-06-15',
            'specializzazione' => 'Elettrodomestici Smart',
            'centro_assistenza_id' => 1
        ]);

        // Utente Staff (Livello 3)
        User::create([
            'username' => 'staffstaff',
            'password' => $password,
            'nome' => 'Laura',
            'cognome' => 'Bianchi',
            'ruolo' => 'staff'
        ]);

        // Utente Amministratore (Livello 4)
        User::create([
            'username' => 'adminadmin',
            'password' => $password,
            'nome' => 'Alessandro',
            'cognome' => 'Verdi',
            'ruolo' => 'admin'
        ]);
```

```
        // Seeding dati di esempio
        $this->call([
            CentroAssistenzaSeeder::class,
            ProdottoSeeder::class,
            MalfunzionamentoSeeder::class,
        ]);
    }
}
```

2. Documentazione di Progetto

Standard di Documentazione Adottati

- **Commenti PHP**: Documentazione PHPDoc per tutti i metodi
- **Commenti JavaScript**: JSDoc per funzioni complesse
- **README tecnico**: Istruzioni installazione e configurazione
- **API Documentation**: Documentazione endpoint con esempi
- **Database Schema**: Diagrammi ER e descrizione tabelle

Esempio Documentazione Codice

```
<?php
/**
 * Service per gestione completa dei prodotti
 *
 * Questo service incapsula tutta la business logic relativa ai prodotti,
 * inclusa la gestione delle immagini, validazione dati e interazioni
 * con malfunzionamenti associati.
 *
 * @package App\Services
 * @author Filippo Mariucci <1095256@studenti.univpm.it>
 * @version 1.0
 */
class ProdottoService
{
    /**
     * Repository per accesso dati prodotti
     * @var ProdottoRepository
     */
    protected $prodottoRepository;

    /**
     * Service per gestione upload immagini
     * @var ImageUploadService
     */
```

```
protected $imageService;

/**
 * Costruttore con dependency injection
 *
 * @param ProdottoRepository $prodottoRepository Repository prodotti
 * @param ImageUploadService $imageService Service gestione immagini
 */
public function __construct(
    ProdottoRepository $prodottoRepository,
    ImageUploadService $imageService
) {
    $this->prodottoRepository = $prodottoRepository;
    $this->imageService = $imageService;
}

/**
 * Crea un nuovo prodotto con gestione completa dati e immagine
 *
 * Questo metodo gestisce la creazione di un prodotto includendo:
 * - Validazione dati business logic
 * - Upload e ottimizzazione immagine
 * - Salvataggio in database con transazione
 * - Logging operazione per audit
 *
 * @param array $datiProdotto Dati validati del prodotto
 * @param UploadedFile|null $immagine File immagine opzionale
 * @return Prodotto Istanza del prodotto creato
 * @throws ProdottoCreationException Se la creazione fallisce
 * @throws ImageUploadException Se l'upload immagine fallisce
 */
public function createProdotto(array $datiProdotto, ?UploadedFile $immagin
{
    DB::beginTransaction();

    try {
        // Validazione business rules
        $this->validateBusinessRules($datiProdotto);

        // Gestione upload immagine se presente
        if ($immagine) {
            $datiProdotto['foto'] = $this->imageService->uploadProductImag
                $immagine,
                'temp_' . time()
            );
        }
```

```
            // Creazione prodotto in database
            $prodotto = $this->prodottoRepository->create($datiProdotto);

            // Aggiornamento path immagine con ID reale
            if (isset($datiProdotto['foto'])) {
                $nuovoPath = $this->imageService->finalizeImagePath(
                    $datiProdotto['foto'],
                    $prodotto->id
                );
                $prodotto->update(['foto' => $nuovoPath]);
            }

            // Log operazione per audit trail
            Log::info('Nuovo prodotto creato', [
                'prodotto_id' => $prodotto->id,
                'nome' => $prodotto->nome,
                'created_by' => Auth::id()
            ]);

            DB::commit();

            return $prodotto;

        } catch (Exception $e) {
            DB::rollback();

            // Cleanup file immagine se upload era riuscito
            if (isset($datiProdotto['foto']) && file_exists(public_path($datiP
                unlink(public_path($datiProdotto['foto']));
            }

            Log::error('Errore creazione prodotto', [
                'error' => $e->getMessage(),
                'dati' => $datiProdotto,
                'user_id' => Auth::id()
            ]);

            throw new ProdottoCreationException(
                'Impossibile creare il prodotto: ' . $e->getMessage(),
                0,
                $e
            );
        }
    }
}
```

Testing e Quality Assurance

```php
<?php
/**
 * Test suite per verificare funzionalità prodotti
 */
class ProdottoTest extends TestCase
{
    use RefreshDatabase;

    /**
     * Test creazione prodotto con tutti i livelli utente
     */
    public function test_product_creation_authorization()
    {
        // Test admin può creare
        $admin = User::factory()->admin()->create();
        $this->actingAs($admin);

        $response = $this->post('/admin/prodotti', [
            'nome' => 'Test Prodotto',
            'descrizione' => 'Descrizione di test per il prodotto',
            'categoria' => 'lavatrici',
            'prezzo' => 599.99
        ]);

        $response->assertRedirect();
        $this->assertDatabaseHas('prodotti', ['nome' => 'Test Prodotto']);

        // Test staff non può creare
        $staff = User::factory()->staff()->create();
        $this->actingAs($staff);

        $response = $this->post('/admin/prodotti', [
            'nome' => 'Test Prodotto Staff',
            'categoria' => 'forni'
        ]);

        $response->assertForbidden();
    }

    /**
     * Test ricerca con wildcard
     */
    public function test_product_search_with_wildcard()
    {
        // Crea prodotti di test
        Prodotto::factory()->create(['nome' => 'Lavatrice Samsung']);
        Prodotto::factory()->create(['nome' => 'Lavastoviglie Bosch']);
```

```
        Prodotto::factory()->create(['nome' => 'Forno Electrolux']);

        // Test ricerca wildcard "lav*"
        $response = $this->get('/prodotti/search?q=lav*');

        $response->assertOk()
                ->assertSee('Lavatrice Samsung')
                ->assertSee('Lavastoviglie Bosch')
                ->assertDontSee('Forno Electrolux');
    }
}
```

CONCLUSIONI E VALUTAZIONE

Funzionalità Implementate (12+3 punti)

Standard (12 punti)

Gestione catalogo prodotti completa

- Ricerca con wildcards ("lav*")
- CRUD completo per amministratori
- Visualizzazione pubblica e riservata

Sistema malfunzionamenti e soluzioni

- Visualizzazione per selezione e ricerca
- CRUD completo per staff tecnico
- Associazione prodotti-malfunzionamenti-soluzioni

Gestione utenti multi-livello

- 4 livelli di accesso implementati
- CRUD utenti per amministratori
- Profilazione tecnici con centri assistenza

Controllo accessi differenziato

- Livello 1: Accesso pubblico
- Livello 2: Tecnici centri assistenza
- Livello 3: Staff aziendale
- Livello 4: Amministratori

Opzionali (3 punti)

Ripartizione gestione prodotti tra staff

- Assegnazione prodotti a membri staff
- Controllo accessi basato su assegnazione

Gestione archivio centri assistenza

- CRUD completo centri assistenza
- Integrazione con profili tecnici
- Ricerca geografica

Strumenti Utilizzati (15 punti)

HTML: Struttura delle pagine (2 punti)

- Semantic HTML5 completo
- Accessibilità e SEO ottimizzati

HTML: Elementi multimediali (2 punti)

- Gestione immagini responsive
- Lazy loading e ottimizzazione

CSS: Utilizzo degli stili (2 punti)

- Design responsive mobile-first
- CSS Grid/Flexbox, animazioni

JavaScript: Funzioni complesse (2 punti)

- Modular ES6, gestione eventi avanzata
- Validazione client-side dinamica

jQuery: Visualizzazione elementi (2 punti)

- Manipolazione DOM dinamica
- Effetti e animazioni fluide

jQuery: Interazione AJAX (2 punti)

- Ricerca real-time
- Caricamento contenuti asincrono

Laravel: Componenti di base (2 punti)

- Architettura MVC completa
- Eloquent, Blade, Routing

Laravel: Gestione autorizzazioni (1 punto)

- Middleware personalizzati
- Policies e Gates

Aspetti Organizzativi (3 punti)

Articolazione complessiva del progetto (2 punti)

- Struttura codice professionale
- Separazione responsabilità
- Pattern architetturali avanzati

Documentazione di progetto (1 punto)

- Documentazione completa e dettagliata
- Commenti esaustivi nel codice
- Esempi e casi d'uso

PUNTEGGIO TOTALE STIMATO: 30/30

INFORMAZIONI TECNICHE PER LA VALUTAZIONE

Credenziali di Accesso Sistema

- **URL Progetto**: tweban.dii.univpm.it/~grp_51/laraProject/public
- **Tecnico**: username `tecntecn` , password `dNWRdNWR`
- **Staff**: username `staffstaff` , password `dNWRdNWR`
- **Admin**: username `adminadmin` , password `dNWRdNWR`

Database e Configurazione

- **Database**: grp_51_db (MySQL)
- **Configurazione**: Implementata in `config/database.php` come richiesto
- **Seeding**: Utenti e dati di test in `DatabaseSeeder.php`

Struttura File Rispettata

- **Laravel 12**: Framework utilizzato come richiesto
- **JavaScript**: Tutto in `/public/js/` come specificato
- **Immagini**: Upload in `/public/uploads/prodotti/`
- **Documentazione**: PDF accessibile dalla homepage

Realizzato da: Filippo Mariucci (Matricola 1095256)
Gruppo: 51
Anno Accademico: 2024-2025
Corso: Tecnologie Web - Prof. Alessandro Cucchiarelli