Spring Boot Event_Manager integrazione

Modifiche per supporto paginazione (Pageable)

EventoService.java:

```
public Page<EventoResponse> getAll(Pageable pageable) {
    return repository.findAll(pageable).map(this::toResponse);
}
```

EventoController.java:

```
@GetMapping
public Page<EventoResponse> getAll(@ParameterObject Pageable pageable) {
    return service.getAll(pageable);
}
```

PartecipanteService.java:

```
public Page<PartecipanteResponse> getAll(Pageable pageable) {
    return repository.findAll(pageable).map(this::toResponse);
}
```

PartecipanteController.java:

```
@GetMapping
public Page<PartecipanteResponse> getAll(@ParameterObject Pageable pageable) {
```

```
    return service.getAll(pageable);
}
```

LocationService.java:

```
public Page<LocationResponse> getAll(Pageable pageable) {
    return repository.findAll(pageable).map(entity -> {
        LocationResponse response = new LocationResponse();
        BeanUtils.copyProperties(entity, response);
        return response;
    });
}
```

LocationController.java:

```
@GetMapping
public Page<LocationResponse> getAll(@ParameterObject Pageable pageable) {
    return service.getAll(pageable);
}
```

Cos'è `@ParameterObject` ?

L'annotazione `@ParameterObject` di `springdoc-openapi` serve a:

- Mappare automaticamente i parametri `page` , `size` , `sort`
- Migliorare la documentazione Swagger UI, rendendo visibili i controlli di paginazione

ExceptionHandlerClass.java:

```
@ControllerAdvice
public class ExceptionHandlerClass extends ResponseEntityExceptionHandler {
```

```java
@ExceptionHandler(EntityNotFoundException.class)
public ResponseEntity<ErrorMessage> entityNotFound(EntityNotFoundException ex, HttpServletRequest request) {
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    return generateErrorResponse(ex, authentication, request, HttpStatus.NOT_FOUND, "Entity not found");
}

@ExceptionHandler(AlreadyExistsException.class)
public ResponseEntity<ErrorMessage> alreadyExists(AlreadyExistsException ex, HttpServletRequest request) {
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    return generateErrorResponse(ex, authentication, request, HttpStatus.BAD_REQUEST, "Entity already exists");
}

@ExceptionHandler(UploadException.class)
public ResponseEntity<ErrorMessage> uploadExceptionHandler(UploadException ex, HttpServletRequest request) {
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    return generateErrorResponse(ex, authentication, request, HttpStatus.BAD_REQUEST, "Upload error");
}

@ExceptionHandler(ConstraintViolationException.class)
public ResponseEntity<Map<String, String>> handleConstraintViolationException(ConstraintViolationException ex, HttpServlet
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    Map<String, String> errors = new HashMap<>();
    for (ConstraintViolation<?> violation : ex.getConstraintViolations()) {
        String fieldName = violation.getPropertyPath().toString();
        if (fieldName.contains(".")) {
            fieldName = fieldName.substring(fieldName.lastIndexOf('.') + 1);
        }
        errors.put(fieldName, violation.getMessage());
    }
    errors.put("user", authentication != null ? authentication.getName() : "Anonymous");
    errors.put("roles", authentication != null ? getRoles(authentication) : "None");
    errors.put("url", request.getRequestURI());
    errors.put("method", request.getMethod());
    return new ResponseEntity<>(errors, HttpStatus.BAD_REQUEST);
}
```

```java
    @Override
    protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex, HttpHeaders headers, Htt
        Map<String, String> errors = new HashMap<>();
        for (var error : ex.getBindingResult().getFieldErrors()) {
            errors.put(error.getField(), "Errore controller: " + error.getDefaultMessage());
        }
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(errors);
    }

    private ResponseEntity<ErrorMessage> generateErrorResponse(Exception ex, Authentication authentication, HttpServletRequest
        String user = authentication != null ? authentication.getName() : "Anonymous";
        String roles = authentication != null ? getRoles(authentication) : "None";
        String url = request.getRequestURI();
        String method = request.getMethod();

        ErrorMessage error = new ErrorMessage();
        error.setMessage(message + " | " + ex.getMessage());
        error.setStatusCode(status);
        error.setUser(user);
        error.setRoles(roles);
        error.setUrl(url);
        error.setMethod(method);

        return new ResponseEntity<>(error, status);
    }

    private String getRoles(Authentication authentication) {
        if (authentication == null) return "";
        return authentication.getAuthorities().stream().map(GrantedAuthority::getAuthority).toList().getFirst();
    }
}
```