Integration Testing

Integration Testing The Holiday Creation Process

Arthur Clarkson

4th September 2024

Integration tests assess an application's components at a higher level compared to unit tests. While unit tests focus on individual software components, such as specific class methods, integration tests verify that multiple components interact correctly and produce the expected outcome. These tests can involve all the elements needed to fully process a request.

Integration tests typically cover broader aspects of the application, such as:

- Database
- File system
- Network devices
- Request-response pipeline

Unlike unit tests, which rely on mock objects or fakes to stand in for actual infrastructure components, integration tests:

- Use the real components the application will use in production
- Involve more code and data handling
- Take longer to execute.

To make sure the technical specification was followed, I created this integration test using the NUnit framework:

It generates the holiday entry items, it then passes them into the main `CreateHolidayEntriesForUser` method. This method then calls all the smaller software components required to create holiday entries for a user, including the `CreateHolidayCalendarEvent` method I unit tested. It then queries the database to see if the timesheet entries and calendar events have been created with the correct information.

```
[Test]
public void IntegrationTest_CreateHolidayEntriesForUser()
{
        int userId = 2120;

        var testCase = new
        {
                StartDate = new DateTime(2024, 07, 15),
                EndDate = new DateTime(2024, 07, 18),
                Duration = HolidayEntryItemDurationEnum.AllDay,

                ExpectedDescription = "aclarkson HOL Mon - Thu",
                ExpectedNumberOfTimesheetEntries = 4,
                ExpectedNumberOfCalendarEvents = 1,
                ExpectedTimsheetHoursPerDay = 7.5
        };

        List<HolidayEntryItem> holidayEntryItems = new();

        DateTime currentDate = testCase.StartDate;
        while(currentDate <= testCase.EndDate)
        {
                var holidayEntryItem = new HolidayEntryItem
                {
                        Date = currentDate,
                        Duration = testCase.Duration
                };

                holidayEntryItems.Add(holidayEntryItem);
                currentDate = currentDate.AddDays(1);
        }

        _userController.CreateHolidayEntriesForUser(holidayEntryItems, userId);

        CalendarEvent? calendarEventRecord = _dataContext.CalendarEvents
                .Where(ce => ce.Description == testCase.ExpectedDescription)
                .Where(ce => ce.PrimaryContactUserId == userId)
                .Where(ce => ce.CalendarId == 1)
                .Where(ce => ce.StartDate == testCase.StartDate)
                .Where(ce => ce.EndDate == testCase.EndDate)
                .FirstOrDefault();



        bool calendarEventRecordNotFound = calendarEventRecord is null;
        if(calendarEventRecordNotFound)
        {
                Assert.Fail("The calendar event record not found");
        }

        List<Timesheet> timesheetEntries = _dataContext.Timesheets
                .Where(te => te.Userid == userId)
                .Where(te => te.Entrydate >= testCase.StartDate)
                .Where(te => te.Entrydate <= testCase.EndDate)
                .Where(te => te.Project.Jobcode == "HOL")
                .ToList();

        Assert.Multiple(() =>
        {
                Assert.That(calendarEventRecord?.Description, Is.EqualTo(testCase.ExpectedDescription));
                Assert.That(timesheetEntries,
Has.Count.EqualTo(testCase.ExpectedNumberOfTimesheetEntries));

                foreach (var timesheetEntry in timesheetEntries)
                {
                        Assert.That(timesheetEntry.Hours,
Is.EqualTo(testCase.ExpectedTimsheetHoursPerDay));
                }
        });
}
```